Fast Algorithms for Mining Co-evolving Time Series

Lei Li

September 2011

CMU-CS-11-127

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Christos Faloutsos, Chair
Nancy Pollard
Eric P. Xing
Jiawei Han, University of Illinois at Urbana-Champaign

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

This research was sponsored by the National Science Foundation under grant numbers DBI-0640543 and IIS-0326322, DOE/NNSA under grant number DE-AC52-07NA27344, the Air Force Research Laboratory under grant number FA8750-11-C-0115, the Army Research Laboratory under grant number W911NF-09-2-0053, and iCAST. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: time series forecasting, dimensionality reduction, feature extraction, clustering, parallel algorithms, linear dynamical systems, motion capture, data center energy efficiency

to my parents

Abstract

Time series data arise in many applications, from motion capture, environmental monitoring, temperatures in data centers, to physiological signals in health care. In the thesis, I will focus on the theme of learning and mining large collections of co-evolving sequences, with the goal of developing fast algorithms for finding patterns, summarization, and anomalies. In particular, this thesis will answer the following recurring challenges for time series:

1. *Forecasting and imputation*: How to do forecasting and to recover missing values in time series data?
2. *Pattern discovery and summarization*: How to identify the patterns in the time sequences that would facilitate further mining tasks such as compression, segmentation and anomaly detection?
3. *Similarity and feature extraction*: How to extract compact and meaningful features from multiple co-evolving sequences that will enable better clustering and similarity queries of time series?
4. *Scale up*: How to handle large data sets on modern computing hardware?

We develop models to mine time series with missing values, to extract compact representation from time sequences, to segment the sequences, and to do forecasting. For large scale data, we propose algorithms for learning time series models, in particular, including Linear Dynamical Systems (LDS) and Hidden Markov Models (HMM). We also develop a distributed algorithm for finding patterns in large web-click streams. Our thesis will present special models and algorithms that incorporate domain knowledge. For motion capture, we will describe the natural motion stitching and occlusion filling for human motion. In particular, we provide a metric for evaluating the naturalness of motion stitching, based which we choose the best stitching. Thanks to domain knowledge (body structure and bone lengths), our algorithm is capable of recovering occlusions in mocap sequences, better in accuracy and longer in missing period. We also develop an algorithm for forecasting thermal conditions in a warehouse-sized data center. The forecast will help us control and manage the data center in a energy-efficient way, which can save a significant percentage of electric power consumption in data centers.

Acknowledgments

I am indebted to many people for the help along the journey.

First, I would like to thank my advisor Christos Faloutsos for his advise and endless support. This thesis would not have been possible without his guidance, encouragement, and help. He has always been a source of expertise, igniting "crazy ideas", nurturing eyeball attracting project names, and completing social triangles.

I also thank my committee: Nancy Pollard has been a great collaborator and mentor on computer graphics and character animation, helped me pick out promising and important problems, and provided many exciting discussions. Eric Xing taught me two courses in machine learning and graphical models, and provided much personal and career guidance beyond research techniques. Jiawei Han discussed about my research every time we met in conferences and elsewhere and made great suggestions from new perspectives.

I thank the amazing group of collaborators and co-authors whom I have been fortunate to work with: Leman Akoglu, Tina Eliassi-Rad, Bin Fu, Wenjie Fu, Brian Gallagher, Fan Guo, Donna Haverkamp, Keith Henderson, Ellen Hughes, Danai Koutra, Chieh-Jan Mike Liang, Jie Liu, Siyuan Liu, David Lo, Koji Maruhashi, Yasuko Matsubara, James McCann, Todd C. Mowry, Suman Nath, Jia-Yu (Tim) Pan, B. Aditya Prakash, Nancy Pollard, Marcela Xavier Ribeiro, Yasushi Sakurai, Pedro Stancioli, Jimeng Sun, Andreas Terzis, Hanghang Tong, Eric Xing, and Wanhong Xu. Jim has been a great collaborator and supportive friend, and helped me with many insightful ideas and feedbacks. Thanks to my house mates, Wanhong and Fan, with whom I had many relaxing chats. Thanks to Aditya, with whom I had very enjoyable iterations of discussion and seen the growth of sparkling ideas. Tim has been a great helper all along the way since my first few days at CMU and later during my internship at Google. I have spent great summers at Google, Microsoft Research, and IBM TJ Watson Research Center. Jie and Suman lead me to the area of data center monitoring with wireless sensor networks. Thank you for taking me to a production data center and exposing me to real world problems. I would also like to thank Jimeng for his support at IBM and his every effort in connecting me to other researchers and physicians. Many thanks extends to members and colleagues at the ads-spam group at Google, the networked embedded computing group at MSR, and the healthcare transform group at IBM.

I would like to thank colleagues at Database group, Deepayan Chakrabarti, Polo Chau, Debabrata Dash, U Kang, Jure Leskovec, Mary McGlohon, Ippokratis Pandis, Spiros Papadimitriou, Stratos Papadomanolakis, Minglong Shao, and Charalampos Tsourakakis, and visitors, Ana Paula Appel, Robson Cordeiro, Sang-Wook Kim, Sunhee Kim, Kensuke Onuma, and Hyungjeong Yang. They have helped significantly through group discussions, feedback on practice talks and papers. Thanks to Ziv Bar-Joseph, who illustrated how to teach a large graduate course when I was TA for the machine learning course. Thanks to Deborah Cavlovich, Catherine Copetas, Joan Digney, Karen Lindenfelser, Michelle Martin, Diane Stidle, Marilyn Walgora, Charlotte Yano, and many staffs at SCS who have been constantly providing administrative support including planning trips and meetings, processing reimbursement, designing posters, and filling various forms. Charlotte and Marilyn always responded to my requests at "cutting down the wire".

Of course, life would not be the same without my friends at Computer Science department, SCS and their “extended families”: Ning Chen, Xi Chen, Yuxin Deng, Duo Ding, Bin Fan and Shuang Su, Sicun Gao, Lie Gu, Jenny Han, Jingrui He, Li Huang, Zhaoyin Jia and Jing Xia, Junchen Jiang, Xiaoqian Jiang, Hongwen Kang and Xiaolan Shu, Ruogu Kang, Yunchuan Kong, Zhenzhen Kou, Boyan Li, Nan Li, Wen Li, Yan Li, Yanlin Li, Jialiu Lin, Bin Liu, Xi Liu, Liu Liu, Yandong Liu, Yanjin Long, Yong Lu, Yilin Mo, Juan Peng, Yang Richard Peng, Kriti Puniyani, Zhengwei Qi, Kai Ren, Dafna Shahaf, Runting Shi, Yanxin Shi, Le Song, Ming Sun, Huimin Tan, Likun Tan, Xi Tan, Kanat Tangwongsan, Yuandong Tian, Tiankai Tu, Vijay Vasudevan, Xiaohui Wang, Xuezhi Wang, Yiming Wang, Chenyu Wu, Chuang Wu, Yi Wu, Guangyu Xia, Guang Xiang, Lin Xiao, Liang Xiong, Hong Yan and Xiaonan Zhang, Rong Yan and Yan Liu, Eric You, Jun Yang, Junming Yin, Xin Zhang, Yi Zhang, Yimeng Zhang, Le Zhao, Hua Zhong and Min Luo, Feng Zhou, Yuan Zhou, Zongwei Zhou, Haiyi Zhu, Yangbo Zhu, and Jun Zhu. They have played a critical role in this work through helpful discussions, feedback on dry-runs, comments on papers, collaboration in coursework and other Ph.D. pursuits, and moral support.

I would also like to thank the SJTU and SCZ gang: Shenghua Bao, Chenxi Lin, Qiaoling Liu, Yunfeng Tao, Guirong Xue, Lei Zhang, and Jian Zhou, for collaboration on semantic web research and mentoring at Apex lab; Yaodong Zhang and Linji Yang for co-developing the Fatworm DBMS; Hao Lv, Yunfeng Tao, Kewei Tu, Qiqi Yan, Xi Zhang and Lin Zhu for much helpful advice on many personal decisions; Yong Yu, Enshao Shen, Yuxi Fu, John Dezhang Lin and Liren Wang for being great advisor and mentors on my study, research and personal development; and all of my fellow classmates, Erdong Chen, Wenyuan Dai, Zheren Hu, Wei Guo, Xiaohui Liang, Hao Yuan, Congle Zhang, and many friends . . . . My days during ACM class have really paved ways for my later research endeavour. Sincere thanks to Mr. Wen Cao who lead me to computer programming and design of algorithm in my very early days, and to Yingliu Chen, Yuan Li, Yun Shen, Jingjing Tu, Jing Wang, Xuan Zhang, and many friends and academic brothers and sisters who helped me along the way. Xuan has always offered encouragement and tremendous help along the journey.

Finally, I thank my family: my father for leading me to the joy of mathematics when in childhood; my mother for much patience and support of my each decisions; and my grandma for endless love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many scientific applications generate numerous time series data, i.e. sequences of time stamped numerical or categorical values. Yet there are not a set of readily tools for analyzing the data and exploiting the patterns (e.g. dynamics, correlation, trend and anomalies) in the sequences. Finding patterns in such collections of sequences is crucial for leveraging them to solve real-world, domain specific problems, for motivation examples: (1) motion capture, where analyzing databases of motion sequences helps create animation of natural human actions in movie and game industry ($57 billion business) and design assisting robots; (2) environmental monitoring (e.g. chlorine level measurements in drinking water systems), where the goal is to alert households to safety issues in environments; (3) data center monitoring, with the goal of saving energy consumption for better sustainability and lower cost ($4.5 billion cost in 2006); and (4) computer network traffics, where the goal is to identify intrusion or spams in computer networks.

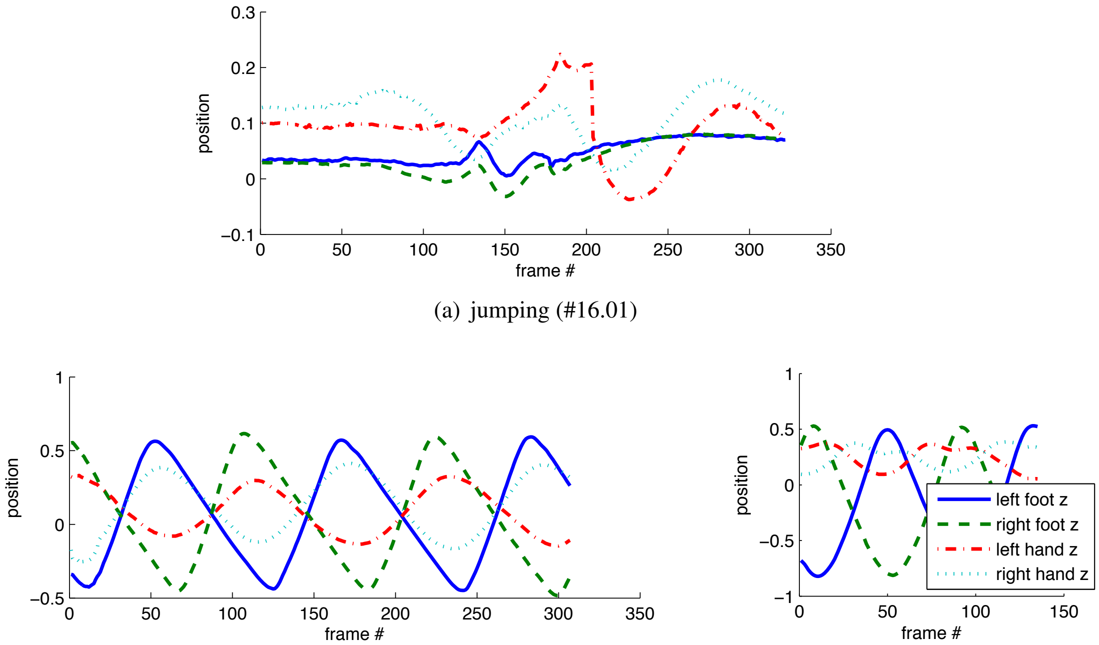
This thesis is motivated by these applications. The problems studied in this thesis are abstracted from the common challenges across these applications. In the following, we will first present a few motivating applications and their time series data; we will describe the problems and general approaches to the challenges. We will both investigate algorithms that are versatile in diverse applications and mining tasks, and also study domain specific scenarios where domain knowledge should be integrated with general models.

1.1 Motivation

Time sequences appear in numerous applications, like sensor measurements [Jain et al., 2004], mobile object tracking [Kollios et al., 1999], data center monitoring [Reeves et al., 2009], computer network monitoring [Sun et al., 2007], motion capture sequences [Keogh et al., 2004], environmental monitoring (like automobile traffic [Papadimitriou et al., 2003] and chlorine levels in drinking water [Papadimitriou et al., 2005, Leskovec et al., 2007]) and many more.

In these scenarios, it is very important to understand the patterns in the data such as correlation and evolving behavior. Better patterns will help make predictions, compress and detect anomalies. Our goal is to develop algorithms for mining and summarizing any time series data, and we list here a few motivating applications.

Modeling human motion sequences Motion capture (mocap) is a technique for modeling human motion. CMU researchers have built several large databases of human motions [CMU, a]. Such databases are used to create models of human motion for many applications such as movies, computer games, medical care, sports and surveillance among others. The revenue merely in global video game and interactive entertainment industry is $62.7 billion and expected to be $65 billion in 2011 [Reuters, 2011]. Besides the monetary benefits, research on motion capture databases has increasing number of applications in improving the quality of life. For example, there is already a motion capture database with various tasks performed in the kitchen [CMU, b], and analyzing motions in such a database will help design robots that can, say, prepare a balanced diet for the elderly [la Torre Frade et al., 2008].



(a) jumping (#16.01)

(b) walking (#16.22)

(c) running (#16.45)

Figure 1.1: Motion capture sequences: marker positions in body center coordinate versus time. The curves are z-coordinates of four markers: left foot (solid line), right foot (dashed line), left hand (dash-dotted line) and right hand (dotted line). The data is from [CMU, a].

Figure 1.1 shows three example motion sequences for jumping, walking and running. We are particularly interested in the following important problems:

- *Naturalness:* How to create new and natural human motions from a motion capture database?
- *Similarity:* How to index a large database of motion capture clips and find similar motions?
- *Missing values and imputation:* How to recover the occlusion that is common in mocap sequences?

Environmental monitoring Wireless sensors are deployed in many environmental monitoring applications, such as monitoring chlorine levels in drinking waters systems [Papadimitriou et al., 2005], water levels in rivers, and automobile traffic in major infrastructure roads [Papadimitriou et al., 2003]. Figure 1.2 shows sample chlorine level data. Sensor data are usually in streaming fashion, and well suited in the context of our time series mining algorithms.

Typical problems in sensor data mining include:

- *Summarization:* How to summarize the data to reduce the transmission over network? Since in wireless sensors, data transmission consumes much of battery energy.
- *Patterns and anomallies:* How to detect anomalies in sensor data? For example, detecting the a leak or an attack in drinking water by monitoring the chlorine levels.
- *Missing values and imputation:* How to find incorrect observations or recover missing values in sensor data? It is common to have missing observations due to various factors, say, low battery or radio frequency (RF) error.

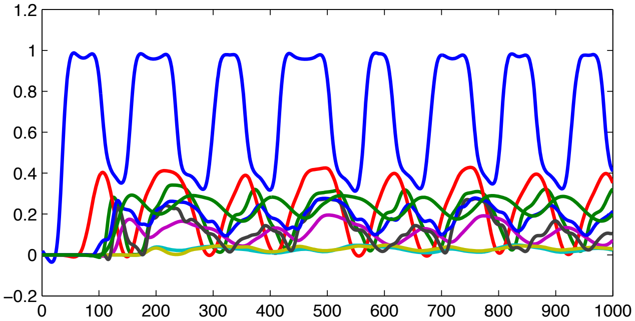


Figure 1.2: Sample snippets of Chlorine concentration versus time, in drinking water for eight households. The data is from [VanBriesen].

Data center monitoring Modern cloud computing applications, like Google's search engine for the whole web, heavily rely on large computer clusters (e.g. 5000 servers as in [Fan et al., 2007]). Thus many companies and labs build, manage their own data centers, and study their power efficiency and reliability [Hoke et al., 2006, Barroso and Hölzle, 2009, Patnaik et al., 2009]. The number and the scale of data centers grow tremendously, and such growth of data centers creates an increasing demand of new electric power plants. It is reported by EPA that in 2006 US data centers consume 61 billion kilo-watt-hours of electricity, which amounts to 1.5 percent of US total electricity consumption that year, or 4.5 billion dollars in expense [EPA, 2007]. With such growing trend, it is projected that by 2011 the expense of electricity in data centers will reach $ 7.4 billion, and ten more power plants have to be built to meet the additional electricity needs. If we could save 2 percent of the energy consumption, we would save about $150 million in electricity expense each year.

As expected, there are plenty of streaming data in datacenters, e.g. segments of measurements of temperatures, humidity, workload and server utilization. The challenge is, how to design algorithms and systems that automatically find patterns in such data streams and use the findings to better control the datacenters in order to save energy.

Computer network traffic Another important time series application is computer communication streams, such as port to port tcp/ip traffic [Sun et al., 2007] and web click streams [Liu et al., 2009]. Understanding such sequencies is crucial to the cyber security. Figure 1.3 shows a sample BGP (Border Gateway Protocol) traffic sequence for a router at Washington DC [Feamster et al.]. We are particularly interested in the following problems:

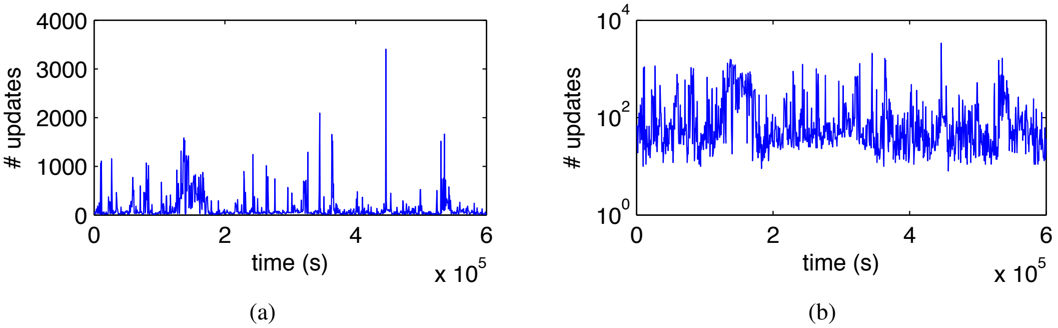


Figure 1.3: Sample snippets from BGP router data at Washington DC: number of updates versus time (in seconds). Notice the original sequence is bursty with no periodicity (shown in part (a)), thus we take the logarithm (shown in part (b)). No obvious patterns, in neither (a) nor (b). Data is from [Feamster et al.].

- *Patterns:* How to find patterns in such time series? How to group similar traffic patterns together? The challenge lies in the bursty nature of these data sequences.
- *Anomallies:* How to identify intrusion/anomalies in such computer network traffic data?

Medical time sequences Medical domain generates numerous amount of data sequences, which received little attention from machine learning or data mining area. In particular, patients in intensive care units (ICU) are often monitored in real time with multiple medical instruments, yielding many indicative physiological sequences such as blood pressure (BP), heart rate (HR), electrocardiogram signals (ECG). Automatic mining patterns and prediction in those medical data can support medical diagnosis. For example, patients with acute hypotensive episodes (AHE, i.e. sudden drop in blood pressure) have twice as high fatality rate as those patients without AHE. Algorithms that can predict AHE events in advance would greatly help those patients and medical doctors. Figure 1.4 shows a segment of average blood pressure for one patient.

Typical problems include:

- *Similarity:* How to find similar patients with physiological records? So that we can recommend similar treatment.
- *Forecasting:* How to predict acute events for patients based on continuous monitoring of medical signals?

1.2 Thesis overview and contributions

Across the motivating settings described above, the recurring research challenges for time series mining are:

1. *Forecasting and imputation*: How to do forecasting and to recover missing values in time series data?

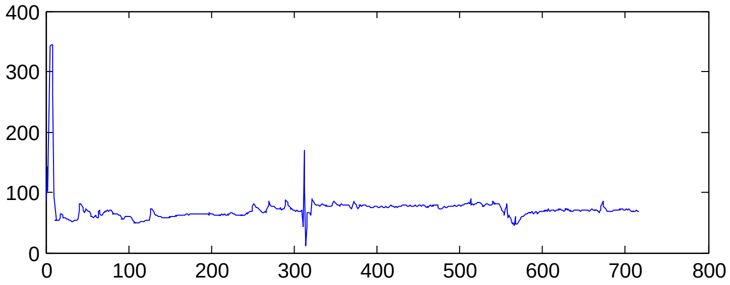


Figure 1.4: A sample snippet of average blood pressure (ABP) for one patient.

2. *Pattern discovery and summarization*: How to identify the patterns in the time sequences that would facilitate further mining tasks such as compression, segmentation and anomaly detection?
3. *Similarity and feature extraction*: How to extract compact and meaningful features from multiple co-evolving sequences that will enable better clustering and similarity queries of time series?
4. *Scale up*: How to handle large data sets on modern computing hardware?

We want to highlight that in general pattern discovery and feature extraction are closely related. Once we discover patterns (like cross-correlations, auto-correlations) in time series, we can do (a) forecasting (by continuing pattern trends), (b) summarization (by a compact representation of the pattern, like a covariance matrix, or auto-regression coefficients), (c) segmentation (by detecting a change in the observed pattern), and (d) anomaly detection (by identifying data points that deviating too much from what the pattern predicts). Similarly, feature extraction are closely related to data mining tasks. Once we have good features, we can do (a) clustering of similar time sequences, (b) indexing large time series database, and (c) visualizing long time series, plotting them as points in a lower-dimensional feature space.

In the thesis, we will focus on the theme of mining large collections of co-evolving sequences, with the goal of developing fast algorithms for finding patterns, summarization, and anomalies. In answering four questions above, the algorithms proposed in the thesis will be organized into three categories: part (i), general algorithms for missing value and feature extraction (Chapter 3, 4 and 5); part (ii), parallel algorithms learning large data set (Chapter 6, 7 and 8); and part (iii) domain specific algorithms for modeling human motion and data centers. Table 1.1 lists all the problems in time series mining covered in this thesis and proposed solutions.

Chapter 2 will review the basic models, tools and major algorithms in time series mining. Chapter 3 will describe the missing value problem and algorithms for recovering missing values, compression and segmentation. Chapter 4 will describe a feature extraction method, PLiF, and will demonstrate its application in time series clustering and compression. Chapter 5 will describe a unified graphical model for time series, with more succinct representation: complex linear dynamical systems (CLDS). Chapter 6 will describe CAS-LDS, a parallel algorithm for learning linear dynamical systems (LDS). Chapter 7 will describe CAS-HMM, a parallel algorithm for learning hidden Markov models (HMM). Chapter 9 will describe a metric for evaluating the quality of motion stitching, and an algorithm for generating natural human motions. Chapter 10 will describe BoLeRO, a specific model to recover the occlusion in the motion data. BoLeRO will exploit the domain knowledge and structural information to improve the occlusion filling. Chapter 11 will describe ThermoCast, a model for forecasting the thermal dynamics in data center server room. ThermoCast could predict the server temperatures based on workload, therefore it can be used in identify thermal alarms in data centers.

Contributions

- We developed the algorithms that outperform the best competitors in missing value recovery for time series. It can also achieve the highest compression ratio within a given error;
- We developed an effective algorithm and a uniform model for feature extraction. It can achieve the best clustering accuracy.
- We developed the first parallel algorithm for learning Linear Dynamical Systems. It achieves linear speed up on both super-computers and multicore desktop machines.

Impact on real world applications

- Our algorithms have been successfully applied in motion capture practice, to generate realistic human motions and recover occluded motion sequences;
- Our algorithms have been applied in data centers at a large company and a university, and help improve the energy efficiency and reduce the power consumption in data centers.
- Our algorithms have been applied to identify patterns and anomalies in web-clicks.

Table 1.1: Time series mining challenges, and proposed solutions (in italics) covered in the thesis

	mining	parallel learning
general purpose models	1. similarity and feature extraction (*PLiF* and *CLDS*) 2. forecasting and imputation (*DynaMMo*) 3. pattern discovery and summarization (*DynaMMo*, *PLiF*, and *CLDS*)	4. parallel LDS on SMPs (*CAS-LDS*) 5. parallel HMM on SMPs (*CAS-HMM*)
domain specific	6. natural motion stitching (*L-Score*) 7. motion occlusion filling (*BoLeRO*) 8. thermal prediction in data center (*ThermoCast*)	9. web-click stream monitoring (*WindMine*)

Chapter 2

Overview and Background

This chapter will review the basic definitions of time series and time series mining problems. We will review general approaches and models in literature. More specific ones will come later in each chapter.

2.1 Definitions

A time series is a sequence of data points measured at equal time intervals. Unless otherwise noted, we will use $\mathcal{X}$ as the observation data, with T as duration and m as the dimensionality. $\vec{x}_1, \vec{x}_2$ denotes the data at time $t = 1$ and $t = 2$, respectively. For each time tick, $\vec{x}_i$ can be numerical values or categorical values. In particular, we are interested in co-evolving time series, i.e. multi-dimensional correlated sequences of data with equal time stamps. Though our thesis is mainly for multi-dimensional time series of numerical values, we will also mention models for categorical data.

Our work is based on two key properties in those co-evolving time series, dynamics and correlation. Dynamics captures the temporal evolving trend, while the correlation represents the relationship between multiple sequences. For example, markers on body parts are often correlated when an actor is performing. Figure 2.1 shows sequences of xyz-coordinates of the left and right wrist for a walking motion. Note the left wrist sequence and the right one are correlated with similar dynamics. By exploiting both properties, we are able to discover rich patterns from the data. Throughout the whole thesis, we will be presenting specific methods and models for learning patterns from the data and approaches to apply those methods in real applications.

2.2 A survey on time series methods

There is a lot of work on time series analysis, on indexing, dimensionality reduction, forecasting, and parallelization.

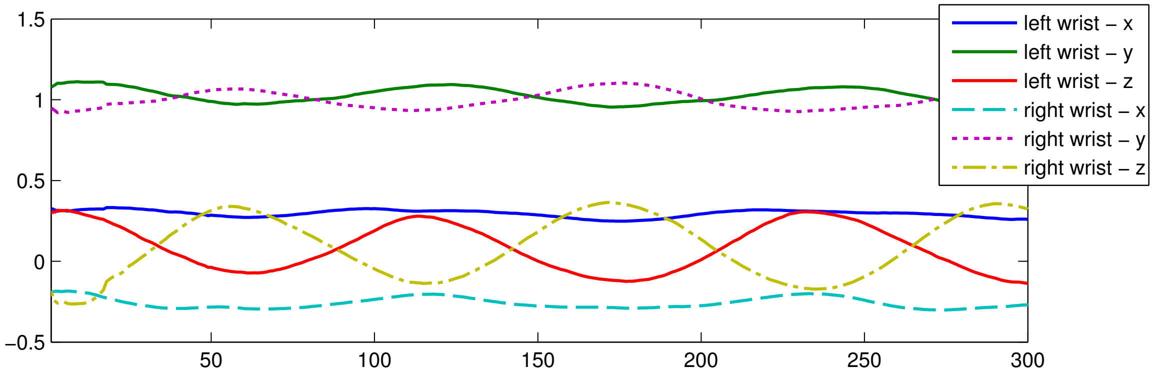


Figure 2.1: A walking motion sequence: xyz-coordinates of the markers on left and right wrist. Note the correlation between left and right wrists.

2.2.1 Indexing, signals and streams

For indexing, the idea is to extract features [Faloutsos and Lin, 1994] and then use a spatial access method. Typical features include the Fourier transform coefficients, wavelets [Gilbert et al., 2001, Jahangiri et al., 2005], piece-wise linear approximations [Keogh et al., 2001]. These are mainly useful for the Euclidean distance, or variations [Rafiei and Mendelzon, 1997, Ogras and Ferhatosmanoglu, 2006]. Indexing for motion databases has also attracted attention, both in the database community (eg., [Keogh et al., 2004]) as well as in graphics (e.g., [Safonova and Hodgins, 2007]).

Typical distance functions are the Euclidean distance and the time warping distance, also known as *Dynamic Time Warping* (DTW) (e.g., see the tutorial by Gunopulos and Das [Gunopulos and Das, 2001]). Wang and Bodenheimer have used *windowed Euclidean distance* to assess the quality of stitched motion segments and proposed an algorithm to select the best transition [Wang and Bodenheimer, 2003]. The original, quadratic-time DTW, has been studied in [Yi et al., 1998], and its linear-time constrained versions (Itakura parallelogram, Sakoe-Chiba band) in [Keogh, 2002, Fu et al., 2005].

There is also vast, recent literature on indexing moving objects [Jensen and Pakalnis, 2007, Mouratidis et al., 2006], as well as streams (e.g., see the edited volume [Garofalakis et al., 2009]). An additional recent application for time series is monitoring a data center [Reeves et al., 2009], where the goal is to observe patterns in order to minimize energy consumption. An equally important monitoring application is environmental sensors [Deshpande et al., 2004, Leskovec et al., 2007].

2.2.2 Dimensionality reduction

There are numerous papers on the topic, with typical methods being PCA [Jolliffe, 2002], SVD/LSI [Dumais, 1994], ICA [Hyvärinen et al., 2001], random projections [Papadimitriou et al., 1998], fractals [Traina et al., 2000]; and a vast literature on feature selection and non-linear dimensionality reduction.

Principal Component Analysis and Singular Value Decomposition

For a data matrix $\mathbf{X}$ (assume $\mathbf{X}$ is zero-centered), SVD computes the decomposition

$$\underbrace{\mathbf{X}}_{n\times m} = \underbrace{\mathbf{U}}_{n\times h} \cdot \underbrace{\mathbf{S}}_{h\times h} \cdot \underbrace{\mathbf{V}^T}_{h\times m}$$

where both $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices, and $\mathbf{S}$ is a diagonal matrix with singular values on the diagonal.

Independent Component Analysis

Unlike looking for orthogonality in PCA, Independent Component Analysis (ICA) compute the independence between components by minimizing the mutual information. To find the directions of minimal entropy the well known fastICA algorithm [Hyvärinen and Oja, 2000] requires us to transform the data set into white space, i.e., the data set must be centered and normalized so that it has unit variance in all directions. This may be achieve from the eigenvalue decomposition of the covariance matrix (i.e., $V \cdot \Lambda \cdot V^T := \Sigma$ where V is an orthonormal matrix consisting of the eigen vectors, and Λ is a diagonal matrix($\Lambda = diag(\lambda_1, \ldots, \lambda_d)$). The matrix $\Lambda^{-1/2}$ is a diagonal matrix with the elements $\Lambda^{-1/2} = diag(\sqrt{1/\lambda_i}, \ldots, \sqrt{1/\lambda_d})$. The fastICA algorithm then determines a matrix B that contains the independent components. This matrix is orthonormal in white space but not in the original space. FastICA is an iterative method that finds $B = (b_1, \ldots, b_d)$ by optimizing the vectors b_i using the following updating rule:

$$b_i := E\{y \cdot g(b_i^T \cdot y)\} - E\{g'(b_i^T \cdot y)\} \cdot b_i \tag{2.1}$$

where $g(s)$ is a non-linear contrast function (such as $tanh(s)$) and $g'(s) = \frac{d}{ds}g(s)$ is its derivative. We denote the expected value with $E\{\ldots\}$. After each application of the update rule to $b_i, \ldots, b_d$, the matrix B is orthonormalized. This is repeated until convergence. The de-mixing matrix A^{-1}, which describes the overall transformation from the original data space to the independent components, can be determined as

$$A^{-1} = B^T\Lambda^{-1/2} \cdot V^T, A = V \cdot \Lambda^{+1/2} \cdot B \tag{2.2}$$

and, since V and B are orthonormal matrices, the determinant of A^{-1} is simply the determinant of $\Lambda^{-1/2}$, i.e,

$$det(A^{-1}) = \prod_{1\leq i\leq d} \sqrt{1/\lambda_i}. \tag{2.3}$$

2.2.3 Multi-resolution methods: Fourier and Wavelets

Mining time series often relies on good features extracted from data sequences. Among the typical features are the Fourier transform coefficients and wavelets [Gilbert et al., 2001, Jahangiri et al., 2005],

The T-point Discrete Fourier Transform (DFT) of sequence $(x_0, \ldots, x_{T-1})$ is a set of T complex numbers c_k, given by the formula

$$c_k = \sum_{t=0}^{\mathrm{T}-1} x_t e^{-\frac{2\pi i}{\mathrm{T}}kt} \qquad k = 0, \ldots, \mathrm{T}-1$$

where $i = \sqrt{-1}$ is imaginary unit.

The c_k numbers are also referred to as the *spectrum* of the input sequence. DFT is powerful in spotting periodicities in a single sequence, with numerous uses in signal, voice, and image processing.

2.2.4 Time series forecasting

Autoregression is the standard first step for forecasting. It is part of the ARIMA methodology, pioneered by Box and Jenkins [Box et al., 1994], and it discussed in every textbook in time series analysis and forecasting (e.g., [Brockwell and Davis, 1987],[Tong, 1990]). [Kalpakis et al., 2001] used autoregression to extract features, using the so-called *cepstrum* method from voice processing.

2.2.5 State space models

Linear Dynamical Systems (LDS), also known as Kalman filters, have been used previously to model multi-dimensional continuous valued time series. Kalman filters and Linear Dynamical Systems are closely related to autoregression, trying to detect hidden variables (like velocity, acceleration) at every time-tick, and use them for forecasting [Harvey, 1990]. In the data mining community, Kalman filters have been proposed for sensor data [Jain et al., 2004] as well as for moving objects [Tao et al., 2004].

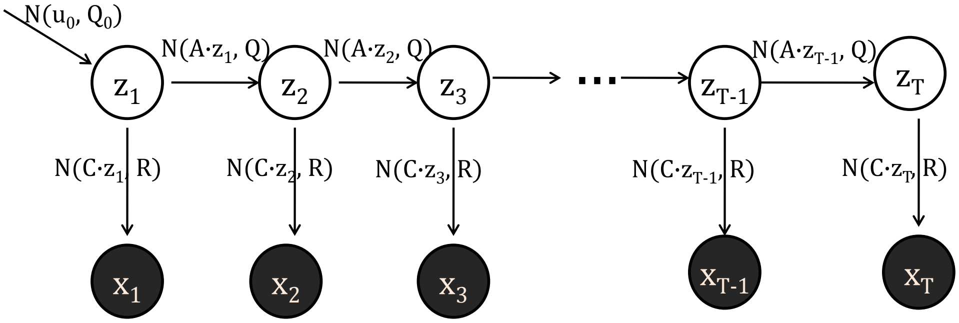


Figure 2.2: Graphical representation of Linear Dynamical Systems

Linear Dynamical Systems can be described in the following equations:

$$\vec{z}_1 = \vec{\mu}_0 + \vec{\omega}_1 \tag{2.4}$$
$$\vec{z}_{n+1} = \mathbf{A}\vec{z}_n + \vec{\omega}_{n+1} \tag{2.5}$$
$$\vec{x}_n = \mathbf{C}\vec{z}_n + \vec{\epsilon}_n \tag{2.6}$$

where $\vec{\mu}_0$ is initial state of the whole system, and the noises follow

$$\vec{\omega}_1 \sim \mathcal{N}(0, \mathbf{Q}_0) \qquad \vec{\omega}_{n+1} \sim \mathcal{N}(0, \mathbf{Q}) \qquad \vec{\epsilon}_n \sim \mathcal{N}(0, \mathbf{R})$$

The model assumes the observed data sequences ($\vec{x}_n$) are generated from the a series of hidden variables ($\vec{z}_n$) with a linear projection matrix $\mathbf{C}$, and the hidden variables are evolving over time with linear transition matrix $\mathbf{A}$, so that next time tick only depends on the previous time tick as in Markov chains. All noises ($\vec{\omega}$'s and $\vec{\epsilon}$'s) arising from the process are modeled as independent Gaussian noises with covariances $\mathbf{Q}_0$, $\mathbf{Q}$ and $\mathbf{R}$ respectively. Figure 2.2 shows the graphical model representation.

Given the observation series, there exist algorithms for estimating hidden variables [Kalman, 1960, Rauch et al., 1965] and EM algorithms for learning the model parameters [Shumway and Stoffer, 1982, Ghahramani and Hinton, 1996], with publicly available implementations[1].

The EM algorithm maximizes $L(\theta)$, the expected log-likelihood defined in Eq. 2.7, iteratively.

In the **E** step, it estimates the posterior distribution of the hidden variables conditioned on the data sequence with fixed model parameters; in the **M** step, it then updates the model parameters by maximizing the likelihood using some sufficient statistics (e.g. mean and covariance) from the posterior distribution.

$$\begin{aligned} L(\theta; \mathcal{X}) &= \mathbb{E}_{\mathcal{Z}|\mathcal{X};\theta}[\log P(\mathcal{X}, \mathcal{Z}; \theta)] \\ &= \mathbb{E}_{\mathcal{Z}|\mathcal{X};\theta}\Bigg[-D(\vec{z}_1, \vec{\mu}_0, \mathbf{Q}_0) - \sum_{t=2}^{\mathrm{T}} D(\vec{z}_t, \mathbf{A}\vec{z}_{t-1}, \mathbf{Q}) - \sum_{t=1}^{\mathrm{T}} D(\vec{x}_t, \mathbf{C}\vec{z}_t, \mathbf{R}) \\ &\qquad - \frac{1}{2}\log|\mathbf{Q}_0| - \frac{\mathrm{T}-1}{2}\log|\mathbf{Q}| - \frac{T}{2}\log|\mathbf{R}|\Bigg] \end{aligned} \tag{2.7}$$

where $D()$ is the square of the Mahalanobis distance, i.e. $D(\vec{x}, \vec{y}, \Sigma) = (\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})$.

2.2.6 Parallel programming for data mining

Data mining and parallel programming receives increasing interest. [Buehrer et al., 2007] develop parallel algorithms for mining terabytes of data for frequent item sets, demonstrating a near-linear scale-up on up to 48 nodes. Reinhardt and Karypis [Reinhardt and Karypis, 2007] used OpenMP[2] to parallelize the discovery of frequent patterns in large graphs, showing excellent speedup of up to 30 processors. [Cong et al., 2005] develop the Par-CSP algorithm that detects closed sequential patterns on a distributed memory system, and report good scale-up on a 64-node Linux cluster. [Graf et al., 2005] developed a parallel algorithm to learn SVM ('*Support Vector Machines*') through cascade SVM. [Collobert et al., 2002] proposed a method to learn a mixture of SVM in parallel. Both of them adopted the idea of splitting dataset into small subsets, training SVM on each, and then combining those SVMs. [Chang et al., 2007] proposed PSVM to train SVMs on distributed computers through approximate factorization of the kernel matrix.

There is also work on using Google's Map-Reduce [Dean and Ghemawat, 2008] to parallelize a set of learning algorithm such as naïve-Bayes, PCA, linear regression and other related algorithms [Chu et al., 2006, Ranger et al., 2007]. Their framework requires the summation form (like dot-product) in the learning algorithm, and hence could distribute independent calculations to many processors and then summarize them together. Unfortunately, the same techniques could hardly be used to learn long sequential graphical models such as Hidden Markov Models and Linear Dynamical Systems (LDS). On the contrary, we will show later our proposed *Cut-And-Stitch* method can achieve *almost linear* speedup for learning LDS on shared memory multiprocessors.

[1] http://people.cs.ubc.ca/∼murphy/software/kalman/kalman.html

[2] http://www.openmp.org

Part I

Theory and Models

Chapter 3

Prediction and Missing Values

Given multiple time sequences with missing values, how to find patterns and fill in missing values? In this chapter, we will describe DynaMMo, which summarizes, compresses, and finds latent variables. The idea is to discover hidden variables and learn their dynamics, making our algorithm able to function even when there are missing values.

We will present experiments on both real and synthetic data sets spanning several megabytes, including motion capture sequences and chlorine levels in drinking water. We show that our proposed DynaMMo method (a) can successfully learn the latent variables and their evolution; (b) can provide high compression for little loss of reconstruction accuracy; (c) can extract compact but powerful features for segmentation, interpretation, and forecasting; (d) has complexity linear on the duration of sequences.

3.1 Introduction

Time series data are abundant in many application areas such as motion capture, sensor networks, weather forecasting, and financial market modeling. The major goal of analyzing these time sequences is to identify hidden patterns so as to forecast the future trends. There exist many mathematical tools to model the evolutionary behavior of time series (e.g. Linear Regression, Auto-Regression, and AWSOM [Papadimitriou et al., 2003]). These methods generally assume completely available data. However, missing observations are hardly rare in many real applications, thus it remains a big challenge to model time series in the presence of missing data.

We propose a method to handle the challenge, with occlusion in motion capture as our driving application. However, as shown in the experiments, our method is capable of handling missing values in diverse settings: sensor data, chlorine levels in drinking water system, and other similar co-evolving sequences.

Motion capture is a technique to produce realistic motion animation. Typical motion capture system use cameras to track passive markers on human actors. However, even when multiple cameras are used, some markers may be out of view – especially in complex motions like handshaking or modern dance. Handling occlusions is currently a manual process, taking hours/days for human experts to fill in the gaps. Figure 3.2 illustrates a case of motion-capture data, with occlusions: A dark cell at row j, and column t denotes a missing value, for that specific time (t-th frame/column) and for that specific joint-angle (j-th row).

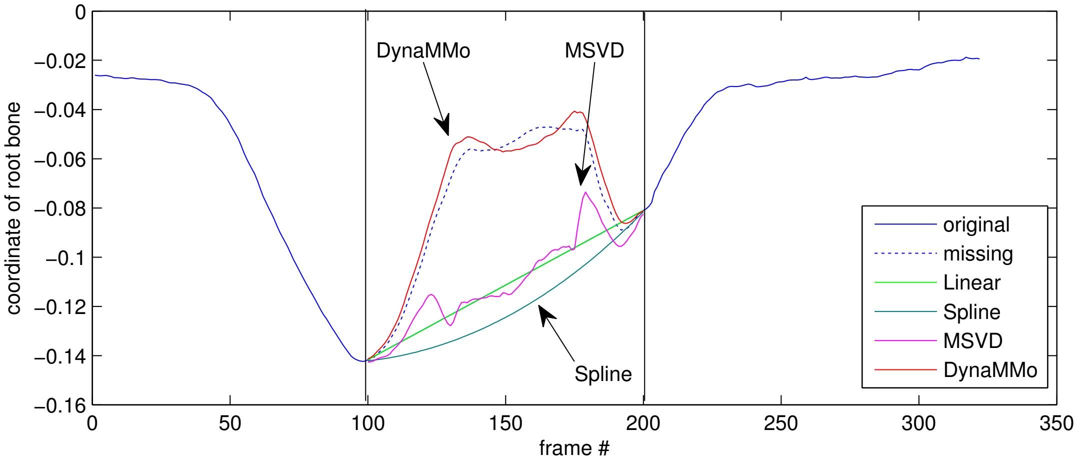


Figure 3.1: Reconstruction for a jump motion with 322 frames in 93 dimensions of bone coordinates. Blue line: the original signal for *root bone* z-coordinate - the dash portion indicates occlusion from frame 100 to 200. The proposed DynaMMo, in red, gets very close to the original, outperforming all competitors.

The focus of our work is to handle occlusions automatically. Straightforward methods like linear interpolation and spline interpolation give poor results (see Section 3.3.4). Ideally we would like a method with the following properties:

1. **Effective:** It should give good results, both with respect to reconstruction error, but primarily agreeing with human intuition.

2. **Scalable:** The computation time of the method should grow slowly with the input and the time-duration T of the motion-capture. Ideally, it should be $O(T)$ or $O(T\log(T))$, but below (T^2).

3. **Black-outs:** It should be able to handle "black-outs", when all markers disappear (e.g., a person running behind a wall, for a moment).

In this chapter, we propose DynaMMo, an automatic method to learn the hidden pattern and handle missing values. Figure 3.1 shows the reconstructed signal for an occluded jumping motion. Our DynaMMo gives the best result close to the original value. Our main idea is to simultaneously exploit smoothness and correlation. Smoothness is what splines and linear interpolation exploit: for a single time-sequence (say, the left-elbow x-value over time), we expect successive entries to have nearby values ($x_n \approx x_{n+1}$). Correlation reflects the fact that sequences are not independent; for a given motion (say, "walking"), the left-elbow and the right-elbow are correlated, lagging each other by half a period. Thus, when we are missing x_n, say, the left elbow at time-tick n, we can reconstruct it by examining the corresponding values of the right elbow (say, y_{n-1}, y_n, y_{n+1}). This two-prong approach can help us handle even "*black-outs*", which we define as time intervals where we lose track of all the time-sequences.

The main contribution of our approach is that it shows how to exploit both sources of redundancy (smoothness and correlation) in a principled way. Specifically, we show how to set up the problem as a Dynamic Bayesian Network and solve it efficiently, yielding results with the best reconstruction error and agreeing with human intuition. Furthermore, we propose several variants based on DynaMMo for additional time series mining tasks such as forecasting, compressing, and segmentation.

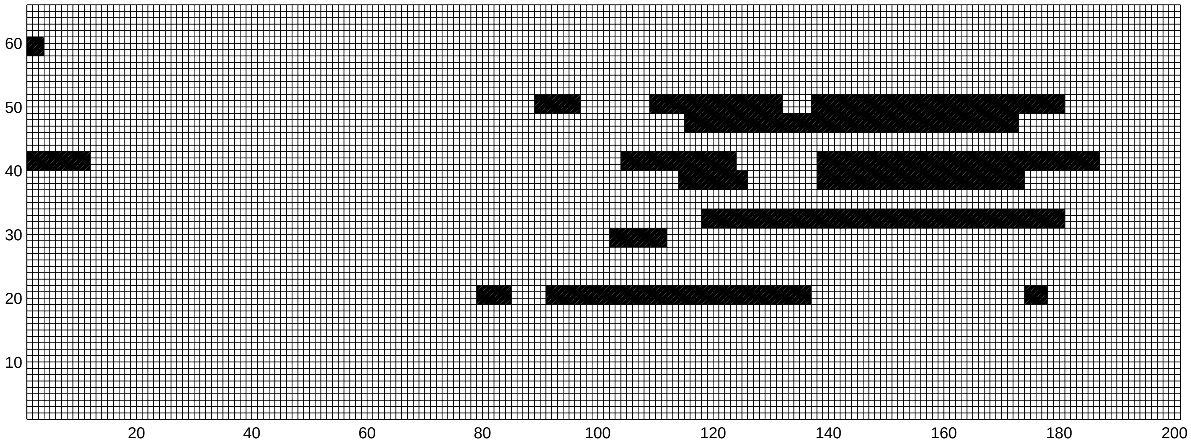


Figure 3.2: Illustration of occlusion in handshake motion. 66 joint angles (rows), for ≈ 200 frames. Dark cells indicate missing values due to occlusion. Notice that occlusions are clustered.

The rest of the chapter is organized as follows: In Section 3.2, we review the related work; the proposed method and its discussion are presented in Section 3.3; the experimental results are presented in Section 3.3.4. Section 3.4 and 3.5 discuss additional application in time series compression and segmentation.

3.2 Related work

Interpolation methods, such as linear interpolation and splines, are commonly used to handle missing values in time series. Both linear interpolation and splines estimate the missing values based on continuity in a single sequence. While these methods are generally effective for short gaps, they ignore the correlations among multiple dimensions.

Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) [Wall et al., 2003] are powerful tools to discover linear correlations across multiple sequences, with which it is possible to recover missing values in one sequence based on observations from others. Srebro and Jaakkola [Srebro and Jaakkola, 2003] have proposed an EM approach (MSVD) to factor the data into low rank matrices and approximate missing value from them. We will describe MSVD in appendix, and we show that it is a special case of our model. Brand [Brand, 2002] further develop an incremental algorithm to fast compute the singular decomposition with missing values. Similar to the missing value SVD approach, Liu and McMillan [Liu and McMillan, 2006] have proposed a method that projects motion capture markers positions into linear principal components and reconstructs the missing parts from the linear models. Furthermore, they proposed an enhanced Local Linear method from a mixture of such linear models. Park and Hodgins [Park and Hodgins, 2006] have also used PCA to estimate the missing markers for skin deformation capturing. In another direction, Yi et al [Yi et al., 2000] have proposed an online regression model over time across multiple dimension that is in extension to Autoregression (AR), thus could handle missing values.

There are several methods specifically for modeling motion capture data. Herda et al [Herda et al., 2000] have used a human body skeleton to track and reconstruct the 3-d marker positions. If a marker is missing, it could predict the position using three previous markers by calculating the kinetics. Hsu et al [Hsu et al., 2004] have proposed a method to map from a motion control specification to a target motion by searching over patterns in existing database. Chai and Hodgins [Chai and Hodgins, 2005] uses a small set of markers as control signals and reconstruct the full body motion from a pre-recorded database. The subset of markers should be known in advance, while our method does not assume fixed subsets observed or missing. As an alternative non-parametric approach, Lawrence and Moore [Lawrence and Moore, 2007] model the human motion using hierarchical Gaussian processes. [Liu and McMillan, 2006] provides a nice summary of related work on occlusion for motion capture data as well as of techniques for related tasks such as motion tracking.

There are many related work in time series representation [Mehta et al., 2006, Lin et al., 2003, Shieh and Keogh, 2008], indexing [Keogh et al., 2004], classification [Gao et al., 2008, Tao et al., 2004] and outlier detection [Lee et al., 2008]. Mehta et al [Mehta et al., 2006] proposed a representation method for time varying data based on motion and shape information including linear velocity and angular velocity. With this representation, they track the tangible features to segment the sequence trajectory. Symbolic aggregate approximation (SAX) [Lin et al., 2003] is a symbolic representation for time series data, and later generalized for massive time series indexing (iSAX) [Shieh and Keogh, 2008]. Keogh et al use uniform scaling when indexing a large human motion database [Keogh et al., 2004]. Lee et al [Lee et al., 2008] proposed the TRAOD algorithm to identify outliers in a trajectory database. In their approach, they first partition the trajectories into small segments and then use both distance and density to detect abnormal sub-trajectories. Gao et al [Gao et al., 2008] proposed an ensemble model to classify the data streams with skewed class distributions and concept drifts. Their approach is to undersample the dominating class, oversample or repeat the rare class and then partition the data set and perform individual training. The trained models are then combined evenly into the resulting classification function. However, none of these methods can handle missing values.

Our method is also related to Kalman Filters and other adaptive filters conventionally used in tracking system. Jain et al [Jain et al., 2004] have adapted Kalman Filters for reducing communication cost in data stream. Tao et al [Tao et al., 2004] have proposed a recursive filter to predict and index moving objects. Li et al [Li et al., 2008] used Kalman filter to stitch motions in a natural way. While our method includes Kalman Filter as a special case, DynaMMo can effectively cope with missing values.

3.3 Prediction with missing values

Given a partially observed multi-dimensional sequence, we propose DynaMMo, to identify hidden variables, to mine their dynamics, and to recover missing values. Our motivation comes from noticing two common properties of time series data: temporal continuity and spatial correlation. On one hand, by exploiting continuity as many interpolation methods do, we expect that missing values are close to observations in neighboring time ticks and follow their moving trends. On the other hand, by using the correlation between difference sequences as SVD does, missing values can be inferred from other observation sources. Our proposed approach makes use of both, to better capture patterns in co-evolving sequences.

3.3.1 The Model

We will first define the problem of time series missing value recovery, and then present our proposed DynaMMo. Table 3.1 explains the symbols and annotations used in this chapter.

Table 3.1: Symbols and Definitions

Symbol	Definition
$\mathcal{X}$	a multi-dimensional sequence of observations with missing values $(\vec{x}_1, ...\vec{x}_\mathrm{T})$
$\mathcal{X}_g$	the observed values in the sequence $\mathcal{X}$
$\mathcal{X}_m$	variables for the missing values in the sequence $\mathcal{X}$
m	dimension of $\mathcal{X}$
T	duration of $\mathcal{X}$
$\mathcal{W}$	missing value indication matrix with the same duration and dimension of $\mathcal{X}$
$\mathcal{Z}$	a sequence of latent variables $(\vec{z}_1, \ldots \vec{z}_\mathrm{T})$
H	dimension of latent variables $(\vec{z}_1 \cdots \vec{z}_\mathrm{T})$

Definition 3.1. *Given a time sequence $\mathcal{X}$ with duration* T *in m dimensions, $\mathcal{X} = \{\vec{x}_1, \ldots, \vec{x}_\mathrm{T}\}$, to recover the missing part of the observations indicated by $\mathcal{W}$. $\vec{w}_{t,k} = 0$ whenever $\mathcal{X}$'s k-th dimensional observation is missing at time t, and otherwise $\vec{w}_{t,k} = 1$.Let us denote the observed part as $\mathcal{X}_g$, and the missing part as $\mathcal{X}_m$.*

We build a probabilistic model (Figure 3.3), to represent the data sequence with missing observations, and the underlying process to generate the data. The imputation problem is to estimate the expectation of missing values conditioned on the observed parts, $\mathbb{E}[\mathcal{X}_m|\mathcal{X}_g]$. We use a sequence of latent variables (hidden states), $\vec{z}_n$, to model the dynamics and hidden patterns of the observation sequence. Like SVD, we assume a linear projection matrix $\mathbf{G}$ from the latent variables to the data sequence (both observed and missing) for each time tick. This mapping automatically captures the correlation between the observation dimensions; thus, if some of the dimensions are missing, they can be inferred from the latent variables. For example, the states could correspond to degrees of freedom, the velocities, and the accelerations in human motion capture data (although we let DynaMMo determine them, automatically); while the observed marker positions could be calculated from these hidden states.To model temporal continuity, we assume the latent variables are time dependent with the values determined from the previous time tick by a linear mapping $\mathbf{F}$. In addition, we assume an initial state for latent variables at the first time tick. Eq (3.1 - 3.3) give the mathematical equations of our proposed model, with the parameters $\theta = \{\mathbf{F}, \mathbf{G}, \vec{z}_0, \Gamma, \Lambda, \Sigma\}$.

$$\vec{z}_1 = \vec{z}_0 + \vec{\omega}_0 \tag{3.1}$$

$$\vec{z}_{n+1} = \mathbf{F}\vec{z}_n + \vec{\omega}_n \tag{3.2}$$

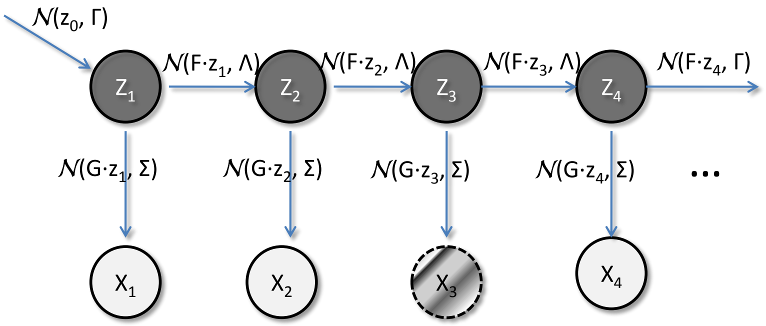
$$\vec{x}_n = \mathbf{G}\vec{z}_n + \vec{\epsilon}_n \tag{3.3}$$

where $\vec{z}_0$ is initial state of the latent variables. $\mathbf{F}$ implies the transition and $\mathbf{G}$ is the observation projection. $\vec{\omega}_0$, $\vec{\omega}_i$ and $\vec{\epsilon}_i (i = 1 \ldots \mathrm{T})$ are multivariate Gaussian noises with the following distributions:

$$\vec{\omega}_0 \sim \mathcal{N}(0, \Gamma) \;\; \vec{\omega}_i \sim \mathcal{N}(0, \Lambda) \;\; \vec{\epsilon}_j \sim \mathcal{N}(0, \Sigma) \tag{3.4}$$

The model is similar to Linear Dynamical System except that it includes an additional matrix W to indicate the missing observations. The joint distribution of $\mathcal{X}_m$, $\mathcal{X}_g$ and $\mathcal{Z}$ is given by

$$P(\mathcal{X}_m, \mathcal{X}_g and \mathcal{Z}) = P(\vec{z}_1) \cdot \prod_{i=2}^{\mathrm{T}} P(\vec{z}_i|\vec{z}_{i-1}) \cdot \prod_{i=1}^{\mathrm{T}} P(\vec{x}_i|\vec{z}_i) \tag{3.5}$$



$\mathcal{N}(z_0, \Gamma)$
$\mathcal{N}(F \cdot z_1, \Lambda)$
$\mathcal{N}(F \cdot z_2, \Lambda)$
$\mathcal{N}(F \cdot z_3, \Lambda)$
$\mathcal{N}(F \cdot z_4, \Gamma)$
Z_1
Z_2
Z_3
Z_4
$\mathcal{N}(G \cdot z_1, \Sigma)$
$\mathcal{N}(G \cdot z_2, \Sigma)$
$\mathcal{N}(G \cdot z_3, \Sigma)$
$\mathcal{N}(G \cdot z_4, \Sigma)$
...
X_1
X_2
X_3
X_4

Figure 3.3: Graphical Illustration of the model used in DynaMMo. $\vec{z}_{1\ldots4}$: latent variables; $\vec{x}_{1,2,4}$: observations; $\vec{x}_3$: partial observations. Arrows denote Gaussian distributions.

3.3.2 The Learning Algorithm

Given an incomplete data sequence $\mathcal{X}$ and the indication sequence $\mathcal{W}$, we propose DynaMMo method to estimate:

1. the governing dynamics **F** and **G**, as well as other parameters z_0, Γ, Λ and Σ;
2. the latent variables $\hat{\vec{z}}_n = \mathbb{E}[\vec{z}_n]$, $(n = 1 \ldots \mathrm{T})$;
3. the missing values of the observation sequence $\mathbb{E}[\mathcal{X}_m | \mathcal{X}_g]$.

The goal of parameter estimation is achieved through maximizing the likelihood of observed data, $\mathcal{L}(\theta) = P(\mathcal{X}_g)$. However, it is difficult to directly maximize the data likelihood in missing value setting, instead, we maximize the expected log-likelihood of the observation sequence. Once we get the model parameters, we use belief propagation to estimate the occluded marker positions. We define the following objective function as the expected log-likelihood $Q(\theta)$ with respect to the parameters $\theta = \{\mathbf{F}, \mathbf{G}, z_0, \Gamma, \Lambda, \Sigma\}$:

$$Q(\theta) = \mathbb{E}_{\mathcal{X}_m, \mathcal{Z} | \mathcal{X}_g, \mathcal{W}}[P(\mathcal{X}_g, \mathcal{Z}_m, \mathcal{Z})] \tag{3.6}$$

$$= \mathbb{E}_{\mathcal{X}_m, \mathcal{Z} | \mathcal{X}_g, \mathcal{W}} \Bigg[-D(\vec{z}_1, z_0, \Gamma) - \sum_{t=2}^{\mathrm{T}} D(\vec{z}_t, \mathbf{F}\vec{z}_{t-1}, \Gamma) - \sum_{t=1}^{\mathrm{T}} D(\vec{x}_t, \mathbf{G}\vec{z}_t, \Sigma) - \frac{\log|\Gamma|}{2} - \frac{(\mathrm{T}-1)\log|\Lambda|}{2} - \frac{\mathrm{T}\log|\Sigma|}{2} \Bigg] \tag{3.7}$$

where $D()$ is the square of the Mahalanobis distance $D(\vec{x}, \vec{y}, \Sigma) = (\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})$

Our proposed DynaMMo searches for the optimal solution using an iterative procedure of Expectation-Maximization-Recovery (EMR), which generalizes the Expectation-Maximization algorithm for LDS [Ghahramani and Hinton, 1996]. The algorithm is an iterative, coordinate descent procedure: estimating the latent variables (Expectation), maximizing with respect to parameters (Maximization), estimating the missing values (Recovery), and iterating until convergence.

E-step The objective function is based on posterior distribution of the hidden variables ($\vec{z}$'s) given the partial observed data sequence, e.g. $\hat{\vec{z}}_n = \mathbb{E}[\vec{z}_n \mid \mathcal{X}_g, \mathcal{X}_m](n = 1, \ldots, \mathrm{T})$. In the very first step, we will initialize the missing values to be some random number (or we can fill them using linear interpolation). In E-step, we use a belief propagation algorithm to estimate the posterior expectations of latent variables, similar to message passing in Hidden Markov Model and Linear Dynamical Systems. The general idea is to compute the posterior distribution of latent variables tick by tick, based on the computation of previous time tick.

Since both prior and conditional distributions in the model are Gaussian, the posterior up to current time tick $p(\vec{z}_n|\vec{x}_1, \ldots, \vec{x}_T)$ should also be Gaussian, denoted by $\hat{\alpha}(\vec{z}_n) = \mathcal{N}(\mu_n, \mathbf{V}_n)$. Let $p(\vec{x}_n|\vec{x}_1, \ldots, \vec{x}_{n-1})$ denoted as c_n, We have the following propagation equation:

$$c_n \hat{\alpha}(\vec{z}_n) = p(\vec{x}_n|\vec{z}_n) \int \hat{\alpha}(\vec{z}_{n-1}) p(\vec{z}_n|\vec{z}_{n-1}) d\vec{z}_{n-1} \tag{3.8}$$

From Eq 3.8 we could obtain the following forward passing of the belief. The messages here are $\vec{\mu}_n$, $\mathbf{V}_n$ and $\mathbf{P}_{n-1}$(needed in later backward passing).

$$\begin{aligned}
\mathbf{P}_{n-1} &= \mathbf{F}\mathbf{V}_{n-1}\mathbf{F}^T + \Lambda & (3.9)\\
\mathbf{K}_n &= \mathbf{P}_{n-1}\mathbf{G}^T(\mathbf{G}\mathbf{P}_{n-1}\mathbf{G}^T + \Sigma)^{-1} & (3.10)\\
\vec{\mu}_n &= \mathbf{F}\vec{\mu}_{n-1} + \mathbf{K}_n(\vec{x}_n - \mathbf{G}\mathbf{F}\vec{\mu}_{n-1}) & (3.11)\\
\mathbf{V}_n &= (\mathbf{I} - \mathbf{K}_n)\mathbf{P}_{n-1} & (3.12)\\
c_n &= \mathcal{N}(\mathbf{G}\mathbf{F}\vec{\mu}_{n-1}, \mathbf{G}\mathbf{P}_{n-1}\mathbf{G}^T + \Sigma) & (3.13)
\end{aligned}$$

The initial messages are given by:

$$\begin{aligned}
\mathbf{K}_1 &= \mathbf{\Gamma}\mathbf{G}^T(\mathbf{G}\mathbf{\Gamma}\mathbf{G}^T + \Sigma)^{-1} & (3.14)\\
\vec{\mu}_1 &= \vec{\mu}_0 + \mathbf{K}_1(\vec{x}_1 - \mathbf{G}\mathbf{F}\vec{\mu}_0) & (3.15)\\
\mathbf{V}_1 &= (\mathbf{I} - \mathbf{K}_1)\mathbf{\Gamma} & (3.16)\\
c_1 &= \mathcal{N}(\mathbf{G}\vec{\mu}_0, \mathbf{G}\mathbf{\Gamma}\mathbf{G}^T + \Sigma) & (3.17)
\end{aligned}$$

For the backward passing, let $\gamma(\vec{z}_n)$ denote the marginal posterior probability $p(\vec{z}_n|\vec{x}_1, \ldots, \vec{x}_N)$ with the assumption (we can prove it's Gaussian):

$$\gamma(\vec{z}_n) = \mathcal{N}(\hat{\mu}_n, \hat{\mathbf{V}}_n) \tag{3.18}$$

The backward passing equations are:

$$\begin{aligned}
\mathbf{J}_n &= \mathbf{V}_n\mathbf{F}^T(\mathbf{P}_n)^{-1} & (3.19)\\
\hat{\vec{\mu}}_n &= \vec{\mu}_n + \mathbf{J}_n(\hat{\vec{\mu}}_{n+1} - \mathbf{F}\vec{\mu}_n) & (3.20)\\
\hat{\mathbf{V}}_n &= \mathbf{V}_n + \mathbf{J}_n(\hat{\mathbf{V}}_{n+1} - \mathbf{P}_n)\mathbf{J}_n^T & (3.21)
\end{aligned}$$

Hence, the expectation for Algorithm 3.1 line 5 are computed using the following equations:

$$\begin{aligned}
\mathbb{E}[\vec{z}_n] &= \hat{\vec{\mu}}_n & (3.22)\\
\mathbb{E}[\vec{z}_n\vec{z}_{n-1}^T] &= \mathbf{J}_{n-1}\hat{\mathbf{V}}_n + \hat{\vec{\mu}}_n\hat{\vec{\mu}}_{n-1}^T & (3.23)\\
\mathbb{E}[\vec{z}_n\vec{z}_n^T] &= \hat{\mathbf{V}}_n + \hat{\vec{\mu}}_n\hat{\vec{\mu}}_n^T & (3.24)
\end{aligned}$$

where the expectations are taken over the posterior marginal distribution $p(\vec{z}_n|\vec{y}_1, \ldots, \vec{y}_N)$.

These expectation in together consist sufficient statistics, which provides sufficient information for updating the model parameters θ.

M-step To estimate the parameters, taking the derivatives of Eq (3.6-3.7) with respect to the components of θ^{new} and setting them to zero yield the following results:

$$\vec{\mu}_0^{new} = \mathbb{E}[\vec{z}_1] \tag{3.25}$$

$$\mathbf{\Gamma}^{new} = \mathbb{E}[\vec{z}_1 \vec{z}_1^T] - \mathbb{E}[\vec{z}_1]\mathbb{E}[\vec{z}_1^T] \tag{3.26}$$

$$\mathbf{F}^{new} = \left(\sum_{n=2}^{N} \mathbb{E}[\vec{z}_n \vec{z}_{n-1}^T]\right) \left(\sum_{n=1}^{N-1} \mathbb{E}[\vec{z}_n \vec{z}_n^T]\right)^{-1} \tag{3.27}$$

$$\mathbf{\Lambda}^{new} = \frac{1}{N-1} \sum_{n=2}^{N} \left(\mathbb{E}[\vec{z}_n \vec{z}_n^T] - \mathbf{F}^{new}\mathbb{E}[\vec{z}_{n-1} \vec{z}_n^T] - \mathbb{E}[\vec{z}_n \vec{z}_{n-1}^T](\mathbf{F}^{new})^T + \mathbf{F}^{new}\mathbb{E}[\vec{z}_n \vec{z}_{n-1}^T](\mathbf{F}^{new})^T\right) \tag{3.28}$$

$$\mathbf{G}^{new} = \left(\sum_{n=1}^{N} \vec{x}_n \mathbb{E}[\vec{z}_n^T]\right) \left(\sum_{n=1}^{N} \mathbb{E}[\vec{z}_n \vec{z}_n^T]\right)^{-1} \tag{3.29}$$

$$\mathbf{\Sigma}^{new} = \frac{1}{N} \sum_{n=1}^{N} \left(\vec{y}_n \vec{y}_n^T - \mathbf{G}^{new}\mathbb{E}[\vec{z}_n]\vec{y}_n^T - \vec{y}_n \mathbb{E}[\vec{z}_n^T](\mathbf{G}^{new})^T + \mathbf{G}^{new}\mathbb{E}[\vec{z}_n \vec{z}_n^T](\mathbf{G}^{new})^T\right) \tag{3.30}$$

The calculation of optimal parameters in Eq (3.25-3.30) requires estimation of latent variables, which is computed in E-step Eq (3.23-3.24).

R-step Finally, the missing values are easily computed from the estimation of latent variables using Markov property in the graphical model (Figure 3.3). We have the following equation:

$$\mathbb{E}[\mathcal{X}_m | \mathcal{X}_g, \mathcal{Z}; \theta] = \mathbf{G} \cdot \mathbb{E}[\mathcal{Z}]_{\{i,j\}}(\{i,j\} \in \mathcal{W}) \tag{3.31}$$

The overall algorithm is outlined in Algorithm 3.1. Note that our algorithm is general to handle sequences with or without missing values. In the case of full observation, our algorithm will be the same as traditional EM algorithm [Ghahramani and Hinton, 1996].

3.3.3 Discussion

Model Generality: Our model includes MSVD, linear interpolation, and Kalman filters as special cases:

- MSVD: If we set **F** and z_0 to 0, and $\Gamma = \Lambda$, then the model becomes MSVD, We describe MSVD in Section 3.3.4
- Linear interpolation: For one dimensional data, we obtain the linear interpolation by setting $\Lambda = 0$ and the rest of the parameters to the following values:

$$\mathbf{F} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{G} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

- Equations (3.1-3.3) of DynaMMo become the equations of the traditional Kalman filters if there are no missing values. In that case, the well-known, single-pass Kalman method applies.

Algorithm 3.1: DynaMMo

```
Input: Observed data sequence: $X = X_g$,
Missing value indication matrix:$W$
the number of latent dimension $H$
Output:
    • Estimated sequence:$\hat{X}$
    • Latent variables $\hat{\vec{z}}_1 \cdots \hat{\vec{z}}_{\mathrm{T}|}$
    • Model parameters $\theta$
Initialize $\hat{X}$ with $X_g$ and the missing value filled by linear interpolation or other methods;
Initialize $\mathbf{F}, \mathbf{G}, z_0$;
Initialize $\Gamma, \Lambda, \Sigma$ to be identity matrix;
$\theta \leftarrow \{\mathbf{F}, \mathbf{G}, z_0, \Gamma, \Lambda, \Sigma\}$;
repeat
    Estimate $\hat{\vec{z}}_{1\cdots\mathrm{T}} = \mathbb{E}[\mathcal{Z}|\hat{X};\theta]$ using belief propagation;
    Maximizing Eq (3.7) with $\mathbb{E}[\mathcal{Z}|\hat{X};\theta]$ using Eq. (3.25-3.30),
        $\theta^{new} \leftarrow \arg\max_\theta Q(\theta)$;
    forall i,j do
        // update the missing values
        if $W_{i,j} = 0$ then $X_{i,j}$ is missing
            $\hat{X}^{new}_{i,j} \leftarrow (\mathbf{G}^{new} \cdot \mathbb{E}[\mathcal{Z}|\hat{X};\theta])_{i,j}$
        end
    end
until converge ;
```

Penalty and Constraints: In the algorithm described above, the first part in Eq. (3.7) is error term for the initial state. The second term in Eq. (3.7) is trying to estimate the dynamics for the hidden states, while the third term in Eq. (3.7) is getting the best projection from observed motion sequence to hidden states. Eq. (3.7) is penalty for the covariance, similar to model complexity in BIC. It is easy to extend the model by putting a further penalty on the model complexity through a Bayesian approach. For example, we could constraint the covariance to be diagonal $\sigma^2\mathbf{I}$, which is used in our experiments, since it is faster to compute.

Time Complexity: The algorithm needs time linear on the duration T, and specifically $O(\#(iterations) \cdot T \cdot m^3)$. Thus, we expect it to scale well for longer sequences. As a point of reference, it takes about 6 to 10 minutes per sequence with several hundreds time ticks, on a Pentium class desktop.

Robustness: Our model uses linear dynamical system as the underlying process to generate the data, as a result we assume Gaussian noises associated with the generation process. Thus our model is robust under Gaussian noises. However, in reality we may sometimes encounter non-Gaussian noises, such as noises in power laws. In those cases, we might need to perform an additional pre-processing step, for instance, taking logarithms will often do the job for spiky data (e.g. BGP data).

Algorithm 3.2: Missing Value SVD (MSVD)

```
Input: Observed data matrix: $X_g$,
Occlusion indication matrix:$W$
the number of hidden dimensions $H$
Output: Estimated data matrix:$\hat{X}$
Initialize $\hat{X}$ with $X_g$ and the missing value filled by linear interpolation;
repeat
    taking SVD of $\hat{X}$, $\hat{X} \approx U_H \Lambda_H V_H^T$;
    // update estimation
    $Y \leftarrow U_H \Lambda_H V_H^T$;
    forall i,j do
        if $W_{i,j} = 0$ then $X_{i,j}$ is missing
            $\hat{X}_{i,j}^{new} \leftarrow Y_{i,j}$
        end
    end
until converge ;
```

3.3.4 Experiments

We evaluate both quality and scalability of DynaMMo on several datasets. To evaluate the quality of recovering missing values, we use a real dataset with part of data treated as "missing" so that it enables comparing the real observations with the reconstructed ones. In the following we first describe the dataset and baseline methods, and then present the reconstruction results.

Experiment Setup

Baseline Methods We use linear interpolation and Missing Value SVD (MSVD) as the baseline methods. We also compare to spline interpolation.

MSVD involves iteratively taking the SVD and fitting the missing values from the result [Srebro and Jaakkola, 2003]. This method is very easy to implement and already used on motion capture datasets in [Liu and McMillan, 2006] and [Park and Hodgins, 2006]. In our implementation (Alg. 3.2), we initialized the holes by linear interpolation, and use 15 principal dimensions (99% of energy).

Datasets **Chlorine Dataset (Chlorine):** The Chlorine dataset (see sample in Figure 3.6(a)) was produced by EPANET 2[1] that models the hydraulic and water quality behavior of water distribution piping systems. EPANET can track, in a given water network, the water level and pressure in each tank, the water flow in the pipes and the concentration of a chemical species (Chlorine in this case) throughout the network within a simulated duration. The data set consists of 166 nodes (pipe junctions) and measurement of the Chlorine concentration level at all these nodes during 15 days (one measurement for every 5 minutes, a total of 4310 time ticks). Since the water demand pattern during the 15 days follows a clear global

[1] http://www.epa.gov/nrmrl/wswrd/dw/epanet.html

periodic pattern (daily cycle, dominating residential demand pattern), EPANET would correctly reflect the pattern in the Chlorine concentration with a few exceptions and slight time shifts.

Full body motion set (Motion): This data set contains 58 full body motions of walking, running, and jumping motions from subject #16 of mocap database[2]. Each motion spans several hundred of frames with 93 features of bone positions in body local coordinates. The total size of the dataset is 17MB. We make random dropouts and reconstruct the missing values on the data set.

Simulate Missing Values We create synthetic occlusions (dropouts) on the Motion data and evaluate the effectiveness of reconstruction by DynaMMo. To mimic a real occlusion, we collected the occlusion statistics from handshake motions. For example, there are 10.44% of occluded values in typical handshake motions, and occlusions often occur consecutively (Figure 10.4). To create synthetic occlusions, we randomly pick a marker j and the starting point (frame) n for the occlusion of this marker; we pick the duration as a Poisson distributed random variable according to the observed statistics, and we repeat, until we have occluded 10.44% of the input values.

Experimental Results

We present three sets of results, to illustrate the quality of reconstruction of DynaMMo.

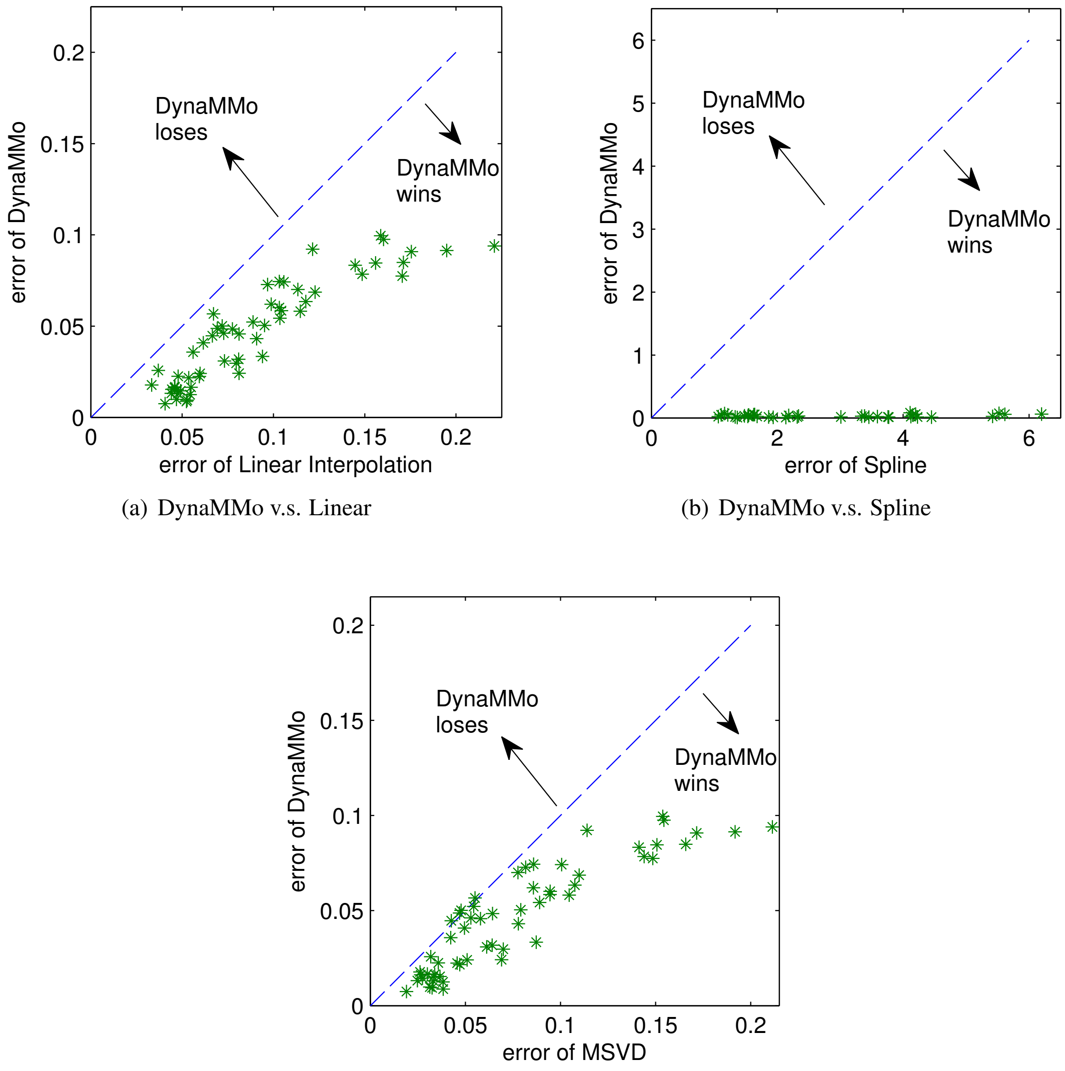
Qualitative result Figure 3.1 shows the reconstructed signal (root bone z-coordinate) for a jump motion. Splines find a rather smooth curve which is not what the human actor really did. Linear interpolation and MSVD are a bit better while still far from the ground truth. Our proposed DynaMMo (with 15 hidden dimensions) captured both the dynamics of the motion as well as the correlations across the given inputs, and achieved a very good reconstruction of the signal.

Reconstruction Error For each motion in the data set, we create a synthetic occluded motion sequence as described above, reconstruct using DynaMMo, then compare the effectiveness against linear interpolation, splines and MSVD. To reduce random effects, we repeat each experiment 10 times and we report the average of MSE. To evaluate the quality, we use the MSE: Given the original motion X, the occlusion indication matrix W and the fitted motion $\hat{X}$, the RMSE is the average of squared differences between the actual (X) and reconstructed ($\hat{X}$) missing values - formally:

$$RMSE(X, W, \hat{X}) = \frac{||(1-W) \circ (X - \hat{X})||^2}{||1-W||^2} = \frac{1}{\sum_{t,k}(1-W_{t,k})} \sum_{t,k} (1-W_{t,k})(X_{t,k} - \hat{X}_{t,k})^2 \tag{3.32}$$

Both MSVD and our method use 15 hidden dimensions ($H = 15$). Figure 3.4 shows the scatter plots of the average reconstruction error over 58 motions in the Motion dataset, with 10% missing values and 50 average occlusion length. It is worth to noting that the reconstruction grows little with increasing occlusion length, compared with other alternative methods (Figure 3.5). There is a similar result found in experiments on Chlorine data as shown in Figure 3.6(b). Again, our proposed DynaMMo achieves the best performance among the four methods.

[2] http://mocap.cs.cmu.edu



(a) DynaMMo v.s. Linear

(b) DynaMMo v.s. Spline

(c) DynaMMo v.s. MSVD

Figure 3.4: Comparison of missing value reconstruction methods for Mocap dataset: the scatter plot of reconstruction error.

Scalability As we discussed in Section 3.3, the complexity of DynaMMo is $O(\#(iterations) \cdot T \cdot m^3)$. Figure 3.7 shows the running time of the algorithm on the Chlorine dataset versus the sequence length. For each run, 10% of the Chlorine concentration levels are treated as missing with average missing length 40. As expected, the wall clock time is almost linear to sequence duration.

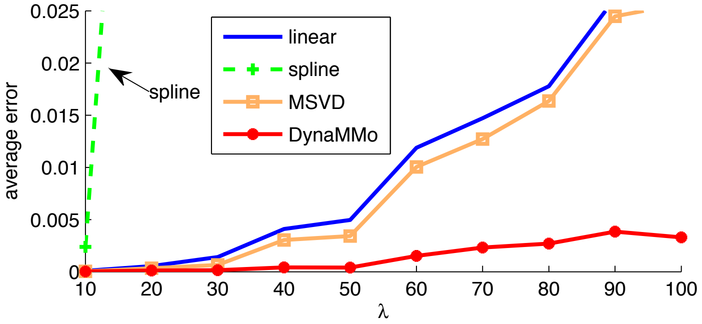
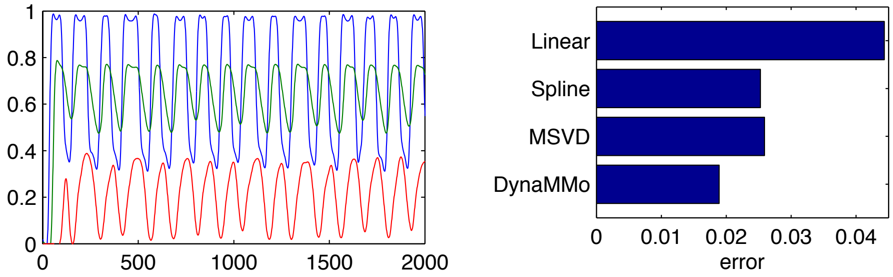


Figure 3.5: Average error for missing value recovery on a sample mocap data (subject#16.22). Average rmse over 10 runs, versus average missing length λ(from 10 to 100). Randomly 10.44% of the values are treated as "missing". DynaMMo (in red solid line) wins. Splines are off the scale.



(a) Sample Chlorine data.

(b) Reconstruction error.

Figure 3.6: Reconstruction experiment on Chlorine with 10% missing and average occlusion length 40.

3.4 Compression

Time series data are usually real valued which makes it hard to achieve a high compression ratio using lossless methods. However, lossy compression is reasonable if it gets a high compression ratio and low recovery error. As described in Section 3.3, DynaMMo produces three outputs: model parameters, latent variables (posterior expectation) and missing values. To compress, we record some of the hidden variables learned from DynaMMo instead of storing direct observations. By controlling the hidden dimension and the number of time ticks of hidden variables to keep, it is easy to trade off between compression ratio and error. We provide three alternatives for compression.

Here we first present the decompression algorithm in Algorithm 3.3.

3.4.1 Fixed Compression: DynaMMo $_f$

The fixed compression will first learn the hidden variables using DynaMMo and store the hidden variables for every k time ticks. In addition, it stores the matrix $\mathbf{F}$, $\mathbf{\Lambda}$, $\mathbf{G}$ and $\mathbf{\Sigma}$. Both covariance $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$ are constrained to $\lambda^2 I$ and $\sigma^2 I$ respectively. It also stores the number k.

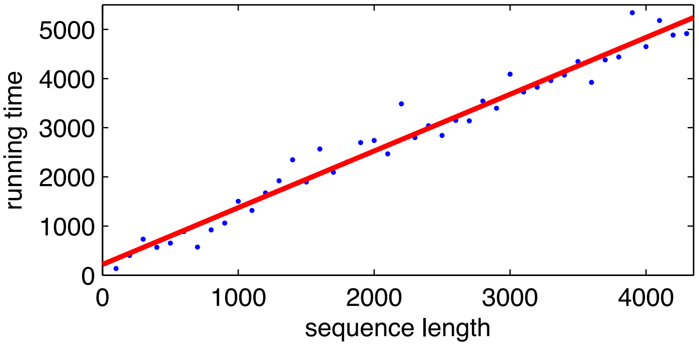


Figure 3.7: Running time versus the sequence length on Chlorine dataset. For each run, 10% of the values are treated as “missing”.

Algorithm 3.3: DynaMMo _Decompress

```
Input: $\hat{\vec{z}}_S$, hidden variables, indexed by $S \subseteq [1 \cdots \mathrm{T}]$, F, G.
Output: The decompressed data sequence $\hat{\vec{x}}_{1 \cdots \mathrm{T}}$
$\vec{y} \leftarrow \hat{\vec{z}}_1$;
for $n \leftarrow 1$ to $\mathrm{T}$ do
    if $n$ in $S$ then
        $\vec{y} \leftarrow \hat{\vec{z}}_n$;
    else
        $\vec{y} \leftarrow \mathbf{F} \cdot \vec{y}$;
    end
    $\hat{\vec{x}}_n \leftarrow \mathbf{G} \cdot \vec{y}$;
end
```

The total space required for fixed compression is $S_f = \frac{\mathrm{T}}{k} \cdot H + H^2 + H \cdot m + 3$, where k is the gap number given.

3.4.2 Adaptive Compression: DynaMMo $_a$

The adaptive compression will first learn the hidden variables using DynaMMo and store the hidden variable only for the necessary time ticks, when the error is greater than a given threshold. Like fixed compression, it also stores the matrix $\mathbf{F}$, $\mathbf{\Lambda}$, $\mathbf{G}$ and $\mathbf{\Sigma}$. Both covariance $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$ are constrained to $\lambda^2 I$ and $\sigma^2 I$ respectively. For each stored time tick, it also records the offset of next storing time tick.

The total space required for adaptive compression is $S_a = l \cdot (H + 1) + H^2 + H \cdot m + 2$ where l is the number of stored time ticks.

3.4.3 Optimal Compression: DynaMMo $_d$

The optimal compression will first learn the hidden variables using DynaMMo and store the hidden variables for the time ticks determined by dynamic programming, so as to achieve the smallest error for a given number of stored time ticks. Like the fixed compression, it also stores the matrix $\mathbf{F}$, $\mathbf{\Lambda}$, $\mathbf{G}$ and $\mathbf{\Sigma}$. Both covariance $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$ are constrained to $\lambda^2 I$ and $\sigma^2 I$ respectively.

The total space required for optimal compression is $S_d = l \cdot (H + 1) + H^2 + H \cdot m + 2$ where k is the number of stored time ticks.

Baseline Method

We use a combined method of SVD and linear interpolation as our baseline. It works as follows: given k and h, it first projects the data into h principle dimensions using SVD, then records the hidden variables for every k time ticks. In addition, it will also record the projection matrix from SVD. When decompressing, the hidden variables are projected back using the stored matrix and the gaps filled with linear interpolation.

The total space required for baseline compression is $S_b = \frac{\mathrm{T}}{k} \cdot h + h \cdot m + h + 1$, where k is the gap number given, and h is the number of principle dimensions.

For all of these methods, the compression ratio is defined as

$$R_* = \frac{\text{Total uncompressed space}}{\text{compressed space}} = \frac{\mathrm{T} \cdot m}{S_*}$$

Figure 3.8 shows the decompression error (in terms of RMSE) with respect to compression ratio compared with the baseline compression using a combined method SVD and Linear Interpolation. DynaMMo $_d$ wins especially in high compression ratio.

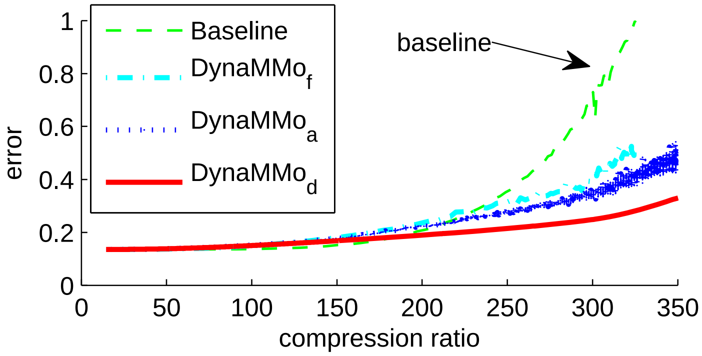


Figure 3.8: Compression for CHLORINE dataset: RMSE versus compression ratio. Lower is better. DynaMMo $_d$ (in red solid) is the best.

3.5 Segmentation

As a further merit, our DynaMMo is able to segment the data sequence. Intuitively, this is possible because DynaMMo identifies the dynamics and patterns in data sequences, so segments with different patterns can be expected to have different model parameters and latent variables. We use the reconstruction error as an instrument of segmentation. Note that since there might be missing value in the data sequences, a normalization procedure with respect to the number observation at each time tick is required. We present our segmentation method in Algorithm 3.4.

Algorithm 3.4: DynaMMo _Segment

```
Input: Data sequence: $X$,
With or without missing value indication matrix:$W$
the number of latent dimension $H$
Output: The segmentation position $s$
$\{\mathbf{G}, \hat{\vec{z}}_{1\cdots m}\} \leftarrow$ DynaMMo$(X, W, H)$;
for $n = 1$ to $m$ do
    Reconstruct the data for time tick $n$: $\hat{\vec{x}}_n \leftarrow \mathbf{G} \cdot \hat{\vec{z}}_n$;
    Computer the reconstruction error for time tick $n$:
        $$\triangle_n \leftarrow \frac{||\vec{w}_n^T \circ (\hat{\vec{x}}_n - \vec{x}_n)||^2}{\sum \vec{w}_n}$$
end
find the split: $s \leftarrow \arg\max_k \triangle_k$;
```

To illustrate, Figure 3.9 shows the segmentation result on a sequence composed of two pieces of sinusoidal signals with different frequencies. Our segmentation method could correctly identify the time of frequency change by tracking the spikes in reconstruction error. Figure 3.10 shows the reconstruction error from segmentation experiment on a real human motion sequence in which an actor running to a complete stop. Two (y-coordinates of left hip and femur) of 93 joint coordinates are shown in the top of the plot. Note the spikes in the error plot coincide with the slowdown of the pace and transition to stop.

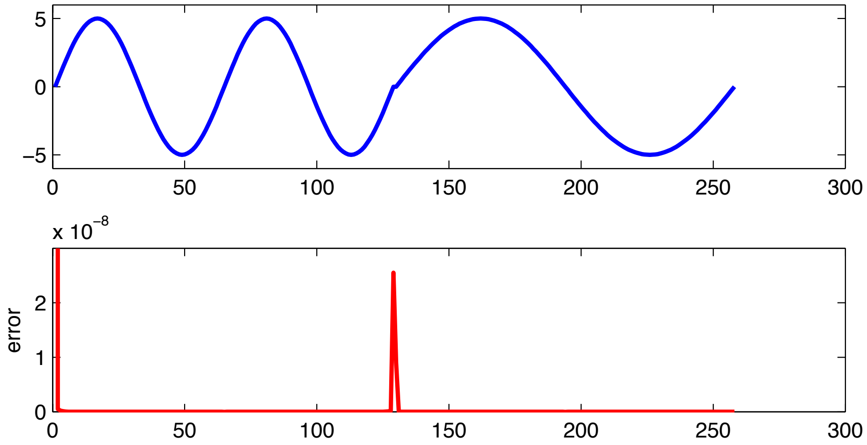


Figure 3.9: Segmentation result on one dimensional synthetic data. Top is a sequence composed of two pieces of sinusoidal signals with different frequencies 64 and 128 respectively. Bottom is the reconstruction error per time tick. Note the spike in the middle correctly identify the shifting of frequencies.

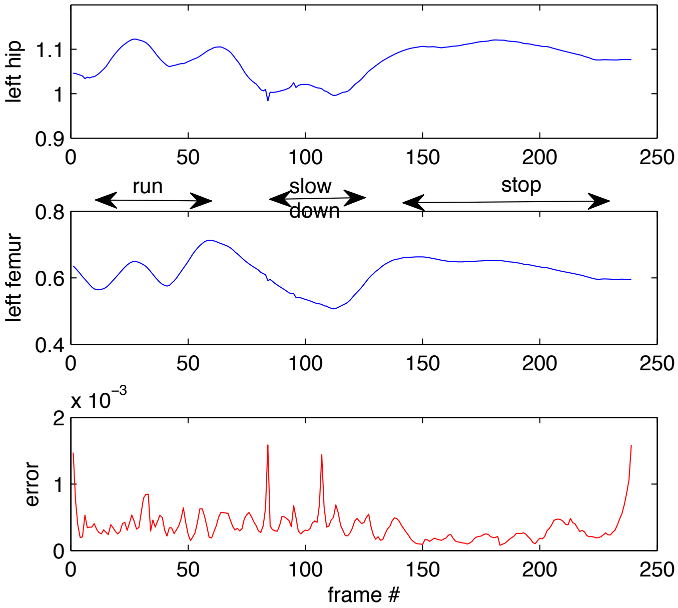


Figure 3.10: Reconstruction error plot for segmentation on a real motion capture sequence in 93 dimensions (subject#16.8) with 250 frames, an actor running to a complete stop, with left hip and femur y-coordinates shown in top plots. The spikes in bottom plot coincide with the slowdown of the pace and transition to stop.

3.6 Summary

In this chapter, we present DynaMMo (Dynamics Mining with Missing values), which includes a learning algorithm and its variant extensions to summarize, compress and find latent variables. The idea is to automatically discover a few, hidden variables and to compactly describe how the hidden variables evolve by learning their transition matrix F. Our algorithm can even work when there are missing observations, and includes Kalman filters as special case.

We presented experiments on motion capture sequences and chlorine measurements and demonstrated that our proposed DynaMMo method and its extensions (a) can successfully learn the latent variables and their evolution, (b) can provide high compression for little loss of reconstruction accuracy, and (c) can extract compact, but powerful features, for sequence forecasting, interpretation and segmentation, (d) scalable on duration of time series.

Chapter 4

Feature Extraction and Clustering

How to tell the similarities among time sequences? In this chapter and continuing at next chapter, we study the problem of summarizing multiple time series effectively and efficiently. We propose PLiF, a novel method to discover essential characteristics ("fingerprints"), by exploiting the joint dynamics in numerical sequences. Our fingerprinting method has the following benefits: (a) it leads to *interpretable* features; (b) it is *versatile*: PLiF enables numerous mining tasks, including clustering, compression, visualization, forecasting, and segmentation, matching top competitors in each task; and (c) it is fast and *scalable*, with linear complexity on the length of the sequences.

We will demonstrate the effectiveness of PLiF on both synthetic and real datasets, including human motion capture data (17MB of human motions), sensor data (166 sensors), and network router traffic data (18 million raw updates over 2 years). Despite its generality, PLiF outperforms the top clustering methods on clustering; the top compression methods on compression (3 times better reconstruction error, for the same compression ratio); it gives meaningful visualization and at the same time, enjoys a *linear* scale-up.

4.1 Introduction

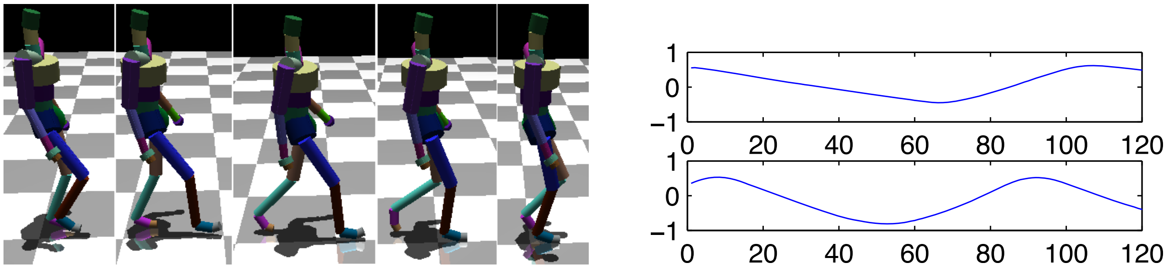
Time sequences appear in countless applications, like sensor measurements [Jain et al., 2004], mobile object tracking [Kollios et al., 1999], data center monitoring [Reeves et al., 2009], motion capture sequences [Keogh et al., 2004, Li et al., 2009], environmental monitoring (like chlorine levels in drinking water [Papadimitriou et al., 2005])and many more. Given multiple, interacting time sequences, how can we group them according to similarity? How can we find compact, numerical features ("*fingerprints*"), to describe and distinguish each of them?

Researches in time sequences form two broad classes:

(a) Feature extraction (and similarity search, indexing etc), using, say, Fourier or wavelet coefficients, piece-wise linear approximations and similar methods [Keogh et al., 2001, 2004] and
(b) forecasting, like an autoregressive integrated moving average model (ARIMA) and related methods [Box et al., 1994].

The former class is useful for querying: indexing, similarity searching, clustering. The latter class is useful for mining: forecasting, missing value imputation, anomaly detection. Can we develop a method that has the best of both worlds? Extracting the essence of time sequences is already very useful - it would be even more useful if those features are easy to interpret, and even better if they could help us do forecasting. Ability to forecast automatically leads to anomaly detection (every time-tick that deviates too much from our forecast), segmentation (a time interval deviating too much from our forecast), compression (storing the deltas from the forecasts), and missing value imputation, extrapolation and interpolation. And of course, we would like the method to be scalable, with linear complexity on the length of the sequences. Is it possible to achieve as many as possible of the above goals, any of which alone is already very useful? The proposed PLiF method gives a positive answer: the idea is to extract the essential numerical representation that characterizes the evolving dynamics of the sequences, specifically, to fit a *Linear Dynamical System* (LDS) on the collection of m sequences, and then we show how to extract a few, but meaningful features out of the LDS. We will refer to those features as the "*fingerprints*" of each sequence. We further show that the proposed *fingerprints* achieve all the above goals:

1. *Effectiveness:* the resulting features lead to a natural distance function, which agrees with human intuition and the provided ground truth. Thus, fingerprints lead to good clustering as well as visualization (see Fig. 4.2(b) and Fig. 4.9);
2. *Interpretability*: as we will show, the fingerprints correspond to groups of harmonics;
3. *Forecasting:* they naturally lead to forecasting and compression;
4. *Scalability:* the proposed PLiF method is fast and scalable on the size of the sequences.



(a)

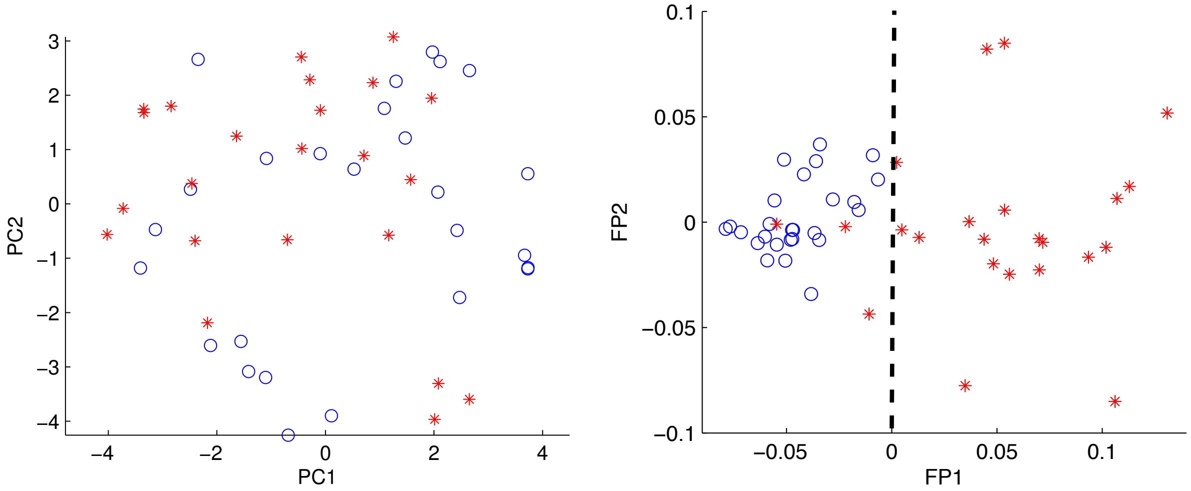
(b) z-value of right foot marker

Figure 4.1: Sample frames and sequences of human motion capture data. 4.1(a): a film-strip of a human walking motion. 4.1(b): sequences of right foot z-coordinates for a walking motion (top), and a running one (bottom).

Table 4.1 compares PLiF against traditional methods and illustrates their strengths and shortcomings: a check-mark (✓) indicates that the corresponding method (column) fulfils the corresponding requirement (row). Only PLiF has all entries as check-marks. In more detail, the desirable fingerprints should allow for lags, and small variations in frequency. While,

- Fourier analysis and wavelet methods could identify the frequencies in a *single* signal, but can not cross-correlate similar signals, nor do forecasting[1].
- Singular value decomposition (SVD) and its "centered" version, principal component analysis (PCA), do capture correlations (by doing soft clustering of similar sequences) and thus derive hidden ("latent") variables, but they can not do forecasting either, nor is it easy to interpret the derived hidden variables.

[1] One might argue that Fourier coefficients can do rudimentary forecasting, since they can generate values for time ticks outside the initial time range $(1, \ldots, T)$; however, these values will just lead to repeating the initial signal, with ringing phenomena if the signal has a trend.



(a) PCA

(b) fingerprints: FP1 vs FP2

Figure 4.2: Motion capture sequences: extracted features and visualization for the data in Figure 4.1. 4.2(a): Scatter-plot of two principal components for walking (blue circles) and running sequences (red stars), without clear separation. 4.2(b): Scatter-plot of the "fingerprints" (FP) by PLiF - first FP versus second FP, for all 49 mocap sequences. Notice the near-perfect separation of the walking motions (blue circles), from the running ones (red stars).

- Standard Linear Dynamical Systems (LDS) and Kalman filters can capture correlations, as well as do forecasting. However, the resulting hidden variables are hard to interpret (see Fig. 4.4(b)), and they do not lead to a natural distance function.

Finally, we do not show typical distance functions for time series clustering in Table 4.1: the Euclidean distance (sum of squares of differences at each time-tick) and the Dynamic Time Warping (DTW) distance. The reason is that none of them leads to forecasting, nor to feature extraction, and thus the interpretability requirement is out of reach. Moreover, the typical, un-constrained, version of DTW fails the scalability requirement, being quadratic on the length of the sequences.

To make the discussion more concrete, we refer to one of our motivating applications, motion capture (*mocap*) sequences: For each motion, we have 93 numbers (positions or angles of markers on the joints of the actor), with a variable number of frames (=time-ticks) for each such sequence, typically 60-120 frames per second. See Fig. 4.1(b) for two such example sequences, both plotting the right-foot z-value as a function of time, for one of the walking motions, and one of the running ones, from the publicly available `mocap.cs.cmu.edu` repository of mocap sequences.

Given a large collection of such human motion sequences, we want to find clusters and to group similar motions together. The desirable features/fingerprints would have the following properties:

- P1: Lag independence: two walking motions should be grouped together even if they start at different footstep or phase;
- P2: Frequency proximity: running motions with nearby speed of motion should be grouped together;
- P3: Harmonics grouping: Several sensor measurements, like human motion, human voice, automobile traffic, obey several periodicities, often with related frequencies ("harmonics"). We would like

to detect such groups of harmonics.

Fig. 4.2(b) gives a quick preview of the visualization and effectiveness of the proposed PLiF method: For the 49 sequences we have, we map each to its two fingerprint values, thus making it a 2-d point. Those 49 points are shown in Fig. 4.2(b), using 'stars' for motions that were (manually) labeled as running motions, and circles for the walking motions. Notice how clearly the two groups can be separated by a vertical line at x=0.

Table 4.1: Capabilities of approaches. Only PLiF meets all specs[2].

	SVD/ PCA	DFT/ DWT	LDS	PLiF
Correlation Discovery	✓		✓	✓
Interpretability		✓		✓
Forecasting			✓	✓

The rest of the chapter is organized in the typical way: In the upcoming sections, we give the problem definition and a running example, then we list the shortcomings of earlier methods, the description of PLiF, experiments, related work and conclusions.

4.2 An illustration of fingerprinting time series

For the sake of exposition, we provide an arithmetic example here to demonstrate our proposed PLiF method. As mentioned in the introduction, the problem is as follows:

Problem 4.1 (Time sequence fingerprinting)**. Given** *m time sequences of length T,* **Extract** *features that match the four goals in the introduction.*

The four goals are that (a) the features should be effective, capturing the essence of what humans consider in similar sequences; (b) interpretable (c) they should lead to forecasting and (d) their computation should be fast and scalable.

More specifically, for the first goal of effectiveness, we want the fingerprints to have properties P1-P3, namely, *lag independence*, *frequency proximity* and *harmonics grouping*.

Thus, we use the following illustrative sequences (see Figure 4.3), of length T=500 time-ticks, as defined in Table 4.2.

Table 4.2: Illustrative sequences

Equation	Comment
(a) $X_1 = \sin(2\pi t/100)$	period 100
(b) $X_2 = \cos(2\pi t/100)$	time-shifted of (a)
(c) $X_3 = \sin(2\pi t/100) + \cos(2\pi t/100)$	time-shifted & higher amplitude
(d) $X_4 = \sin(2\pi t/110) + 0.2\sin(2\pi t/30)$	mixture of two waves of periods 110 and 30
(e) $X_5 = \cos(2\pi t/110) + 0.2\sin(2\pi t/30 + \pi/4)$	same as (d) but lag in both components

[2] Scalability has not been shown as all methods here have computation time linear to the length of data sequences.

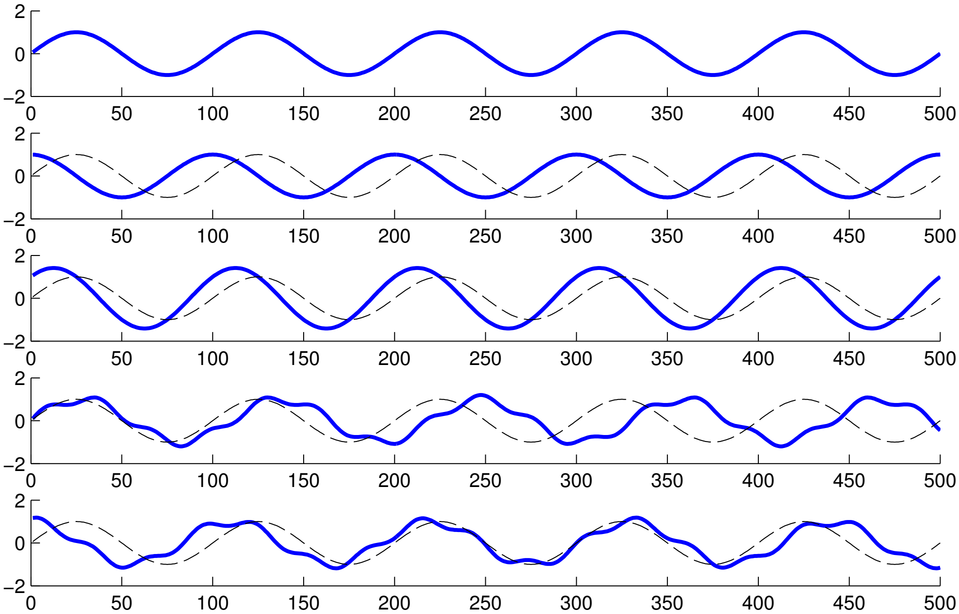


Figure 4.3: Running example: 5 synthetic sequences (top 3, with period 100 and possibly shifts; rest 2, with periods 110 and 30. All are generated with equations in Table 4.2. The first sequence is plotted as dotted reference lines. Note the first and fifth sequences share similar shapes, however they are generated by different process thus belong to different semantic groups.

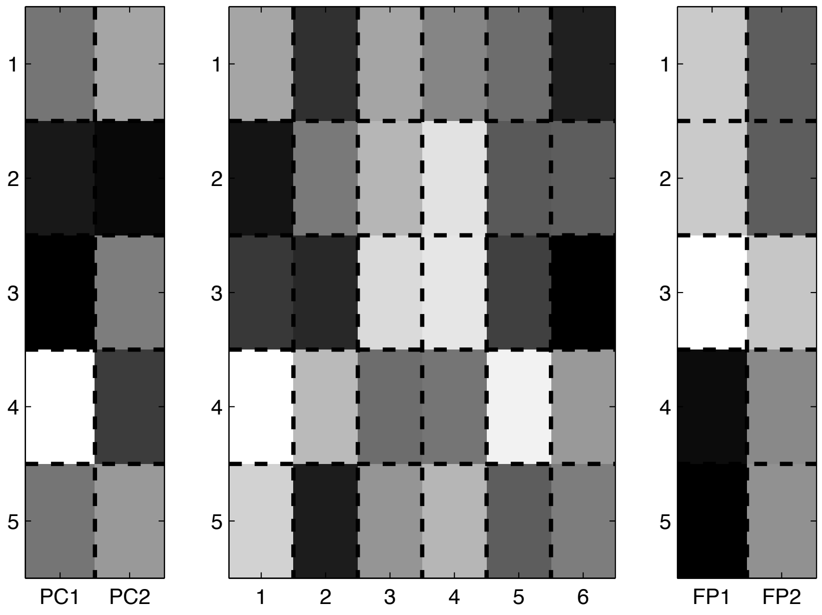
The first three sequences have period 100, with differing lags and amplitudes and thus we would like them to fall into the same group. The remaining two combine two frequencies (with periods 110 and 30), and a small phase difference; according to the P1-P3 properties, we would expect them to form another group of their own.

Fig. 4.3 also shows the first sequence, in dashed line form, so that we can visually compare the five sequences. As mentioned in the introduction, the typical method for dimensionality reduction (and thus feature extraction) is PCA/SVD; and the typical method for forecasting is autoregression and its more general form, Linear Dynamical Systems (LDS, where we specifically use the output matrix).

We show the resulting features for each method as gray-scale heat-maps (Fig. 4.4(a)-4.4(c)), where rows are sequences, columns are features (= fingerprints), and black color indicates high values. Rows that are visually similar means that they have similar feature values and thus would end up in the same cluster.

In short, only PLiF gives effective features. Specifically, notice that PCA (Fig. 4.4(a)) yields similar rows for sequence (a) and (e) - they are indeed visually similar in their time-plots, too, with small Euclidean distance (sum of square of daily differences). This is not surprising, because PCA and SVD actually preserve the Euclidean distance as best as possible - except that the Euclidean distance fails our desirable goals, heavily penalizing lags. Similarly using the output matrix from LDS (Fig. 4.4(b)) gives a poor grouping.

The heat-maps above explain why both PCA/SVD, as well as LDS, will lead to *poor clustering*, after we apply, say, k-means [Hastie et al., 2003]. In contrast, the heat-map of our proposed method PLiF (Fig. 4.4(c)) gives the expected groupings: the last two sequences are clearly together, with dark color in



(a) PCA (b) LDS (c) PLiF

Figure 4.4: Running example: extracted features for the sequences in Table 4.2 and Figure 4.3. 4.4(a),4.4(b),4.4(c): 'heat-maps' of fingerprints for each sequence, using PCA, LDS and PLiF, respectively. PLiF gives similar fingerprints for the top 3 and the bottom 2 sequences respectively, while competitors do not.

their first column; and the first three sequences have dark color in their second column.

As we show in later sections, PLiF also gives the interpretation for each feature: the corresponding "harmonic" groups and the strength of them in each of the sequences (Fig. 4.5(c)). We omit details here and will introduce the meaning of those matrix in Section 4.3. By visual inspection, we could clearly see the ideal group structure in the result by PLiF, while random groups by PCA. Although the above is synthetic data, such lag correlation and frequency combinations are common in time series data such as motion capture data and sensor data. We will show in later sections with real data that PLiF is effective in recognizing those correlation and will group similar sequences together in that sense, while PCA fails to capture the lag correlations.

4.2.1 Alternative methods or "Why not DFT, PCA or LDS?"

There are several existing methods for extracting features, but none matches all the desirable properties illustrated in Table 4.1. Thus, none is a head-on competitor to our proposed PLiF method. This is the reason we call them "quasi"-competitors.

- Fourier analysis and wavelet methods could identify the frequencies in a *single* signal, but can not cross-correlate similar signals, nor do forecasting[3]

[3] One might argue that Fourier coefficients can do rudimentary forecasting, since they can generate values for time ticks out-

- Singular value decomposition (SVD) and its "centered" version, principal component analysis (PCA), do capture correlations (by doing soft clustering of similar sequences) and thus derive hidden ("latent") variables, but they can not do forecasting either, nor is it easy to interpret the derived hidden variables.
- Standard Linear Dynamical Systems (LDS) and Kalman filters can capture correlations, as well as do forecasting. However, the resulting hidden variables are hard to interpret (see Fig. 4.4(b)), and they do not lead to a natural distance function, as we describe later in this section.

Finally, we do not show typical distance functions in Table 4.1: the Euclidean distance (sum of squares of differences at each time-tick) and the Dynamic Time Warping (DTW) distance. The reason is that none of them leads to forecasting, nor to feature extraction, and thus the interpretability requirement is out of reach. Moreover, the typical, un-constrained, version of DTW fails the scalability requirement, being quadratic on the length of the sequences.

We elaborate on PCA, Discrete Fourier Transform and Linear Dynamical Systems here because (a) they are the typical competitors for some (but not all) of the target tasks and (b) they can help in describing our PLiF method as well.

Principal Component Analysis Principal Component Analysis (PCA) is the textbook method of doing dimensionality reduction, by spotting redundancies and (linear) correlations among the given sequences. Technically, it gives the optimal low rank approximation for the data matrix $\mathbf{X}$.

In our running example of Section 4.2, the matrix $\mathbf{X}$ would be a 5×500 matrix, with one row for each sequence and one column per time-tick. Singular value decomposition (SVD) is the typical method to compute PCA.

For a data matrix $\mathbf{X}$ (assume $\mathbf{X}$ is zero-centered), SVD computes the decomposition

$$\mathbf{X} = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^T$$

where both $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices, and $\mathbf{S}$ is a diagonal matrix with singular values on the diagonal.

Using standard terminology from the PCA literature, $\mathbf{V}$ is called the *loading* matrix and $\mathbf{U} \cdot \mathbf{S}$ will be the component *score*. To achieve dimensionality reduction, small singular values are typically set to zero so that the retained ones maintain 80-90% of the energy (= sum of squares of eigenvalues). We shall refer to this rule of thumb as the *energy criterion* [Fukunaga, 1990] (equivalently, this truncation produces a low rank projection). In our running example of Figure 4.3, the $\mathbf{U} \cdot \mathbf{S}$ component score matrix is a 5×2 matrix, since we are retaining 2 hidden variables.

PCA is effective in dimensionality reduction and in finding linear correlations, particularly for Gaussian distributed data [Tipping and Bishop, 1999]. However, the low dimensional projections are hard to interpret. Moreover, PCA can not capture time-evolving patterns (since it is designed to *not* care about the ordering of the rows or the columns), and thus it can not do forecasting. Figure 4.4(a) shows the top two principal components for the synthetic five sequences. It does not show any clear pattern of underlying clusters; thus k-means indeed gives a poor final clustering result on it.

side the initial time range $(1, \ldots, T)$; however, these values will just lead to repeating the initial signal, with ringing phenomena if the signal has a trend.

Discrete Fourier Transform The T-point Discrete Fourier Transform (DFT) of sequence $(x_0, \ldots, x_{T-1})$ is a set of T complex numbers c_k, given by the formula

$$c_k = \sum_{t=0}^{\mathrm{T}-1} x_t e^{-\frac{2\pi i}{\mathrm{T}} kt} \qquad k = 0, \ldots, \mathrm{T} - 1$$

where $i = \sqrt{-1}$ is imaginary unit.

The c_k numbers are also referred to as the *spectrum* of the input sequence. DFT is powerful in spotting periodicities in a single sequence, with numerous uses in signal, voice, and image processing.

However, it is not clear how to assess the similarity between two spectra, and hence DFT can be unsuitable for clustering. Moreover, it has several limitations, namely:

1. it can not find arbitrary frequencies (only ones that are integer multiples of the base frequency),
2. it can not give partial credit for signals with nearby frequences ('frequency proximity', Property P2 in the introduction),
3. it can not do forecasting, other than blindly repeating the original signal.

Linear Dynamical Systems Linear Dynamical Systems (LDS), also known as Kalman filters, have been used previously to model multi-dimensional continuous valued time series. The model is described by the equations (2.4) (2.5), and (2.6):

The output matrix $\mathbf{C}$ in LDS could be viewed as features for the observation data sequence (see Fig. 4.4(b)). The difference between our proposed PLiF and LDS is that in addition to learning straight forward transitions and projections, PLiF will further discover deeper patterns behind them. The problem with LDS learning is that the learned model parameters are hard to interpret.

4.3 Parsimonious Linear Fingerprinting

We describe our proposed method PLiF (Parsimonious Linear Fingerprinting) in this section. First we give the basic version (PLiF-basic) and explain its steps and in later section we show how to make it even faster. Table 4.3 gives an overview of the symbols used and their definitions.

Goal To recap, we want to solve Problem 4.1, *Given* multiple time sequences of same duration T, *find* features which have the four properties namely: (a) Effective (can be used for visualization and clustering); (b) Meaningful; (c) Generative (can be used for forecasting and compression); and (d) Scalable.

Each following subsection describes a step in our algorithm. At the high level, PLiF (a) discovers the "Newton"-like dynamics, using a modified, faster way of learning an LDS; (b) normalizes the resulting *transition matrix* $\mathbf{A}$, which reveals the natural frequencies and exponential decays / explosions of the given set of sequences (which we refer to as *harmonics*, see Definition 4.1); and (c) groups some of those harmonics/hidden variables, after ignoring the phase, thus accounting for lag-correlations. The discovered groups of such frequencies are exactly the "fingerprints" (or features) that PLiF is using for clustering, visualization, compression, etc.

Table 4.3: List of symbols in PLiF

m	number of sequences
T	duration (length) of sequences
h	number of hidden variables for each time tick
h'	number of harmonics excluding conjugates
$\mathbf{X}$	data matrix, $m \times \mathrm{T}$
$\vec{z}_n$	hidden variables for time n, $h \times 1$ vector
$\vec{\mu}_0$	initial state for hidden variable, $h \times 1$ vector
$\mathbf{A}$	transition matrix (like “Newton dynamics”), $h \times h$
$\mathbf{C}$	output matrix (“hidden to observation”) $m \times h$
$\mathbf{V}$	compensation matrix, eigenvectors of $\mathbf{A}$, $h \times h$
$\mathbf{\Lambda}$	eigen-dynamics matrix (eigenvalues of $\mathbf{A}$), $h \times h$
$\mathbf{C}_h$	harmonic mixing matrix, $m \times h$
$\mathbf{C}_m$	harmonic magnitude matrix, $m \times h'$
$\mathbf{F}$	fingerprints

4.3.1 Learning Dynamics

Intuition To understand the hidden pattern in the multiple signals, we want to extract the hidden dynamics, similar to “Newtonian” dynamics like velocities and accelerations. Our basic intuition is to assume that there is a series of hidden variables, representing the states of the hidden pattern, which are evolving according to a linear transformation and are linearly transformed to the observed numerical sequences.

Theory To obtain the dynamics in data, we use an underlying Linear Dynamical System (LDS) to model multiple time series. We use the EM algorithm (described in Section 2.2.5) for learning the model parameters in the following equations (repeat of Eq. (2.4-2.6)). In our implementation, we can use the DynaMMo (Algorithm 3.1 in Chapter 3) to learn the parameters of LDS without missing values.

$$\vec{z}_1 = \vec{\mu}_0 + noise \qquad \vec{z}_{n+1} = \mathbf{A}\vec{z}_n + noise \qquad \vec{x}_n = \mathbf{C}\vec{z}_n + noise$$

The LDS model includes parameters of an initial state vector $\vec{\mu}_0$, a *transition matrix* $\mathbf{A}$ and an *output matrix* $\mathbf{C}$ (along with the noise covariance matrices). Similar to “Newtonian” dynamics, the transition matrix $\mathbf{A}$ will predict the hidden variables (like the velocity and acceleration in human motions) for the next time tick, and the output matrix $\mathbf{C}$ will tell us how the hidden variables (e.g. velocities and accelerations) map to the observed values (e.g. positions) at each time tick (each row of $\mathbf{C}$ corresponds to one sequence).

Note that as discussed before, the transition matrix $\mathbf{A}$ is not unique: it is subject to permutation, rotation and linear combinations, and so is the *output matrix* $\mathbf{C}$. Thus each row in $\mathbf{C}$ can not uniquely identify the characteristics of the corresponding series. We therefore need features that are “invariant” for each time sequence. The same set of observations can generate totally different transition matrices. Hence it is hard to interpret it. For example, suppose we have two LDSs with the same model parameters except the transition matrices. $\mathbf{A}_1$ and $\mathbf{A}_2$, but $\mathbf{A}_2$ is a similarity transformation of $\mathbf{A}_1$, i.e. $\mathbf{A}_2 = \mathbf{P} \cdot \mathbf{A}_1 \cdot \mathbf{P}^*$ with an arbitrary non-singular square matrix $\mathbf{P}$. Here $\mathbf{P}^*$ is the Hermitian transpose of $\mathbf{P}$ i.e. $\mathbf{P}^* = \overline{\mathbf{P}}^T$.

It is easily seen that given a suitable initial state both models can generate functionally *isomorphic* series of hidden variables (the hidden variables will be related by the matrix $\mathbf{P}$). Thus using any matrix *similar* to $\mathbf{A}_1$ as the transition matrix can generate isomorphic series of hidden variables. Note that our goal is to extract the invariant components from the hidden dynamics. So we want to extract the *commonality* of this set of *similar* matrices [4]. Our subsequent steps are motivated by this observation.

Example On using $h = 6$ hidden variables to learn the parameters for the 5 synthetic sequences shown in Figure 4.3, we will get a 6×6 transition matrix $\mathbf{A}$, a 5×6 output matrix $\mathbf{C}$ and a 6×1 initial state vector $\vec{\mu}_0$. Figure 4.4(b) shows the $\mathbf{C}$ matrix. Clearly, it is all jumbled up and doesn't show any clear pattern w.r.t. the sequences.

4.3.2 Canonicalization

Intuition Equations of the linear system (see Section 2.2.5, Eq. 2.5) tell that the hidden variables ($\vec{z_n}$) can have only a limited number of modes of operation that depend on the eigenvalues of the $\mathbf{A}$ matrix: The behavior can be exponential decay (real eigenvalue, with magnitude less than 1), exponential growth (real eigenvalue, with magnitude greater than 1), sinusoidal periodicity of increasing / constant / decreasing amplitude (complex eigenvalue $a + bi$ controlling both amplitude and frequency) and mixtures of the above. Those eigenvalues directly capture the amplitude and frequencies of the underlying signals of *hidden* variables, which we refer to as *harmonics* (Definition 4.1). Our goal in this step is to identify the canonical form of the hidden harmonics and how they mix in the observation sequences.

Theory We know that a set of *similar* matrices share the same eigenvalues [Golub and Van Loan, 1996]. Hence, we propose to perform the eigenvalue decomposition of the transition matrix $\mathbf{A}$, and obtain the corresponding eigen-dynamics matrix and eigenvectors.

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^* \tag{4.1}$$

where $\mathbf{V} \cdot \mathbf{V}^* = I$. The matrix $\mathbf{V}$ contains the eigenvectors of $\mathbf{A}$ and $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues of $\mathbf{A}$. We can justify doing the decomposition because over $\mathbb{C}$ almost every matrix is diagonalizable. Specifically, the probability that a square matrix of arbitrary fixed size with real entries is diagonalizable over $\mathbb{C}$ is 1 (see Zhang [Zhang, 2001]). Also without loss of generality, we assume the eigenvalues are grouped into conjugate pairs (if any) and ordered according to their phases.

Note that the *output matrix* $\mathbf{C}$ in LDS represents how the hidden variables translate into observation sequences with linear combinations. In order to obtain the same observation sequences from $\mathbf{\Lambda}$ as the transition matrix, we need to compensate the *output matrix* $\mathbf{C}$ to get the *harmonic mixing matrix* $\mathbf{C}_h$.

$$\mathbf{C}_h = \mathbf{C} \cdot \mathbf{V} \tag{4.2}$$

Similarly, the *canonical hidden variables* will be:

$$\vec{\mu}_0^{new} = \mathbf{V}^* \cdot \vec{\mu}_0 \tag{4.3}$$

$$\vec{z}_n^{new} = \mathbf{V}^* \cdot \vec{z}_n \tag{4.4}$$

[4] Two matrices $\mathbf{A}$ and $\mathbf{B}$ are called *similar* if there exist an orthonormal matrix $\mathbf{U}$ such that $\mathbf{A} = \mathbf{U} \cdot \mathbf{B}$

The following two lemmas tell how the *harmonic mixing matrix* $\mathbf{C}_h$ looks like and how the canonical hidden variables correspond to the original dynamical system:

Lemma 4.1. $\mathbf{V}$ *has conjugate pairs of columns corresponding to the conjugate pairs of eigenvalues in* $\mathbf{\Lambda}$*. Hence, the* harmonic mixing matrix $\mathbf{C}_h$ *must contain conjugate pair of columns corresponding to the conjugate pairs of the eigenvalues in* $\mathbf{\Lambda}$*.*

Proof Sketch: Consider the eigenvalue equation $\mathbf{A} \cdot x = \lambda x$*, where* x *is the eigenvector. Taking the conjugate on both sides we get* $A \cdot \overline{x} = \overline{\lambda x}$*. As* $\mathbf{A}$ *contains only real entries,* $\mathbf{A} \cdot \overline{x} = \overline{\lambda x}$*. Hence, if the conjugate* $\overline{\lambda}$ *is an eigenvalue of* $\mathbf{A}$*,* $\overline{x}$ *is also a corresponding (conjugate) eigenvector.* □

Lemma 4.2.

$$\vec{z}_n^{new} = \mathbf{\Lambda}^{n-1} \cdot \vec{\mu}_0^{new} + noise \tag{4.5}$$

$$\vec{x}_n = \mathbf{C}_h \cdot \mathbf{\Lambda}^{n-1} \cdot \vec{\mu}_0^{new} + noise \tag{4.6}$$

Proof:

$$\begin{aligned}
\vec{z}_n^{new} &= \mathbf{V}^* \cdot \vec{z}_n \\
&= \mathbf{V}^* \cdot \mathbf{A} \cdot \vec{z}_{n-1} + noise \\
&= \mathbf{V}^* \cdot \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^* \cdot \vec{z}_{n-1} + noise \\
&= \mathbf{\Lambda} \cdot \vec{z}_{n-1}^{new} + noise \\
\vec{x}_1 &= \mathbf{C} \cdot \vec{\mu}_0 + noise = \mathbf{C} \cdot \mathbf{V} \cdot \mathbf{V}^* \cdot \vec{\mu}_0 + noise \\
&= \vec{C}_h \cdot \vec{\mu}_0^{new} + noise \\
\vec{x}_2 &= \mathbf{C} \cdot \vec{z}_2 + noise = \mathbf{C} \cdot \mathbf{A} \cdot \vec{z}_1 + noise \\
&= \mathbf{C} \cdot \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^* \cdot \vec{\mu}_0 + noise \\
&= \mathbf{C}_h \cdot \mathbf{\Lambda} \cdot \vec{\mu}_0^{new} + noise
\end{aligned}$$

The result then follows by induction on the number of time ticks. □

The intuition is that all hidden variables $\vec{z}_n$, all canonical hidden variables $\vec{z}_n^{new}$, and all observations $\vec{x}_n$ ($n = 1, \ldots, T$) are mixtures of a set of growing, shrinking or stationary sinusoid signals, of data-dependent frequencies; we refer to those signals as "*harmonics*", and their characteristic frequencies and amplitudes are completely defined by the eigenvalues of the *transition matrix* $\mathbf{A}$. "Harmonics" are formally defined as:

Definition 4.1. *A signal* y_n *is said to be a harmonic if it is in the form of* $y_n = (a + bi)^n$*, where* $a + bi$ *is an eigenvalue of* $\mathbf{A}$ *and* $i = \sqrt{-1}$*.*

Compared to Fourier analysis which may identify a spectrum of frequencies for a single signal, our eigen-dynamics approach identifies the "common" underlying frequencies across all signals. In the case of small h, our method will force the collapse of nearby frequencies, as required by P2 in Section 4.1.

Our definition of *harmonic* is related to the frequencies that the Fourier transform would discover, with two major differences: (a) *exponential amplitude*: our harmonic functions could be growing or shrinking exponentially (for $a \neq 1$) (b) *generality*: the frequencies of the Discrete Fourier Transform (DFT) are always multiples of the base frequency $1/T$, while our harmonics could have any arbitrary frequency (b could take any value that fits the given data sequences).

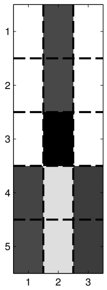
Example On performing this step on the learned *transition matrix* from the 5 synthetic sequences, we will get a 6×6 $\mathbf{\Lambda}$ matrix with eigenvalues $0.998 \pm 0.057i$, $0.998 \pm 0.063i$, and $0.978 \pm 0.208i$ on the diagonal. From our above discussion, we know that when the real part of the eigenvalue = 1, then the signal is a sinusoid - which is the case here. The imaginary part on the other hand corresponds to the frequencies. Thus $0.057 \approx 2\pi/110$ corresponds to frequency 1/110, $0.063 \approx 2\pi/100$ corresponds to frequency 1/100 and $0.208 \approx 2\pi/30$ corresponds to frequency 1/30 - which are precisely the base frequencies in the data.

$$\begin{array}{cccccc}
\multicolumn{6}{c}{\left|\begin{array}{cccccc}
0 & 0 & 0.46e^{-0.82i} & 0.46e^{0.82i} & 0 & 0 \\
0 & 0 & 0.46e^{0.75i} & 0.46e^{-0.75i} & 0 & 0 \\
0 & 0 & 0.65e^{-0.03i} & 0.65e^{0.03i} & 0 & 0 \\
0.50e^{0.70i} & 0.50e^{-0.70i} & \approx 0 & \approx 0 & 0.53e^{-0.48i} & 0.53e^{0.48i} \\
0.50e^{-0.87i} & 0.50e^{0.87i} & \approx 0 & \approx 0 & 0.53e^{0.31i} & 0.53e^{-0.31i}
\end{array}\right|} \\
\multicolumn{2}{c}{\underbrace{\hspace{1.5cm}}_{\text{Harmonic pair } H_1}} & \multicolumn{2}{c}{\underbrace{\hspace{1.5cm}}_{\text{Harmonic pair } H_2}} & \multicolumn{2}{c}{\underbrace{\hspace{1.5cm}}_{\text{Harmonic pair } H_3}}
\end{array}$$

(a) $\mathbf{C}_h$

$$\left|\begin{array}{ccc}
0 & 0.46 & 0 \\
0 & 0.46 & 0 \\
0 & 0.65 & 0 \\
0.50 & \approx 0 & 0.53 \\
0.50 & \approx 0 & 0.53
\end{array}\right|$$

(b) $\mathbf{C}_m$



(c) $\mathbf{C}_m$ heat-map

Figure 4.5: Running example: the synthetic, sinusoidal signals of Figure 4.3, and the output matrices according to PLiF: (a) The *harmonic mixing matrix* $\mathbf{C}_h$ and (b) the *harmonic magnitude matrix* $\mathbf{C}_m$ and (c) its heat-map (darker color - higher value in that cell). Near-zero values: omitted for clarity. Notice that (1) the columns of (a) are complex conjugates, in pairs; (2) the *harmonic magnitude matrix* $\mathbf{C}_m$ makes similar sequences to look similar (top 3, bottom 2).

Figure 4.5(a) shows the matrix $\mathbf{C}_h$ obtained for the sample signals. We have shown the entries in the standard polar form: $Ae^{\phi i}$, A is the magnitude and ϕ is the angle (phase). For clarity, the values which are very small have been shown as 0 in the matrix. Firstly as expected from Lemma 4.1, we have conjugate pairs of columns corresponding to the eigenvalues which correspond to the frequencies. Secondly, note that signals (a) and (b) are the same sinusoid but with just different phases (specifically a phase difference of $\pi/2$). Hence in the matrix $\mathbf{C}_h$ they have the same frequency components (high values only in the conjugate columns 3 and 4) with the same weights (same A value) but with *different* phases (different ϕ values). So, if we directly try to cluster $\mathbf{C}_h$, we will not place them in the same cluster. Thirdly, the phase difference is also preserved in $\mathbf{C}_h$: $0.82 + 0.75 = 1.5708 = \pi/2 =$ the phase difference between signals (a) and (b). This can also be verified for the signals (d) and (e) which have different phases for the two constituent frequencies.

4.3.3 Handling Lag Correlation: Polar Form

Intuition As specified in Section 4.1, the ideal features should catch lag correlation. After computing the *harmonic mixing matrix* $\mathbf{C}_h$, we have found the contribution of each harmonic in the resulting observation sequences. Each row in $\mathbf{C}_h$ represents the characteristics of each data sequence in the domain of the harmonics. Thus $\mathbf{C}_h$ can plausibly be used to cluster the sequences. However, the harmonic mixing matrix not only tells the strength of each eigen-dynamic but will also encode the required *phases* for different sequences. Thus we will fail to group similar motions just due to the lag or phase differences. Intuitively for example, suppose we have two almost identical walking motions, except that one starts from the left foot and another from the right foot. We want to extract features that could identify the walking behavior, no matter which foot it starts with, so that we would be able to group the two walking motions together.

Theory We eliminate phase by taking the magnitude of the *harmonic mixing matrix* $\mathbf{C}_h$ $abs(\mathbf{C}_h)$. From Lemma 4.1 we will get the same column for those conjugate columns of $\mathbf{C}_h$; we drop these duplicate columns to get the *harmonic magnitude matrix* $\mathbf{C}_m$. The *harmonic magnitude matrix* $\mathbf{C}_m$ tells how strong each base harmonic participates in the observation time sequences and naturally leads to lag independent features (P1, Section 4.1).
Lemma 4.3. $abs(\mathbf{C}_h)$ *contains pairs of identical columns.*

Example Figure 4.5 (b) and (c) show the matrix $\mathbf{C}_m$ obtained after applying this step on the generated $\mathbf{C}_h$ matrix for the synthetic signals. Note that $\mathbf{C}_m$ has begun to show some clear patterns corresponding to the underlying true clusters.

4.3.4 Grouping Harmonics

Intuition The *harmonic magnitude matrix* $\mathbf{C}_m$ captures the contributing coefficients of each individual frequency. As we find harmonic frequency sets in music, in real time-series like motions, we will expect to usually find motion sequences composed of several major frequencies. Hence we now want to find such harmonics groups (P3 as stated in Section 4.1) capable of describing common characteristics of similar motion sequences, and the corresponding representations of each sequence in such harmonics group space. As a concrete example, say we want to determine that walking sequences are composed of 10 *units* of frequency 1 and 1 units of frequency 2, while running motions have say 10 units of frequency 2 and 1 units of frequency 3. Furthermore, a fast walking motion may in fact have a proper mix of both a walking frequency-group and running frequency-group.

Theory To achieve this goal, we can use any dimensional reduction method such as SVD/PCA, ICA or nonnegative matrix factorization. For simplicity, we take the singular value decomposition (SVD) of the *harmonic magnitude matrix* $\mathbf{C}_m$. As we introduced earlier, SVD is capable of finding low rank projection of the data matrix. $\bar{\mathbf{C}_m} \approx \mathbf{U}_k \cdot \mathbf{S}_k \cdot \mathbf{V}_k^T$ where $\bar{\mathbf{C}_m}$ is column centered from $\mathbf{C}_m$, $\mathbf{U}_k$ and $\mathbf{V}_k$ are orthogonal matrices with k columns, and $\mathbf{S}_k$ is a diagonal matrix. The diagonal of $\mathbf{S}_k$ contains k singular values which are usually sorted by magnitude. Finally, we obtain the features as follows:

$$\mathbf{F} = \mathbf{U}_k \cdot \mathbf{S}_k \tag{4.7}$$

Example. Figure 4.4(c) shows the final $\mathbf{F}$ matrix obtained from the sample sequences. Note that this matrix very clearly brings out the correct groupings. Also notice that in the corresponding $\mathbf{C}_m$ matrix, sequences (d) and (e) had high components on columns 1 and 3 (which map to the 2 frequencies generating those signals). But after doing SVD, $\mathbf{F}$ gives us a clearer and simpler picture where they are shown to be more related by having the same 'group' of harmonics combined in the same way.

We summarize our algorithm in Algorithm 4.1.

Algorithm 4.1: PLiF

```
   Input: $\mathbf{X}$: $m$ sequences with duration T, and $k$
   Output: fingerprints $\mathbf{F}$
 1 choose $h$ by 80%-95% energy criterion ;
   // learning dynamics (see Sec.4.3.1 and Eq.(2.7))
 2 $\mathbf{A}, \mathbf{C} \leftarrow \arg\max_\theta L(\theta; \mathbf{X})$ ;
   // canonicalization, see Sec.4.3.2
 3 compute $\mathbf{\Lambda}$, $\mathbf{V}$ s.t. $\mathbf{A} \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{\Lambda}$;
   // compensating, see Sec.4.3.2
 4 $\mathbf{C}_h = \mathbf{C} \cdot \mathbf{V}$;
   // obtain polar form, see Sec.4.3.3
 5 $\mathbf{D} \leftarrow$ keep conjugate pairs of columns in $\mathbf{C}_h$;
 6 $\mathbf{E} \leftarrow$ take element-wise magnitude of $\mathbf{D}$;
 7 $\mathbf{C}_m \leftarrow$ eliminate duplicate columns in $\mathbf{E}$;
   // finding harmonics grouping, see Sec.4.3.4
 8 $\overline{\mathbf{C}_m} \leftarrow \mathbf{C}_m - \text{mean}(\mathbf{C}_m)$;
 9 compute $\mathbf{U}_k, \mathbf{S}_k, \mathbf{V}_k \leftarrow \arg\min ||\overline{\mathbf{C}_m} - \mathbf{U}_k \cdot \mathbf{S}_k \cdot \mathbf{V}_k^T||_{fro}$ ;
10 $\mathbf{F} \leftarrow \mathbf{U}_k \cdot \mathbf{S}_k$;
```

4.3.5 Discussion

Choosing h: A particular issue in the learning algorithm is choosing a proper number h for the hidden dimension of underlying LDS. In practice, we use the "80%-95%" energy criterion to determine h [Fukunaga, 1990]. That is, we take the Singular Value Decomposition of $\mathbf{X}$, rank the singular values, and then choose h at the rank with 95% of the total sum of squared singular values: $\hat{h} \leftarrow \arg_h \frac{\sum_{j=1}^{h} s_j^2}{\sum_{i=1}^{m} s_i^2}$, where s_i's are singular values of $\mathbf{X}$ in descending order.

Complexity:

Lemma 4.4. *The straightforward implementation of the algorithm (refered to as PLiF-naïve) costs $O(\#iteration \cdot \mathrm{T} \cdot (m^3 + h^3))$, where $\#iteration$ is the number of iterations for learning LDS.*

Proof: In each iteration of learning LDS (Alg. 4.1, step 2), it does $m \times m$ and $h \times h$ matrix inversion for T *length of sequences. Rest all steps including eigen value decomposition in canonicalization, taking polar form and grouping harmonics with SVD, take at most $O(m^3)$. Therefore, PLiF-naive has complexity of $O(\#iteration \cdot \mathrm{T} \cdot (m^3 + h^3))$.* □

The time complexity of PLiF-naive is linear with respect to the length of sequences, but cubic to the number of sequences. Without going too much into the details, we describe briefly where such cubic complexity arises in PLiF-naive. The major cubic computation involves calculation of the inverse of the following $m \times m$ matrix while learning LDS in the first step:

$$(\mathbf{C}\mathbf{P}_n\mathbf{C}^T + \mathbf{R})^{-1} \tag{4.8}$$

Here $\mathbf{C}$ is size $m \times h$, $\mathbf{P}_n$ is $h \times h$, and $\mathbf{R}$ is $m \times m$. $\mathbf{P}_n$ is a complicated matrix, hence we omit details of $\mathbf{P}_n$ for brevity (full details can be found in [Kalman, 1960]). This is updated in each time tick while learning the LDS model (Alg. 4.1, step 2). Cubic computation elsewhere is only a negligible fraction in the typical case of $m \ll \mathrm{T}$.

However, using the Woodbury matrix identity [Golub and Van Loan, 1996] and incrementally computing inverse of covariance matrix [Papadimitriou et al., 2005], the complexity can be reduced dramatically as stated in Lemma 4.5.

Lemma 4.5. *PLiF can be computed within time of* $O(\#iteration \cdot (\mathrm{T} \cdot (m^2 \cdot h + h^3)) + m \cdot h^2)$.

4.4 Interpretation and Evaluation

4.4.1 Experimental Setup

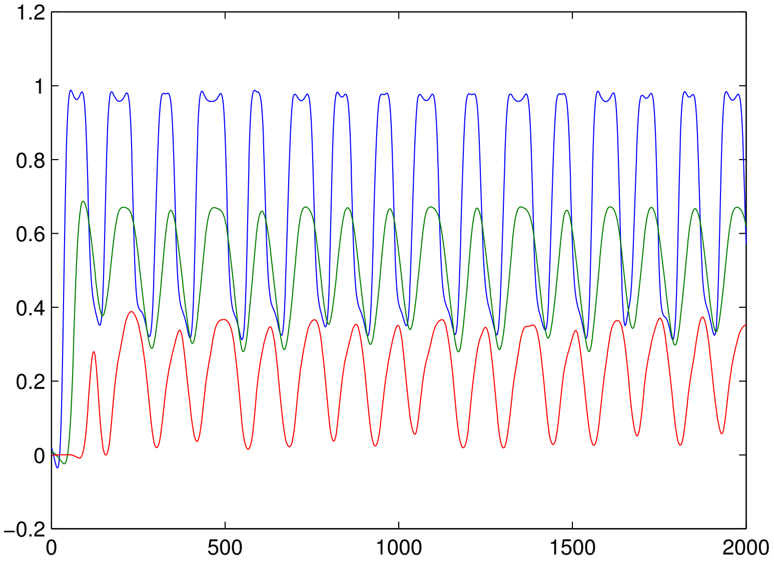
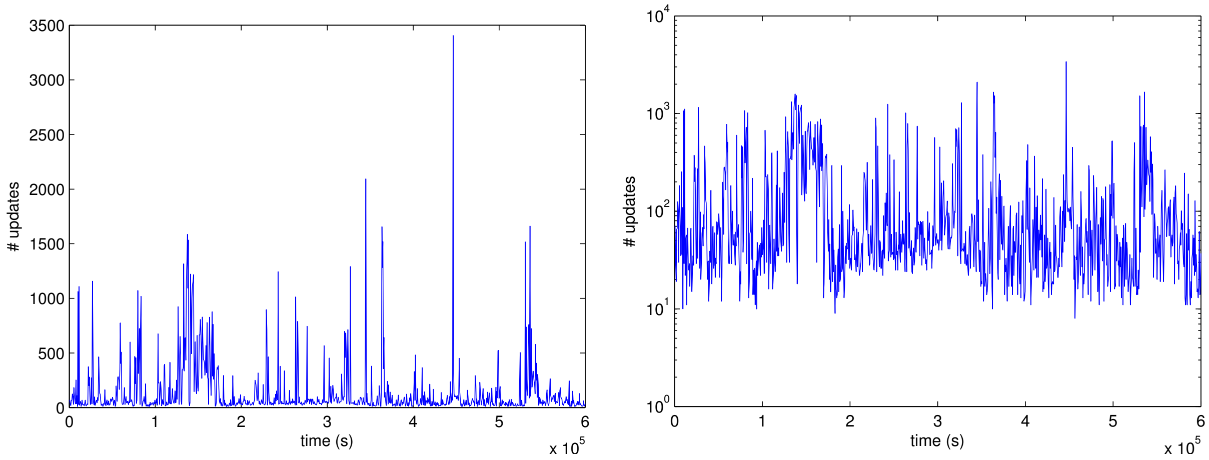


Figure 4.6: Sample snippets from CHLORINE dataset. CHLORINE shows daily periodicity.

We describe here our experimental setup and datasets.

- Mocap data (MOCAP): Motion capture involves recording human motion through tracking the marker movement on human actors and then turn them into a series of multi-dimensional coordinates in 3d space. We use a publicly available mocap dataset from CMU[5]. It includes 49 walking and running

[5] http://mocap.cs.cmu.edu



(a) BGP: for the router at Washington DC

(b) BGP (Washington DC), in log scale

Figure 4.7: Sample snippets from BGP dataset. (a) BGP is bursty with no periodicities, thus we take the logarithm (shown in part (b)). No obvious patterns, in neither (a) nor (b).

motions of subject#16. Each motion sequence contains 93 positions for 31 markers in body local coordinate and three reference coordinates.

- Chlorine Data (CHLORINE): The chlorine dataset is publicly available[6] and it contains m=166 sequences of Chlorine concentration measurements on a water network over 15 days at the rate of one sample per 5 minutes (T=4310 time ticks). The dataset was produced by the EPANET 2 hydraulic analysis package[7], which reflects periodic patterns (daily cycles, dominating residential demand pattern) in the Chlorine concentration, with a few exceptions and time shifts (see sample sequences in Figure 4.6).
- Router Data (BGP): We examine BGP Monitor data containing 18 million BGP update messages over a period of two years (09/2004 to 09/2006) from the Datapository project[8]. We consider the number of updates received by a router every 10 minutes. A snippet is shown in Fig. 4.7(a). As the signals are very bursty, we take their logarithms (see Fig. 4.7(b)). The pre-processed BGP time series in the experiment consists of m=10 sequences (routers) of T=103,968 time ticks. The routers are in 10 major centers (Atlanta, Washington DC, Seattle etc.).

Our algorithms are implemented in Matlab 2008b, and running on a machine with Windows XP, 3.2GHz dual core CPU and 2G RAM.

4.4.2 Effectiveness: Visualization

As stated in the introduction section, our proposed method PLiF is capable of producing meaningful features: each feature column corresponds to a group of “harmonic” frequencies (one or more) and features represent the participation coefficients of the harmonic group in the sequence.

[6] www.cs.cmu.edu/afs/cs/project/spirit-1/www/data/cl2fullLarge.zip

[7] http://www.epa.gov/nrmrl/wswrd/dw/epanet.html

[8] http://www.datapository.net/bgpmon/

For the MOCAP dataset, we found interpretable and interesting patterns in its fingerprints (Fig. 4.8). In our experiment, we use hidden dimension $h = 7$ as suggested by the 95% criteria, and produce two fingerprints for each sequence ($k = 2$). The walking motions exhibit strong correlation with harmonics with eigen-values $0.998 \pm 0.053i$, equivalent to the frequency of $1/119$, while the running ones are correlated with eigenvalues $1.007 \pm 0.082i$ and $0.989 \pm 0.108i$, equivalent to the frequencies of $1/78$ and $1/58$.

We already presented meaningful features, both visually and numerically, extracted from multiple sequences by our proposed PLiF method. Thanks to those features, PLiF can be readily used for almost all mining tasks for time series, namely clustering, compression, forecasting and segmentation. While forecasting and segmentation are straightforward brought by the underlying dynamical system of our method, we will focus on the particular application of PLiF in time series clustering and compression.

4.4.3 Effectiveness: Clustering

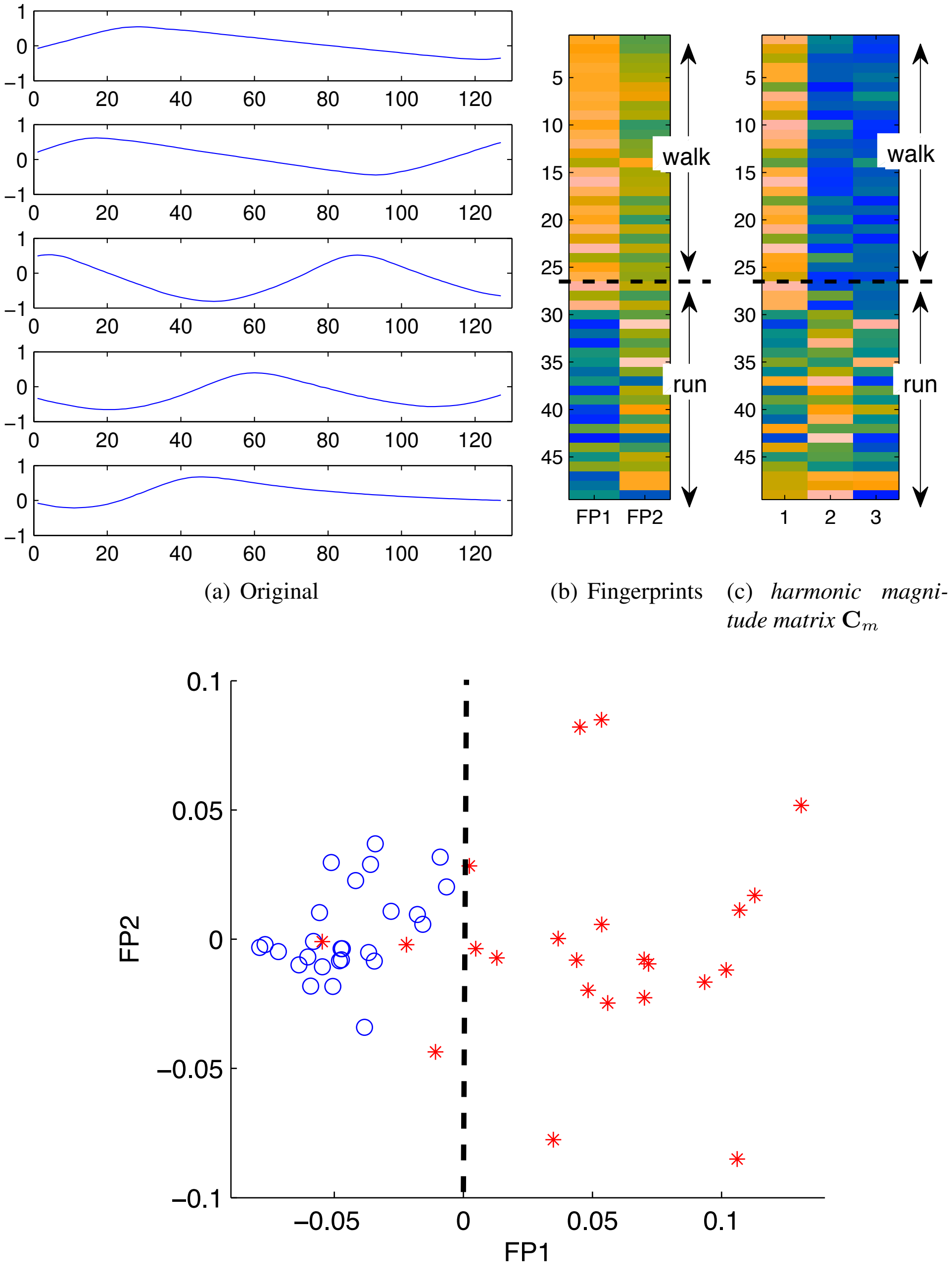
The rationale in our clustering method lies in the fact that the fingerprints (features) computed by PLiF characterize how much each "harmonic" group participates in each of the sequences. Essentially, such a fingerprint tells the projection of each sequence onto the basis of the "harmonic" group. The final clustering result can be then obtained by applying any state-of-the-art clustering algorithm, such as k-means or spectral clustering [Hastie et al., 2003] (Chap 14.3.6 & Chap 14.5.3) on the fingerprints.

In our experiments, we use simple thresholding (=0) on the first fingerprint (FP1) to tell the group, equivalent to k-means on FP1. In this way PLiF can produce two class grouping. But it can be easily extended to handle multiple class case, through the hierarchical framework [Hastie et al., 2003] (Chap 14.3.12): applying PLiF-clustering in each level to produce bi-clustering and further dividing in proper descendants.

In MOCAP we test the clustering result on the right foot marker position with sequences of equal length ($\mathrm{T} = 107, m = 49$). Since we know the true labels of each motion in MOCAP, we adopt a standard measure of conditional entropy from the confusion matrix of prediction labels against true labels to evaluate the clustering quality. The conditional entropy (CE) tells the difference of two clustering (lower is better), based on the following equation: $CE = -\sum \frac{CM_{ij}}{\sum_{ij} CM_{ij}} \log \frac{CM_{ij}}{\sum_j CM_{ij}}$.

We use a commonly practiced method as the baseline for comparison: first projecting the multiple sequences into low dimensional principal components (#dim=#class=2) and then clustering by k-means with Euclidean distance. Tab. 4.4(a) and 4.4(b) show the confusion matrices and their conditional entropy scores from the predicted grouping by the baseline and by PLiF clustering respectively. Note while baseline makes nearly random guesses, our method could identify all walking and almost all running motions correctly. The only three (out of 49) mistakes by PLiF turn out to be two running to stop motions and a right turn. As a typical example in Fig. 4.8(a), those mistakes have a very similar pattern with walking motion, so that even human would be confused.

As an exploratory example, we use PLiF-clustering to find groups on BGP data - we do not have ground-truth labels of each sequence here. Fig. 4.9 shows the results (each cluster is shown encircled). Note that the results match well with the notion that geographically closer routers tend to be more correlated than others. This is because the BGP routing protocol itself tries to find shorter routes which results in packets being sent locally to nearby routers rather than routers far away. Thus closer routers may have time shifts and correlations that are captured by PLiF.



(a) Original

(b) Fingerprints

(c) *harmonic magnitude matrix* $\mathbf{C}_m$

(d) Scatter plot

Figure 4.8: Mocap fingerprints and visualization. 4.8(a) displays several sample sequences, top two of which are walking (#15 and #22), followed by two running ones (#45 and #38) and a running-to-stop motion (#8). 4.8(b): Each motion(row) displays two fingerprints. Upper 26 rows are walking motion, and the rest are running motion. 4.8(d): Walking motion are in blue circles, and running in red stars. Note the three red stars close to circles turn out to be abnormal motions: running to stop (#8 and #57), and right turn (#43).

Table 4.4: Clustering on `MOCAP` right foot marker z-coordinate: Confusion matrix and conditional entropy. Note the ideal confusion matrix will be diagonal, which has conditional entropy of 0. Note in both way PLiF wins.

(a) PCA-Kmeans: $CE = 0.68$

	walk	run
predicted		
-1	15	13
1	11	10

(b) PLiF: $CE = 0.18$

	walk	run
predicted		
-1	26	3
1	0	20

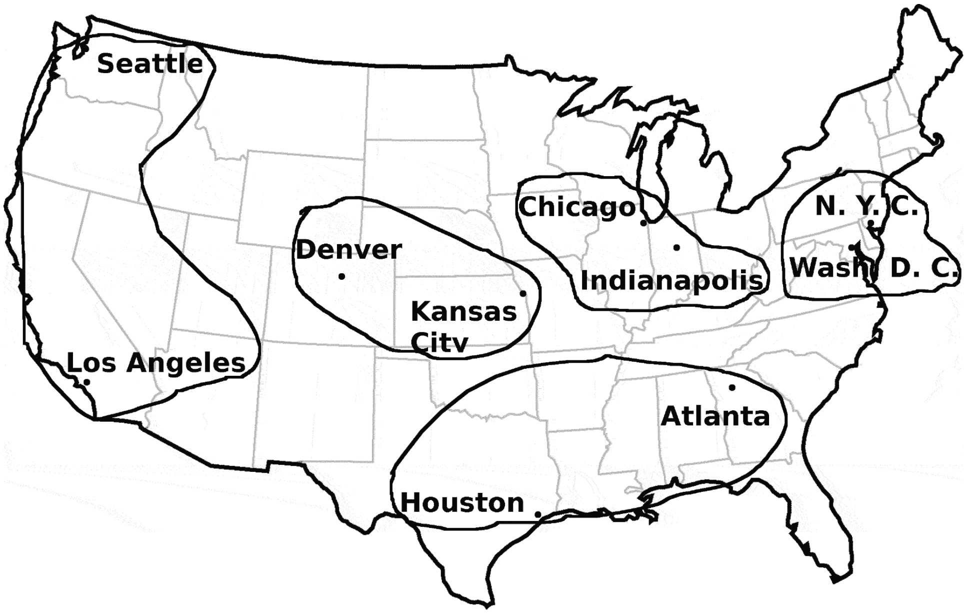


Figure 4.9: PLiF-clustering on BGP traffic data. Note how geographically close routers have been clustered together.

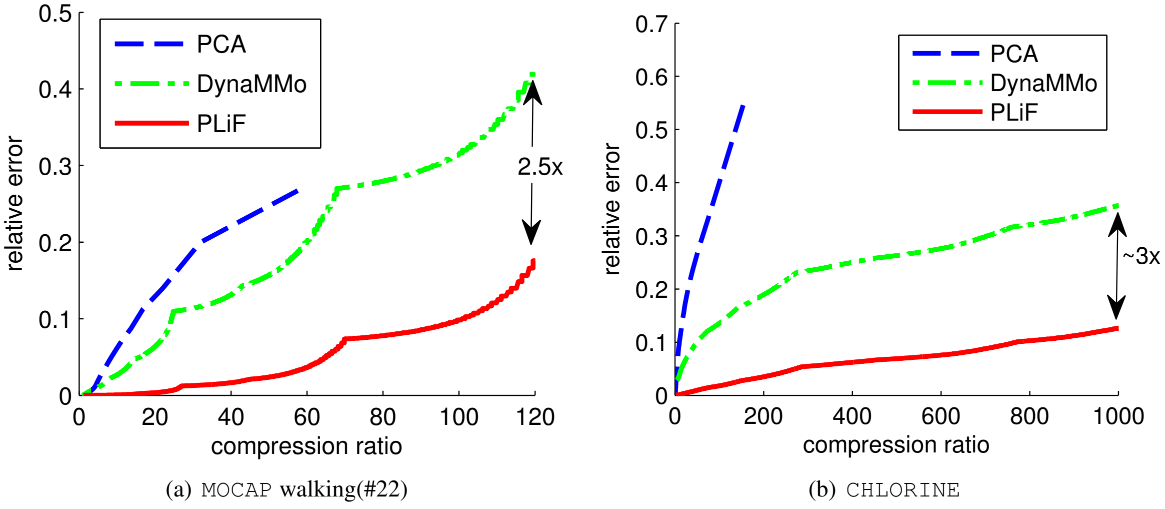
4.4.4 Compression

The fingerprints extracted by PLiF could be used in a compression setting as well. The basic idea is to store the eigen-dynamics matrix ($\mathbf{\Lambda}$), its associated projection matrix ($\mathbf{C}_h$) and a subset of expected value of hidden variables. From Section 4.3.2, the eigen-dynamics $\mathbf{\Lambda}$ is a diagonal matrix, so we only keep the diagonal part. We also keep $\mathbb{E}[\vec{z}_i]$ computed from the E-step of EM algorithm for LDS. To be able to recover from compression, we compute the hidden values using $\vec{\mu}_i = \mathbf{V}^* \cdot \mathbb{E}[\vec{z}_i]$. PLiF-compression finds a subset of $J \subseteq \{1, \dots, \mathrm{T}\}$, determining which time tick of hidden values will be stored. Here we use a similar idea as DynaMMo compression [Li et al., 2009] to select the best subset of time tick index using dynamics programming. To recover the original signal, we project back the data matrix from those hidden variables and dynamics using the following equations:

$$\vec{x}_i = \mathbf{C}_h \cdot \vec{\mu}_i \tag{4.9}$$

$$\vec{\mu}_j = \mathbf{\Lambda}^{j-i} \vec{\mu}_i \qquad \text{if } i \in J \wedge i+1, \dots, j \notin J \tag{4.10}$$

We did compression experiments on both `MOCAP` and `CHLORINE` data and evaluated the quality by rel-



(a) MOCAP walking(#22)

(b) CHLORINE

Figure 4.10: Compression with PLiF: normalized reconstruction error versus compression ratio. Note PLiF achieves up to three times better than state-of-the-art method DynaMMo compression.

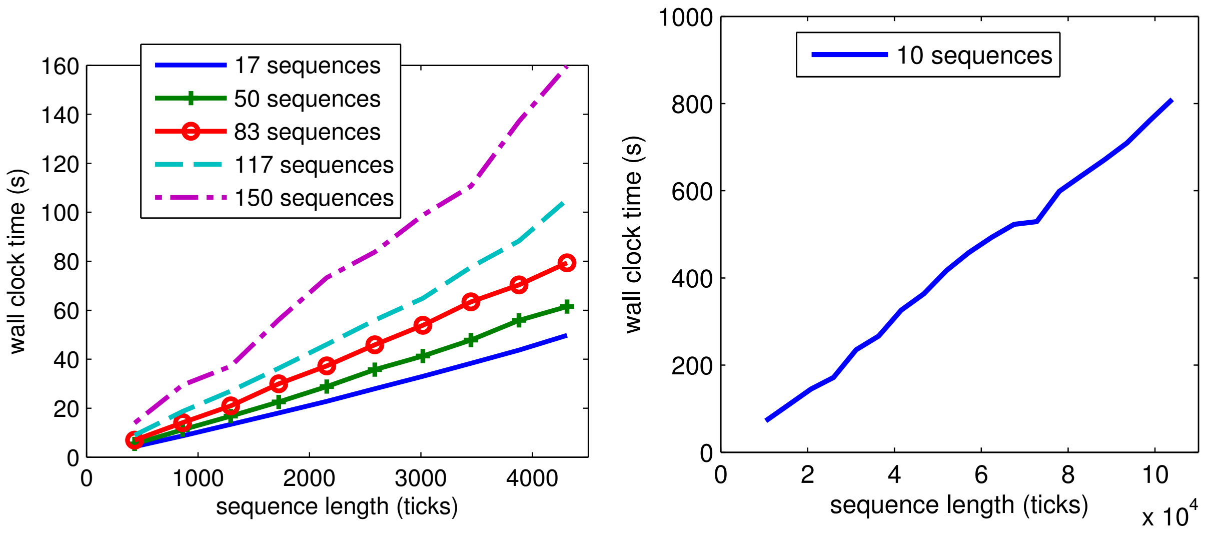
ative error defined as: relative error $= \frac{mse(\hat{\mathbf{X}}-\mathbf{X})\cdot m}{\sum_i var(X_i)}$ where mse denotes mean square error and var variance for each sequence. Fig. 4.10(a) and 4.10(b) show respectively PLiF-compression results for a walking motion (#22) and CHLORINE compared with PCA and DynaMMo [Li et al., 2009]. Note here the statistics are generated by varying over different h and number of time ticks of hidden variables to keep, and we only plot the skyline of compression ratio and error.

4.4.5 Scalability

We now evaluate the scalability of PLiF on both MOCAP and CHLORINE data. We took various sizes of the CHLORINE sequences (by truncation) to test the scalability with respect to the length and the number of sequences.

Fig. 4.11(a) and 4.11(b) show the wall clock time of PLiF with respect to the length of sequences, on five different number of sequences from CHLORINE data, and on 10 sequences from BGP data (after taking the logarithm). In each experiment, we set the number of hidden variables $h = 15$ for CHLORINE and $h = 10$ for BGP and the learning step runs at the same number of iterations (= 20). In Fig. 4.11(a) and 4.11(b), all wall clock times fall in to almost straight line, indicating the linear scalability of PLiF over the length of sequences.

We did experiment on MOCAP dataset to compare the speed of PLiF and PLiF-basic. Fig. 4.12 presents wall clock time on three typical MOCAP sequences, one walking motion (#22), one jumping motion and one running motion (#45). PLiF is 3 times faster PLiF-basic. Experiments on the CHLORINE dataset reveals similarly, PLiF scales much better than the basic algorithm PLiF-basic over the number of sequences and gets up to 3.5 times faster than the latter.



(a) CHLORINE (b) BGP

Figure 4.11: PLiF computation time versus the length of sequences on CHLORINE and BGP datasets: linear as expected.

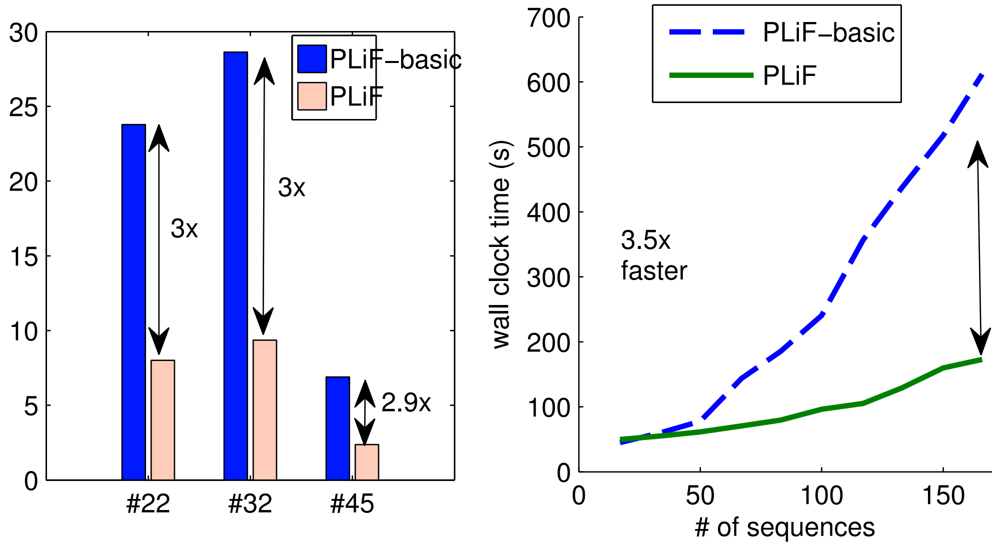


Figure 4.12: Wall clock time of PLiF versus PLiF-basic on and MOCAP: upto 3x gains. Similarly experiment on CHLORINE dataset obtains 3.5x speed up.

4.5 Summary

In this chapter, we present the PLiF algorithm, for the extraction of "*fingerprints*" from a collection of co-evolving time sequences. PLiF has all of the following desirable characteristics,

1. *Effectiveness:* The resulting features correspond to membership weights in each harmonics group; thus, they capture correlations, despite the presence of lags, and despite small shifts in frequency. The resulting distance function agrees with human intuition and the provided ground truth. Thus, fingerprints lead to good clustering, as well as visualization (see Figure 4.8(d)) and the confusion matrix (Table 4.4).
2. *Interpretability*: The fingerprints correspond to groups of harmonics, which are easy to interpret.
3. *Forecasting:* PLiF can easily do forecasting, since it is based on linear dynamical systems and their corresponding difference equations. Thus, it can easily do forecasting and compression, outperforming SVD and state-of-the-art compression methods up to 3 times (see Figure 4.10(a),4.10(b)).
4. *Scalability:* PLiF is fast and scalable, being *linear* on the length of the sequences.

We showed the basic version of PLiF, as well as the final one. Both are linear on the length of sequences, but PLiF can be up to 3.5 times faster, thanks to our Lemma 4.5.

Future work could focus on testing PLiF's performance on additional datasets and its use for segmentation and anomaly detection, which as we mentioned are natural by-products of any method that can do forecasting. One limitation of the current proposed method is the inability to handle sequences of non-uniform lengths. In such cases, the naïve way of truncating sequences wastes part of data. Hence further work may also target extending PLiF to cluster time series with different lengths.

Chapter 5

Complex Linear Dynamical System

Given a motion capture sequence, how to identify the category of the motion? Classifying human motions is a critical task in motion editing and synthesizing, for which manual labeling is clearly inefficient for large databases. The last chapter introduced a method for extracting features from time series, helpful in identifying labels of motion.

Here we study the general model of time series clustering. We propose a novel method of clustering time series that can (a) learn joint temporal dynamics in the data; (b) handle time lags; and (c) produce interpretable features. We achieve this by developing complex-valued linear dynamical systems (CLDS), which include real-valued Kalman filters as a special case; our advantage is that the transition matrix is simpler (just diagonal), and the transmission one easier to interpret. The model looks neater than PLiF while it achieves all benefits of the latter. We then present Complex-Fit, a novel EM algorithm to learn the parameters for the general model and its special case for clustering. Our approach produces significant improvement in clustering quality, 1.5 to 5 times better than well-known competitors on real motion capture sequences.

5.1 Motivation

Motion capture is a useful technology for generating realistic human motions, and is used extensively in computer games, movies and quality of life research[Lee and Shin, 1999, Safonova et al., 2003, Kagami et al., 2003]. With large motion capture database, it is possible to generate new realistic human motion. However, automatically analyzing (e.g. segmentation and labeling) such a large set of motion sequences is a challenging task. This paper is motivated by the application of clustering motion capture sequences (corresponding to different marker positions), an important step towards understanding human motion, but our proposed method is a general one and applies to other time series as well.

Clustering algorithms often rely on features extracted from data. Some most popular ways include using the dynamic time warping (DTW) distance among sequences [Gunopulos and Das, 2001], using Principal Component Analysis (PCA) [Ding and He, 2004] and using Discrete Fourier Transform (DFT) coefficients. But unfortunately, directly applying traditional clustering algorithms to the features may not lead to appealing results. This is largely due to *two* distinct characteristics of time series data, (a) temporal dynamics; and (b) time shifts (lags). Differing from the conventional view of data as points in high dimensional space, time sequences encode temporal dynamics along the time ticks. Such dynamics often imply the grouping of those sequences in many real cases. For example, walking, running, dancing, and jumping motions are characterized by a particular movement of human body, which results in different dynamics among the sequences. Hence by identifying the evolving temporal components, we can find the clusters of sequences by grouping those with similar temporal pattern. As mentioned above, another often overlooked characteristic are time shifts. For example, two walking motions may start from different footsteps, resulting in a lag among the sequences. Traditional methods like PCA and k-means can not handle such lags in sequences, yielding poor clustering results. On the other hand, DTW, while handling lags, misses joint dynamics - thus sequences having the same underlying process but slightly different parameters (e.g. walking veering left vs. walking veering right) will have large DTW distances.

Hence we want the following main properties in any clustering algorithm for time series:

P1 It should be able to identify joint dynamics across the sequences
P2 It should be able to eliminate lags (time shifts) across sequences
P3 The features generated should be interpretable

Our method is designed to automatically identify both the joint dynamics in the multiple sequences, lag correlations. As we show later, our proposed method achieves all of the above characteristics, while other traditional methods miss out on one or more of these.

On one hand, our model directly extends the capability of Linear Dynamical Systems (LDS) in identifying dynamics in the data. Hidden Markov Models (HMM) and LDS are commonly used tools for time series analysis, partly due to their simple implementation and extensibility. LDS can learn a set of hidden variables and the evolving dynamics among them. In addition, it gives us the so-called "output" matrix mapping hidden variables to observations. This output matrix can be viewed as co-ordinates of each sequence in the space spanned by hidden variables (and thus dynamics) and therefore can be used as features in clustering. However, such an approach can not handle time shift common in many cases, often leading to a poor clustering (as we already see in previous chapter).

On the other, our method also includes DFT's capability of generating co-efficients (features) invariant to time shifts. To see this, the DFT identifies the weight (energy) of the signal over a base spectrum of frequencies. Since the basis depends only on the frequencies, signals with a fixed time lag will have the same energy spectrum. Hence, clustering based on such Fourier co-efficients can tolerate lags in sequences and the features are interpretable as well. But because it has a fixed set of basis functions, and it lacks notion of joint dynamics, it can not find arbitrary and near-by frequencies. Our method is inspired by the way of handling time shift in Fourier analysis. In our method, we define hidden variables in similar frequency sense.

Our proposed method is intended to achieve the merits of both these approaches. The basic intuition is to encode Fourier-like frequencies as hidden variables and then model the dynamics over the hidden variables like LDS. However, in contrast to the fixed basis functions of Fourier analysis, our method uses a generative model for modeling frequencies and weights.

The main idea is to use *complex-valued* linear dynamical system, which leads to several advantages: we can afford to have a diagonal transition matrix, which is simpler and faster to estimate; the resulting hidden variables are easy to interpret; and we meet all the design goals, including lag-invariance.

Specifically, the contributions of this paper are:

1. *Design of CLDS* : We develop complex-valued linear dynamical systems (CLDS), which includes

Table 5.1: Symbols and notations

$\mathbb{C}$	field of complex numbers
$\mathbf{X}$	observation data, $= \vec{x}_1 \ldots \vec{x}_{\mathrm{T}}$
$(\cdot)^*$	conjugate transpose, $= \overline{(\cdot)}^T$
$\mathbf{A} \circ \mathbf{B}$	Hadamard product, $(\mathbf{A} \circ \mathbf{B})_{i,j} = \mathbf{A}_{i,j} \cdot \mathbf{B}_{i,j}$

traditional real-valued Kalman Filters as special cases. We then provide a novel complex valued EM algorithm, Complex-Fit, to learn the model parameters from the data.

2. *Application to Clustering*: We also use a special formulation of CLDS for time series clustering by imposing a restricted form of the transition dynamics corresponding to frequencies, without losing any expressiveness. Such an approach enhances the interpretability as well. Our clustering method then uses the participation weight (energy) of the hidden variables as features, thus eliminating lags. Hence it satisfies P1, P2 and P3 mentioned before.
3. *Validation*: Finally, we evaluate our algorithm on both synthetic data and real motion capture data. Our proposed method is able to achieve best clustering results, comparing against several other popular time series clustering methods.

In addition, our proposed CLDS includes as special cases several popular, powerful methods like PCA, DFT and AR.

In the following sections, we will first present several pieces of the related models and techniques in time series clustering. We will also briefly introduce complex normal distributions and a few useful properties, and then present our CLDS and its learning algorithm, along with its application in time series clustering.

5.2 Complex Linear Gaussian Distribution

As the traditional linear dynamical systems heavily rely on multivariate normal distribution, our proposed model is build upon a basic distribution of complex variables and their vectors, complex (valued) normal distributions. This section introduces the basic notations of complex valued normal distribution and its related extension, linear Gaussian distributions (or linear normal distributions), which are building blocks of our proposed method. We will give a concise summary of the joint, the marginal and the posterior distributions as well. For a full description of the normal distributions of complex variables, we refer readers to [Goodman, 1963, Andersen et al., 1995].

Definition 5.1 (One complex random variable)**.** *Let u and v be real random variables, x is a complex random variable where $x = u + iv$.*

Definition 5.2 (Multivariate Complex Normal Distribution)**.** *Let $\vec{x}$ be a vector of complex random variables, with dimensionality of m. $\vec{x}$ follows a multivariate complex normal distribution, denoted as $\vec{x} \sim \mathcal{CN}(\mu, H)$, if its p.d.f is*

$$p(\vec{x}) = \pi^{-m}|H|^{-1}\exp(-(\vec{x}-\mu)^* H^{-1}(\vec{x}-\mu)) \tag{5.1}$$

where H is a positive semi-definite and Hermitian[1] *matrix[Andersen et al., 1995]. The mean and variance are given by $\mathbb{E}[\vec{x}] = \mu$ and $Var(\vec{x}) = H$.*

[1] $(\cdot)^*$ is the Hermitian operator, $(X)^* = (\bar{X})^T$.

Example 5.1. *A standard complex normal distribution $\mathcal{CN}(0,1)$ has the following p.d.f*

$$p(x) = \frac{1}{\pi} e^{-|x|^2}$$

Figure 5.1 shows in complex plane random samples drawn from the standard complex distribution.

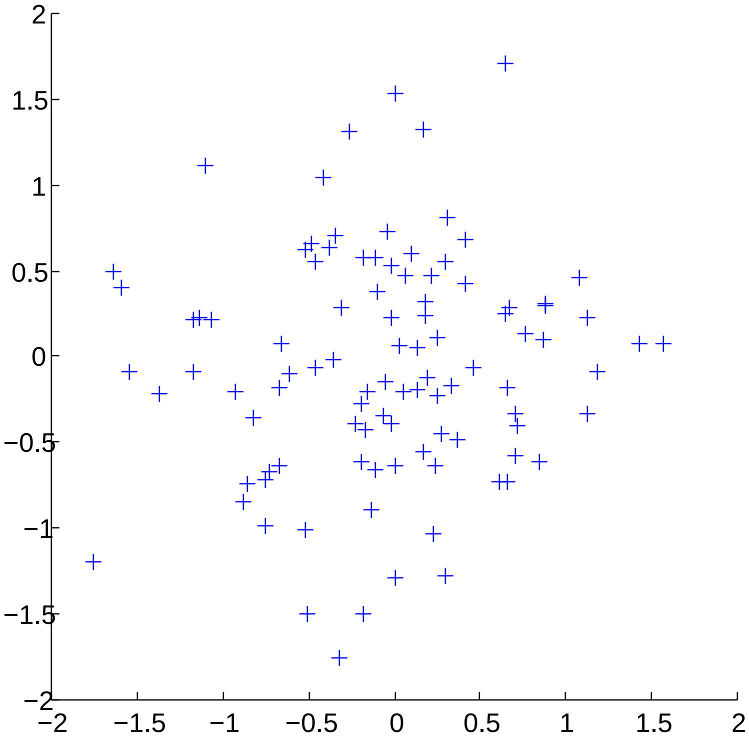


Figure 5.1: 100 samples drawn from the standard complex normal distribution $\mathcal{CN}(0,1)$. The samples are plotted in the complex plane.

The linear Gaussian distribution is a common relation in practice, i.e. the random variable $\vec{y}$ is conditionally dependent on $\vec{x}$ with respect to linear transformation of variables plus Gaussian noises. The following lemmas state such relation with two variables, and it can be recursively extended to multiple variables. All the following lemmas are heavily used in our derivation to obtain the EM algorithm for CLDS.

Lemma 5.1 (Linear Gaussian distributions)**.** *If $\vec{x}$ and $\vec{y}$ random vectors from the following distribution,*

$$\vec{x} \sim \mathcal{CN}(\vec{\mu}, \mathbf{H})$$
$$\vec{y}|\vec{x} \sim \mathcal{CN}(\mathbf{A} \cdot \vec{b}, \mathbf{V})$$

Then $\vec{z} = \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix}$ will follow a complex normal distribution with the following mean and covariance structure.

$$\mathbb{E}\begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} = \begin{pmatrix} \vec{\mu} \\ \mathbf{A} \cdot \vec{\mu} + \vec{b} \end{pmatrix}$$
$$\mathrm{Var}\begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} = \begin{pmatrix} \mathbf{H} & \mathbf{H} \cdot \mathbf{A}^* \\ \mathbf{A} \cdot \mathbf{H} & \mathbf{V} + \mathbf{A} \cdot \mathbf{H} \cdot \mathbf{A}^* \end{pmatrix}$$

Lemma 5.2 (Marginal distribution)**.** *Under the same assumption as Lemma 5.1, it follows that*

$$\vec{y} \sim \mathcal{CN}(\mathbf{A} \cdot \vec{\mu} + \vec{b}, \mathbf{V} + \mathbf{A} \cdot \mathbf{H} \cdot \mathbf{A}^*)$$

Lemma 5.3 (Posterior distribution)**.** *Under the same assumption as Lemma 5.1, the posterior distribution of $\vec{y}|\vec{x}$ is complex normal, and its mean $\vec{\mu}_{\vec{y}|\vec{x}}$ and covariance matrix $\mathbf{\Sigma}_{\vec{y}|\vec{x}}$ given by,*

$$\begin{aligned}\vec{\mu}_{\vec{y}|\vec{x}} &= \vec{\mu} + \mathbf{K} \cdot (\vec{y} - \vec{b} - \mathbf{A} \cdot \vec{\mu}) \\ \mathbf{\Sigma}_{\vec{y}|\vec{x}} &= (I - \mathbf{K} \cdot \mathbf{A}) \cdot \mathbf{H}\end{aligned}$$

where the "gain" matrix $\mathbf{K} = \mathbf{H} \cdot \mathbf{A}^* \cdot (\mathbf{V} + \mathbf{A} \cdot \mathbf{H} \cdot \mathbf{A}^*)^{-1}$.

A nice property of complex linear Gaussian distribution is "rotation invariance". In the simplest form, the marginal will remain the same for a family of linear transformation, i.e. $y = ax \sim P(0, |a|^2)$ iff $x \sim \mathcal{CN}(0, 1)$. In this case, ax and $|a|x$ have the same distribution.

5.3 CLDS and its learning algorithm

In this section we describe the formulation of complex-valued linear dynamical systems and its special case for clustering.

The complex linear dynamical systems (CLDS) is defined with the following equations.

$$\vec{z}_1 = \vec{\mu}_0 + \vec{w}_1 \tag{5.2}$$

$$\vec{z}_{n+1} = \mathbf{A} \cdot \vec{z}_n + \vec{w}_{n+1} \tag{5.3}$$

$$\vec{x}_n = \mathbf{C} \cdot \vec{z}_n + \vec{v}_n \tag{5.4}$$

where the noise vectors follow complex normal distribution. $\vec{w}_1 \sim \mathcal{CN}(0, \mathbf{Q}_0)$, $\vec{w}_i \sim \mathcal{CN}(0, \mathbf{Q})$, and $\vec{v}_j \sim \mathcal{CN}(0, \mathbf{R})$. Note unlike Kalman filters, CLDS allows complex values in the parameters, with the restriction that $\mathbf{Q}_0$, $\mathbf{Q}$ and $\mathbf{R}$ should be Hermitian and positive definite. Figure 5.2 shows the graphical model. It can be viewed as consecutive linear Gaussian distributions on the hidden variable $\vec{z}$'s and observation $\vec{x}$.

Example 5.2. *As an example, consider the synthetic sequences defined in Table 4.2 of Chapter 4. For readability, we rewrite the equations here. Let $X = (X_1, X_2, X_3, X_4, X_5)^T$ where each of the sequence is defined in the following equations:*

$$\begin{aligned}X_1 &= \sin(\frac{2\pi t}{100}) \\ X_2 &= \cos(\frac{2\pi t}{100}) \\ X_3 &= \sin(\frac{2\pi t}{100} + \frac{\pi}{6}) \\ X_4 &= \sin(\frac{2\pi t}{110}) + 0.2\sin(\frac{2\pi t}{30}) \\ X_5 &= \cos(\frac{2\pi t}{110}) + 0.2\sin(\frac{2\pi t}{30} + \frac{\pi}{4})\end{aligned}$$

In particular, X_3 differs with X_1 and X_2 slightly in the time shift. X_1 and X_2 are time shifted from each other. X_1, X_2, X_3 are generated (or approximately) by a frequency of $\frac{1}{100}$, while X_4 and X_5 are generated by another set of frequencies $\frac{1}{110}$ and $\frac{1}{30}$.

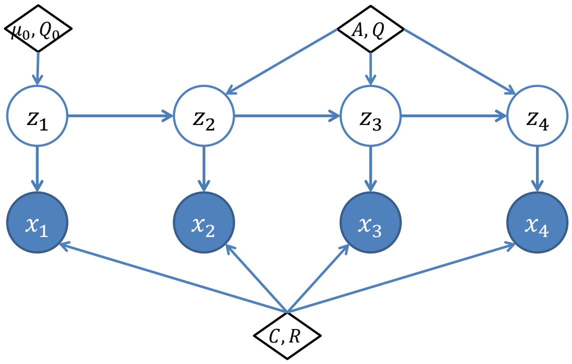


Figure 5.2: Graphical Model for CLDS. $\vec{x}$ are real valued observations and $\vec{z}$ are complex hidden variables. Arrows denote linear Gaussian distributions. Note all the parameters and random variables are in complex domain (vectors of matrices of complex values).

A CLDS for such sequences would be:

$$\vec{\mu}_0 = \begin{pmatrix} 0.9984 + 0.0571i \\ 0.9984 - 0.0571i \\ 0.9980 + 0.0628i \\ 0.9980 - 0.0628i \\ 0.9781 + 0.2079i \\ 0.9781 - 0.2079i \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} 0.9984 + 0.0571i & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.9984 - 0.0571i & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9980 + 0.0628i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9980 - 0.0628i & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9781 + 0.2079i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9781 - 0.2079i \end{pmatrix}$$

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & -0.5i & 0.5i & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.25 - 0.4330i & 0.25 + 0.4330i & 0 & 0 \\ -0.5i & 0.5i & 0 & 0 & -0.1i & 0.1i \\ 0.5 & 0.5 & 0 & 0 & 0.0707 - 0.0707i & 0.0707 + 0.0707i \end{pmatrix}$$

with the covariance matrices be identity matrices.

The problem of learning is to estimate the best fit parameters $\theta = \{\mu_0, \mathbf{Q}_0, \mathbf{A}, \mathbf{Q}, \mathbf{C}, \mathbf{R}\}$, giving the observation sequence $\vec{x}_1 \ldots \vec{x}_{\mathrm{T}}$. We develop Complex-Fit, a novel complex valued expectation-maximization algorithm towards a maximum likelihood fitting.

The expected negative-loglikelihood of the model is

$$\begin{aligned}\mathcal{L}(\theta) =&\mathbb{E}_{\mathbf{Z}|\mathbf{X}}[-\log P(\mathbf{X},\mathbf{Z}|\theta)]\\ =&\mathbb{E}[(\vec{z}_1-\vec{\mu}_0)^* \cdot \mathbf{Q}_0^{-1} \cdot (\vec{z}_1-\vec{\mu}_0)] + \mathbb{E}[\sum_{n=1}^{\mathrm{T}-1}(\vec{z}-\mathbf{A}\cdot\vec{z}_n)^* \cdot Q^{-1} \cdot (\vec{z}_{n+1}-\mathbf{A}\cdot\vec{z}_n)]\\ &+\mathbb{E}[\sum_{n=1}^{\mathrm{T}}(\vec{x}_n-\mathbf{C}\cdot\vec{z}_n)^* \cdot R^{-1} \cdot (\vec{x}_n-\mathbf{C}\cdot\vec{z}_n)] + \log|\mathbf{Q}_0| + (\mathrm{T}-1)\log|Q| + \mathrm{T}\log|R| \quad (5.5)\end{aligned}$$

where the expectation $\mathbb{E}[\cdot]$ is over the posterior distribution of $\mathbf{Z}$ conditioned on $\mathbf{X}$.

Unlike traditional Kalman filters, the objective here is a function of complex values, requiring nonstandard optimization in complex domain. We will first describe the M-step here. In the negative-loglikelihood, there were two sets of unknowns, the parameters and the posterior distribution. The overall idea of the Complex-Fit algorithm is to optimize over the parameter set θ as if we know the posterior and then estimate the posterior with current parameters. It then takes turns to obtain the optimal solution.

Complex-Fit M-step The M-step is derived by taking complex derivatives of the objective function and equating them to zeros. Unlike the real valued version, taking derivatives of complex functions should take extra care, since they are not always analytic or holomorphic. The above objective function Eq. (5.5) is not differentiable in classical setting since it does not satisfy Cauchy-Riemann condition[Mathews and Howell, 2006]. However, if x and $\bar{x}$ are treated independently, we could obtain their generalized partial derivatives, as defined in Definition 5.3 ([Brandwood, 1983, Hjorungnes and Gesbert, 2007]). The optimal of the function $f(x)$ can be achieved when both partial derivatives of $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial \bar{x}}$ equal zero.
Definition 5.3 (Generalized Complex Partial Derivative)**.**

$$\frac{\partial x}{x} = 1 \qquad \textit{and} \qquad \frac{\partial \bar{x}}{x} = 0$$

The solution minimizing $\mathcal{L}$ is given by

$$\frac{\partial}{\partial \vec{\mu}_0}\mathcal{L} = 0 \qquad \frac{\partial}{\partial \overline{\vec{\mu}_0}}\mathcal{L} = 0 \tag{5.6}$$

$$\frac{\partial}{\partial \mathbf{Q}_0}\mathcal{L} = 0 \qquad \frac{\partial}{\partial \overline{\mathbf{Q}_0}}\mathcal{L} = 0 \tag{5.7}$$

where

$$\frac{\partial}{\partial \vec{\mu}_0}\mathcal{L} = -\,(\mathbb{E}[z_1]-\vec{\mu}_0)^* \cdot \mathbf{Q}_0^{-1} \tag{5.8}$$

$$\frac{\partial}{\partial \overline{\vec{\mu}_0}}\mathcal{L} = -\,(\mathbb{E}[z_1]-\vec{\mu}_0)^T \cdot (\mathbf{Q}_0^{-1})^T \tag{5.9}$$

$$\frac{\partial}{\partial \mathbf{Q}_0}\mathcal{L} =(\mathbf{Q}_0^T)^{-1} - (\mathbf{Q}_0^T)^{-1} \cdot \mathbb{E}[\overline{(\vec{z}_1-\vec{\mu}_0)} \cdot (\vec{z}_1-\vec{\mu}_0)^T] \cdot (\mathbf{Q}_0^T)^{-1} \tag{5.10}$$

It follows that:

$$\vec{\mu}_0 =\mathbb{E}[\vec{z}_1] \tag{5.11}$$

$$\mathbf{Q}_0 =\mathbb{E}[\vec{z}_1 \cdot \vec{z}_1^*] - \vec{\mu}_0 \cdot \vec{\mu}_0^* \tag{5.12}$$

Similarly, we obtain update rules for $\mathbf{A}$, $\mathbf{Q}$, $\mathbf{C}$ and $\mathbf{R}$, by taking partial derivatives, $\frac{\partial}{\partial \mathbf{A}}\mathcal{L}$, $\frac{\partial}{\partial \overline{\mathbf{A}}}\mathcal{L}$, $\frac{\partial}{\partial \mathbf{Q}}\mathcal{L}$, $\frac{\partial}{\partial \overline{\mathbf{Q}}}\mathcal{L}$, $\frac{\partial}{\partial \mathbf{C}}\mathcal{L}$, $\frac{\partial}{\partial \overline{\mathbf{C}}}\mathcal{L}$, $\frac{\partial}{\partial \mathbf{R}}\mathcal{L}$, and $\frac{\partial}{\partial \overline{\mathbf{R}}}\mathcal{L}$, and equating them to zeros. Here is the final update rules for each of these parameters. The details are included in Appendix 5.A.1.

$$\mathbf{A} = \left(\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[\vec{z}_{n+1} \cdot \vec{z}_n^*]\right) \cdot \left(\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*]\right)^{-1} \tag{5.13}$$

$$\mathbf{Q} = \frac{1}{\mathrm{T}-1} \sum_{n=1}^{\mathrm{T}-1} \left(\mathbb{E}[\vec{z}_{n+1} \cdot \vec{z}_{n+1}^*] - \mathbb{E}[\vec{z}_{n+1} \cdot \vec{z}_n^*] \cdot \mathbf{A}^* - \mathbf{A} \cdot \mathbb{E}[\vec{z}_n \cdot \vec{z}_{n+1}^*] + \mathbf{A} \cdot \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*] \cdot \mathbf{A}^*\right) \tag{5.14}$$

$$\mathbf{C} = \left(\sum_{n=1}^{\mathrm{T}} \vec{x}_n \cdot \mathbb{E}[\vec{z}_n^*]\right) \cdot \left(\sum_{n=1}^{\mathrm{T}} \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*]\right)^{-1} \tag{5.15}$$

$$\mathbf{R} = \frac{1}{\mathrm{T}} \sum_{n=1}^{\mathrm{T}} \left(\vec{x}_n \cdot \vec{x}_n^* - \vec{x}_n \cdot \mathbb{E}[\vec{z}_n^*] \cdot \mathbf{C}^* - \mathbf{C} \cdot \mathbb{E}[\vec{z}_n] \cdot \vec{x}_n^* + \mathbf{C} \cdot \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*] \cdot \mathbf{C}^*\right) \tag{5.16}$$

Complex-Fit E-step The above M-step requires computation of the sufficient statistics on the posterior distribution of hidden variables $\vec{z}$. During the E-step, we will compute mean and covariance of the marginal and joint posterior distributions $P(\vec{z}_n|\mathbf{X})$ and $P(\vec{z}_n, \vec{z}_{n+1}|\mathbf{X})$. The E-step computes the posteriors in with the forward-backward sub steps (corresponding to Kalman filtering and smoothing in the traditional LDS). The forward step computes the partial posterior $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n$, and the backward pass computes the full posterior distributions. We can show by induction that all these posteriors are complex normal distributions and the transition between them satisfying the condition of linear Gaussian distribution. Such facts will help us derive an algorithm to find the means and variances of those posterior distributions.

The forward step computes the partial posterior $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n$ from the beginning $\vec{z}_1$ to the tail of the chain $\vec{z}_\mathrm{T}$. By exploiting Markov properties and applying Lemma 5.1, Lemma 5.2 and Lemma 5.3 on posteriors $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n$, we can show that $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n \sim \mathcal{CN}(\vec{u}_n, \mathbf{U}_n)$, with following equations for computing $\vec{u}_n$ and $\mathbf{U}_n$ recursively (see proof and derivation in Appendix 5.A.2),

$$\vec{u}_{n+1} = \mathbf{A} \cdot \vec{u}_n + \mathbf{K}_{n+1} \cdot (\vec{x}_{n+1} - \mathbf{C} \cdot \mathbf{A} \cdot \vec{u}_n) \tag{5.17}$$

$$\mathbf{U}_{n+1} = (I - \mathbf{K}_{n+1} \cdot \mathbf{C}) \cdot \mathbf{P}_{n+1} \tag{5.18}$$

and we define,

$$\mathbf{P}_{n+1} = \mathbf{A} \cdot \mathbf{U}_n \cdot \mathbf{A}^* + \mathbf{Q} \tag{5.19}$$

$$\mathbf{K}_{n+1} = \mathbf{P}_{n+1} \cdot \mathbf{C}^* \cdot (\mathbf{R} + \mathbf{C} \cdot \mathbf{P}_{n+1} \cdot \mathbf{C}^*)^{-1} \tag{5.20}$$

The initial step is given by

$$\vec{u}_1 = \vec{\mu}_0 + \mathbf{K}_1 \cdot (\vec{x}_1 - \mathbf{C} \cdot \vec{\mu}_0) \tag{5.21}$$

$$\mathbf{U}_1 = (I - \mathbf{K}_1 \cdot \mathbf{C}) \cdot \mathbf{Q}_0 \tag{5.22}$$

and $\mathbf{K}_1$ is the complex-valued "Kalman gain" matrix,

$$\mathbf{K}_1 = \mathbf{Q}_0 \cdot \mathbf{C}^* \cdot (\mathbf{R} + \mathbf{C} \cdot \mathbf{Q}_0 \cdot \mathbf{C}^*)^{-1}$$

The backward step computes the posterior $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_\mathrm{T}$ from the tail $\vec{z}_\mathrm{T}$ to the head of the chain $\vec{z}_1$. Again using the lemmas of complex linear Gaussian distributions, we can show $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_\mathrm{T} \sim \mathcal{CN}(\vec{v}_n, \mathbf{V}_n)$, and compute the posterior means and variances through the following equations (see proof and derivation in Appendix 5.A.2).

$$\vec{v}_n = \vec{u}_n + \vec{J}_{n+1} \cdot (\vec{v}_{n+1} - \mathbf{A} \cdot \vec{u}_n) \tag{5.23}$$

$$\mathbf{V}_n = \mathbf{U}_n + \mathbf{J}_{n+1} \cdot (\mathbf{V}_{n+1} - \mathbf{P}_{n+1}) \cdot \mathbf{J}^*_{n+1} \tag{5.24}$$

where

$$\mathbf{J}_{n+1} = \mathbf{U}_n \cdot \mathbf{A}^* \cdot (\mathbf{A} \cdot \mathbf{U}_n \cdot \mathbf{A}^* + \mathbf{Q})^{-1} = \mathbf{U}_n \cdot \mathbf{A}^* \cdot \mathbf{P}^{-1}_{n+1}$$

Obviously, $\vec{v}_\mathrm{T} = \vec{u}_\mathrm{T}$ and $\mathbf{V}_\mathrm{T} = \mathbf{U}_\mathrm{T}$.

With a similar induction, from Lemma 5.1 we can compute the following sufficient statistics,

$$\mathbb{E}[\vec{z}_n \cdot \vec{z}^*_n] = \mathbf{V}_n + \vec{v}_n \cdot \vec{v}^*_n \tag{5.25}$$

$$\mathbb{E}[\vec{z}_n \cdot \vec{z}^*_{n+1}] = \mathbf{J}_{n+1} \cdot \mathbf{V}_{n+1} + \vec{v}_n \cdot \vec{v}^*_{n+1} \tag{5.26}$$

5.4 Clustering with CLDS

In addition to the full model as described above, we consider a special case with diagonal transition transition matrix $\mathbf{A}$. The diagonal elements of $\mathbf{A}$ correspond to its eigenvalues, denoted as $\vec{a}$. The eigenvalues of the matrix will be similar to the frequencies in Fourier analysis. The justification of using diagonal matrix lies in the observation of the rotation invariance property in linear Gaussian distributions (Lemma 5.4). In simplest case, such rotation invariant matrix is diagonal.

Lemma 5.4 (Rotation invariance)**.** *Assume $\vec{x} \sim \mathcal{CN}(0, \mathbf{I})$ and $\mathbf{B} = \mathbf{A} \cdot \mathbf{V}$ with unitary matrix [2] $\mathbf{V}$, it follows that $\mathbf{A} \cdot \vec{x}$ and $\mathbf{B}\vec{x}$ have exactly the same distribution. By abusing the definition of $\sim$ slightly, it can be written as*

$$\mathbf{A} \cdot \vec{x} \sim \mathbf{B}\vec{x} \sim \mathcal{CN}(0, \mathbf{A} \cdot \mathbf{A}^*)$$

To get the optimal solution in Eq. (5.5) with such a diagonal $\mathbf{A}$, we will use the definition of Hadamard product[3] and its related results. Let $\vec{a}$ to be the diagonal elements of $\mathbf{A}$. Since $\mathbf{A}$ is diagonal, the difference will result in a rather different solution in partial derivatives. The condition of optimal solutions will be given by

$$\frac{\partial \mathcal{L}}{\partial \vec{a}} = \sum_{n=1}^{N-1} \mathbb{E}[(\mathbf{Q}^{-1} \cdot (\vec{z}_{n+1} - \vec{a} \circ \vec{z}_n)) \circ \overline{\vec{z}_n}]^* = 0 \tag{5.27}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Q}} = (\mathrm{T} - 1)(\mathbf{Q}^T)^{-1} - (\mathbf{Q}^T)^{-1} \cdot \left(\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[(\overline{\vec{z}_{n+1}} - \overline{\vec{a}} \circ \overline{\vec{z}_n}) \cdot (\vec{z}_{n+1} - \vec{a} \circ \vec{z}_n)^T] \right) \cdot (\mathbf{Q}^T)^{-1} = 0 \tag{5.28}$$

[2] A matrix $\mathbf{V}$ is unitary if $\mathbf{V} \cdot \mathbf{V}^* = \mathbf{V}^* \cdot \mathbf{V} = \mathbf{I}$

[3] $(\mathbf{A} \circ \mathbf{B})_{i,j} = \mathbf{A}_{i,j} \cdot \mathbf{B}_{i,j}$

Algorithm 5.1: CLDS Clustering

```
Input: data sequences: $\mathcal{X}$, containing $m$ sequences
number of clusters $k$
Output: features $\mathbf{F}$ and class labels $\mathbf{G}$
$\langle \vec{\mu}_0, \mathbf{Q}_0, \mathbf{A}, \mathbf{Q}, \mathbf{C}, \mathbf{R} \rangle \longleftarrow \arg\min \mathcal{L}(\theta)$ // learn a diagonal CLDS model to
    optimize Eq. (5.5)
$\mathbf{C}_m \longleftarrow \text{abs}(\mathbf{C})$ ;
$\mathbf{F} \longleftarrow \text{PCA}(\mathbf{C}_m)$ ;
$\mathbf{G} \longleftarrow \text{kmeans}(\mathbf{F}, k)$;
```

To solve (5.27) and (5.28), we use the following iterative update rules.

$$\vec{a} = \left(\mathbf{Q}^{-1} \circ \left(\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*] \right)^T \right)^{-1} \cdot \left(\mathbf{Q}^{-1} \circ \left(\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[\vec{z}_{n+1} \cdot \vec{z}_n^*] \right)^T \right) \cdot \mathbf{1} \tag{5.29}$$

$$\mathbf{Q} = \frac{1}{\mathrm{T}-1} \sum_{n=1}^{\mathrm{T}-1} \Big(\mathbb{E}[\vec{z}_{n+1} \cdot \vec{z}_{n+1}^*] - \mathbb{E}[\vec{z}_{n+1} \cdot (\vec{a} \circ \vec{z}_n)^*] - \mathbb{E}[(\vec{a} \circ \vec{z}_n) \cdot \vec{z}_{n+1}^*] + \mathbb{E}[(\mathbf{a} \circ \vec{z}_n) \cdot (\mathbf{a} \circ \vec{z}_n)^*] \Big) \tag{5.30}$$

Once we have the best estimate of such parameters using Complex-Fit (with diagonal transition matrix), the overall idea of CLDS clustering is essentially using the output matrix in CLDS as features, and then applying any off-the-shelf clustering algorithm (e.g. k-means clustering). In more detail, we take only the magnitude of **C** to eliminate the lags in the data, since its magnitude represents the energy or weight of participation of the *learned* hidden variables in the observations. In this sense, our method can also be used as a feature extraction tool in other applications such as signal compression. Optionally, we can take one more step of PCA or SVD to further reduce the dimensionality of the features. A final step is to cluster the extracted features using typical clustering algorithms like k-means. The full algorithm is described in Algorithm 5.1.

5.4.1 Experiments and Evaluation

We used two datasets (MOCAPPOS and MOCAPANG) from a public human motion capture database[4]. MOCAPPOS includes 49 motion sequences of marker positions in body local coordinates, each motion is labeled with either walking or running as annotated in the database. On the other hand, MOCAPANG includes 33 sequences of joint angles, 10 being walking motions and the rest running. While the original motion sequences have different lengths, we trim them with equal duration. Since there are multiple markers used in the motion capture, we only choose the one (e.g. right foot marker z-coordinate in MOCAPPOS and) that is most significant in telling human motions apart, suggested by domain experts. Alternatively, this can also be achieved through an additional feature selection process, which is beyond the focus of our work.

We compare our method against several baselines:

[4] http://mocap.cs.cmu.edu/ subject #16 and #35

Table 5.2: Conditional entropies (S) of clustering methods on both datasets, calculated using Eq. (5.31) against the confusion matrices in Table 5.3 and Table 5.4. Note a lower score corresponds to a better clustering, and in both cases our proposed method CLDS achieves the lowest scores 1.5 to 5 times better than others, yielding clusters most close to the true labels.

methods	MOCAPPOS S	MOCAPANG S
CLDS	**0.2169**	**0.1246**
PCA	0.6828	0.4205
DFT	0.5420	0.3220
DTW	0.5111	0.4985
KF	0.6233	0.4215

PCA: As we mentioned in background, Principal component analysis is a textbook method to extract features from high dimensional data[5]. In this method, we follow a standard pipeline of clustering high dimensional data [Ding and He, 2004]: first performing a dimensionality reduction on the data matrix by keeping k (=2) principal components, and then clustering on the PCA scores using k-means.

DFT: The second baseline is the Fourier method. This method first computes Fourier coefficients for each motion sequences using Discrete Fourier Transform. It then uses PCA to extract two features from the Fourier coefficients, and finally finds clusters again through k-means clustering on top of the DFT-PCA features.

DTW: The third method, dynamic time warping (DTW), is a popular method to calculate the minimal distance between pairs of sequences by allowing flexible shift in alignment (thus it would be fair competitor on time series with time lags). In this method, we compute all pairwise DTW distances and again use the k-means on top of them to find clusters.

KF: Another baseline method is learning a Kalman filter or linear dynamical systems (LDS) from the data and using its output matrix as features in k-mean clustering. In this experiment, we tried a few values for the number of hidden variables and chose the one with best clustering performance (=8).

To evaluate the quality, we use the conditional entropy S of the true labeling with respect to the prediction, defined by the confusion matrix M. The element $M_{i,j}$ corresponds to the number of sequences with true label i in predicted cluster j.

$$S(M) = \sum_{i,j} \frac{M_{i,j}}{\sum_{k,l} M_{k,l}} \log \frac{\sum_k M_{i,k}}{M_{i,j}} \tag{5.31}$$

Intuitively, it tells difference between the prediction and the actual, therefore a lower score indicates a better prediction and the best case is $S = 0$. In information theory, the conditional entropy corresponds to the additional information of the actual labels based on the prediction.

Table 5.2 lists the conditional entropies of each method on the task of clustering MOCAPPOS and MOCAPANG datasets. For each dataset, we generate features (2D) using each candidate method, and then apply k-means clustering (k=2) on the extracted features with the results from the minimal distance of 10 repeats. Note that our method CLDS achieves the best performance with the lowest entropy. It is also confirmed in the scatter plot of top two features using CLDS (Figure 5.4).

[5] Note: the dimensionality in PCA corresponds to the duration in time series. The dimensionality in time series usually refers to the number of sequences.

Table 5.3: Confusion matrices from clustering results on MOCAPPOS. The element (i, j) corresponds to the number of sequences with true label i in cluster j. Note in perfect case the confusion matrix should be diagonal.

(a) CLDS

predicts	C0	C1
walking	26	0
running	4	19

(b) PCA

predicts	C0	C1
walking	15	11
running	13	10

(c) DFT

predicts	C0	C1
walking	22	4
running	9	14

(d) DTW

predicts	C0	C1
walking	23	3
running	10	13

(e) KF

predicts	C0	C1
walking	19	7
running	9	14

Table 5.4: Confusion matrices from clustering results on MOCAPANG. The element (i, j) corresponds to the number of sequences with true label i in cluster j. Note in perfect case the confusion matrix should be diagonal.

(a) CLDS

predicts	C0	C1
walking	22	1
running	0	10

(b) PCA

predicts	C0	C1
walking	19	4
running	1	9

(c) DFT

predicts	C0	C1
walking	19	4
running	0	10

(d) DTW

predicts	C0	C1
walking	17	6
running	1	9

(e) KF

predicts	C0	C1
walking	20	3
running	2	8

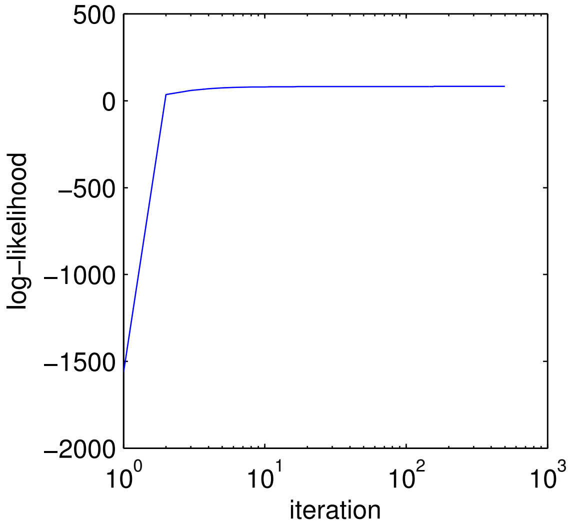
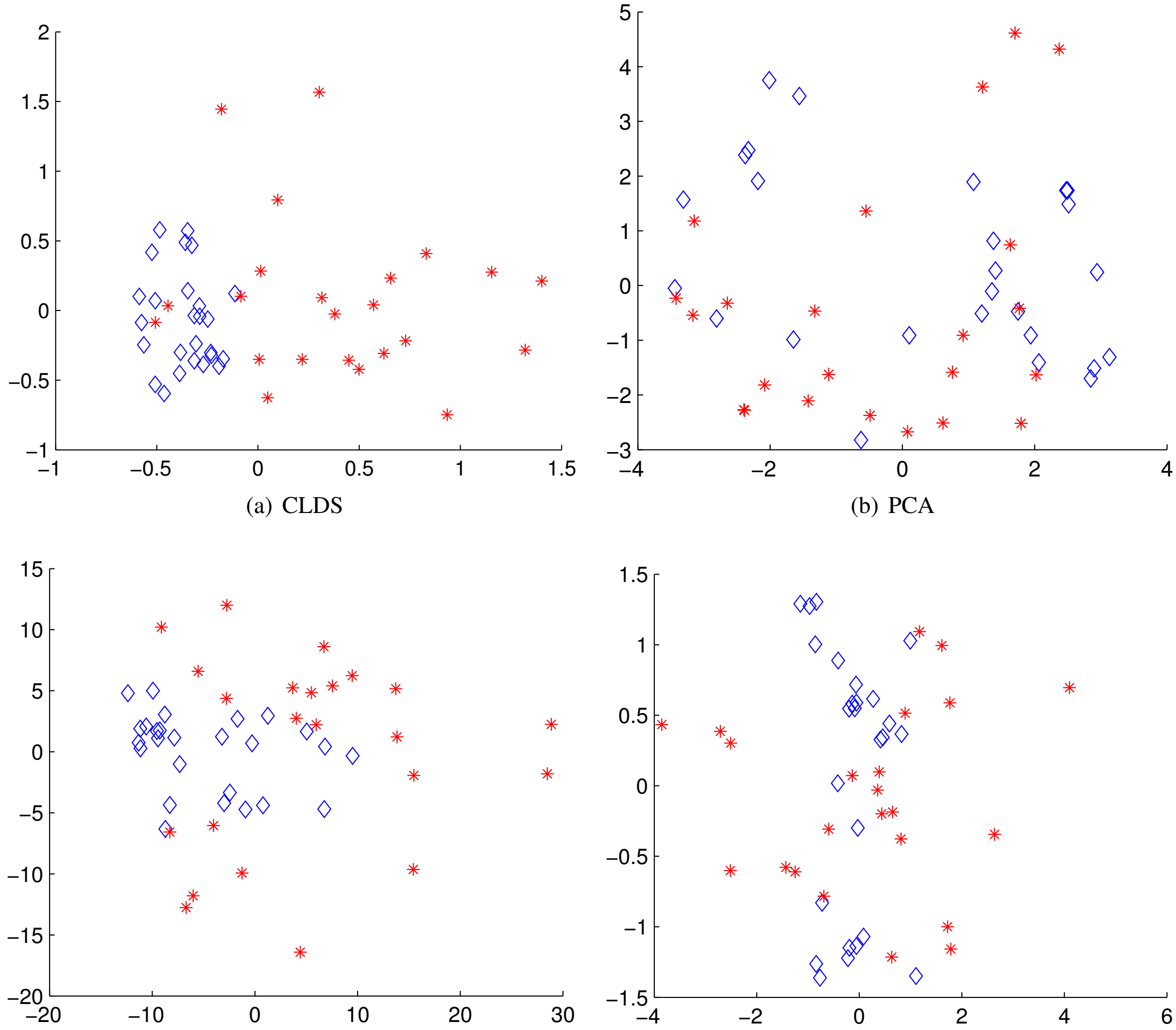


Figure 5.3: Learning curve of CLDS. Note CLDS converges very fast in a few iterations.



(a) CLDS

(b) PCA

(c) DFT

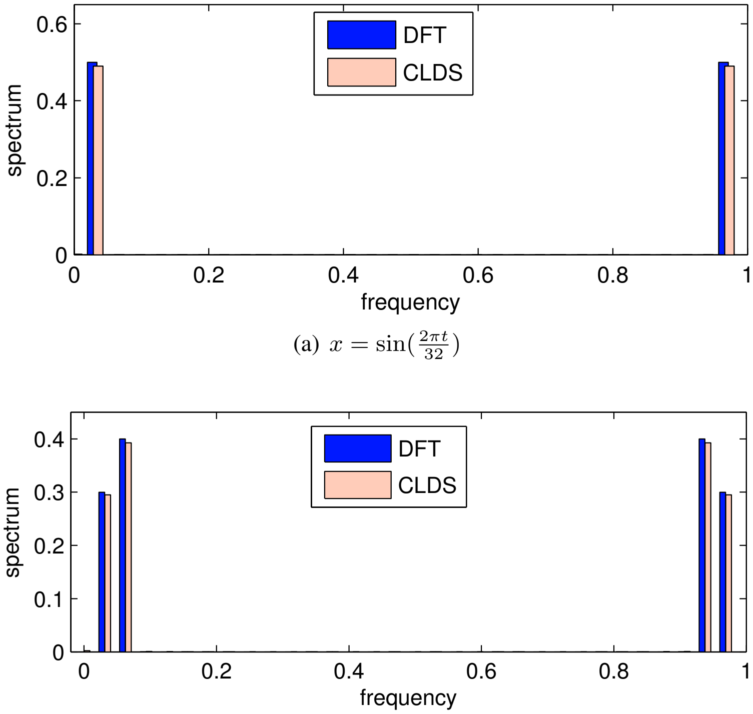
(d) KF

Figure 5.4: Typical scatter plots: Top two features extracted by different methods on `MOCAPPOS`. Note that CLDS produces a clear separated grouping of walking motions (blue ◇) and running motion (red ⋆).

5.5 Discussion and relation with other methods

Relationship to (real-valued) Linear Dynamical Systems The graphical representation of our CLDS is similar to the real-valued linear dynamical systems (LDS, also known as Kalman filters), except that the conditional distribution changes to complex-valued normal distribution.

But due to this, there is a significant difference in the space of the optimal solutions. In LDS, such a space contains many essentially equivalent solutions. Consider a set of estimated parameters for LDS: it will yield equivalent parameters simply by exchanging the order in hidden variables and initial state (and correspondingly columns of $\mathbf{A}$ and $\mathbf{C}$). A generalization of this would be a proper “rotation” of the hidden space, by applying a linear transformation with a orthogonal matrix. Our approach actually tries to find



(a) $x = \sin(\frac{2\pi t}{32})$

(b) $x = 0.6\sin(\frac{2\pi t}{32} + 0.8\sin(\frac{2\pi t}{16})$

Figure 5.5: Frequency spectrums of synthetic signals. Note CLDS can learn spectrums very close to DFT's, by fixing diagonal transition matrices corresponding to base frequencies.

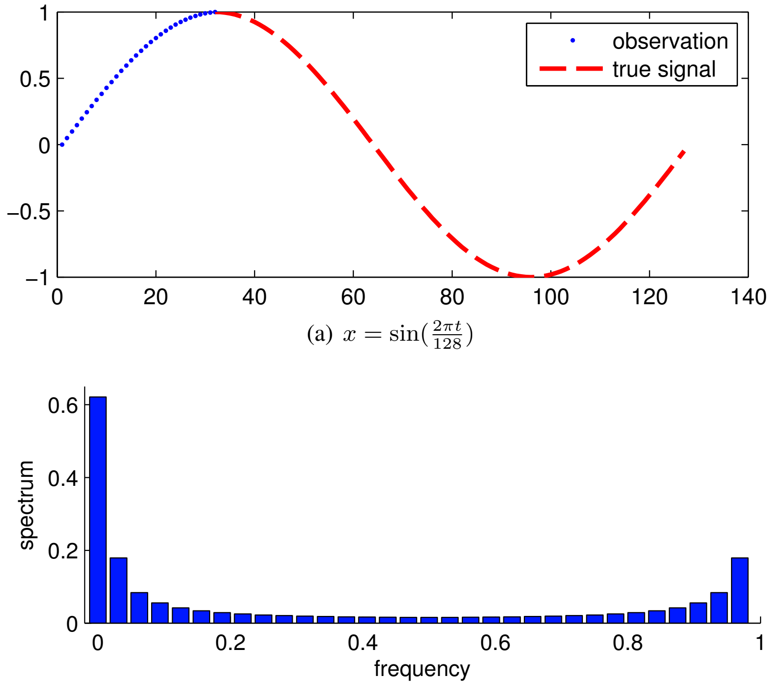
a representative for such an equivalent family of solutions. In traditional Kalman filters, it is not always possible to get the most compact solution with real valued transition matrix, while in our model with the diagonal transition matrix, the solution is invariant in a proper sense.

Furthermore, LDS does not have a explicit notion of time shifts in its model, while in our method, it is already encoded in the phase of initial states and the output matrix $\mathbf{C}$. This is also confirmed by our experiments: LDS does not generate features helpful in clustering, while CLDS significantly improves that.

Relationship to Discrete Fourier Transform CLDS is closely related to Fourier analysis, since the eigenvalues of the transition matrix $\mathbf{A}$ essentially encode a set of base frequencies. In the special restricted case (used for clustering), the diagonal elements of $\mathbf{A}$ directly tell those frequencies. Hence, with proper construction, CLDS includes Discrete Fourier transform (DFT) as a special instance.

Consider one dimensional sequence $x_{1,\cdots,\mathrm{T}}$: we can build a probabilistic version of DFT by fixing $\vec{\mu_0} = \mathbf{1}$, and $\mathbf{A} = \mathrm{diag}(\exp(\frac{2\pi i}{\mathrm{T}}k)), k = 1, \ldots, \mathrm{T}$. We conjecture that if we train such a model on the data, the estimated output matrix $\mathbf{C}$ will be equivalent to the Fourier coefficients from DFT. This is also confirmed by our experiments on synthetic signals. Figure 5.5 exhibits the spectrum of coefficients from DFT and the output matrix $\mathbf{C}$ from CLDS for two signals. They almost perfectly match each other.

Compared to DFT, our proposed method clearly enjoys four benefits: (a) it allows dynamics corresponding



(a) $x = \sin(\frac{2\pi t}{128})$

(b) Spectrum of DFT

Figure 5.6: Limitation of DFT (right) on a partially observed synthetic signal (left). Note DFT can not recover exact frequencies, while by setting hidden dimension to be two, CLDS's estimates are $\vec{a} = \{0.9991 \pm 0.0494i\}$, equivalent to frequencies of $\pm 1/127.19$, close to true signal.

to arbitrary frequency components, contrary to a fixed set of base frequencies as in DFT; (b) being an explicit probabilistic model allows a rich family of extension to other non Gaussian noises; (c) it has direct control over the model complexity and sparsity with the number of hidden variables, i.e. choosing a small number will result in forcing the approximation of the harmonics or frequencies in the data; (d) it can estimate harmonic components jointly present in multiple signals (but with small noise), while it is not straightforward to extend DFT to multiple sequences. For e.g., Figure 5.6 showcases the limitation of DFT on signals only observed for partial cycles: it fails to recognize the exact frequency component in the signal (non-integer multiple of the base frequency), while CLDS can almost perfectly identify the frequency components with two hidden variables.

Other related models Autoregression (AR) is another popular model for time series used for forecasting. CLDS also includes AR as a special case, which can be obtained by setting the output matrix **C** to be the identity matrix. Principal component analysis (PCA) can also be viewed as a special case of CLDS. By setting the transition matrix to be zeros, CLDS degenerates to Probabilistic PCA [Tipping and Bishop, 1999].

5.6 Summary

Motivated by clustering human motion-capture time sequences, in this chapter we developed a novel method of clustering time series data, that can learn joint temporal dynamics in the data (Property P1), handle time lags (Property P2) and produces interpretable features (Property P3). Specifically, our contributions are:

1. *Design of CLDS* : We developed CLDS, complex-valued linear dynamical systems. We then provided Complex-Fit, a novel complex valued EM algorithm to learn the model parameters from the data.
2. *Application to Clustering*: We used a special case of CLDS for time series clustering by forcing a diagonal transition matrix, corresponding to frequencies, without losing any expressiveness but resulting in interpretability. Our clustering method then uses the participation weight (energy) of the hidden variables as features, thus eliminating lags. Hence it satisfies P1, P2 and P3 mentioned before.
3. *Validation*: Our approach produces significant improvement in clustering quality (1.5 to 5 times better than several popular time series clustering methods) when evaluated on synthetic data and real motion capture data.

CLDS is insensitive to the rotations in the hidden variables due to properties of the complex normal distributions. Moreover we showed that CLDS includes PCA, DFT and AR as special cases.

5.A Appendix: Prove and derivation of Complex-Fit

We will describe algorithmic details of the learning algorithm for CLDS. The objective function is expected negative-loglikelihood of CLDS.

$$\begin{aligned}\mathcal{L}(\theta) =& \mathbb{E}_{\mathbf{Z}|\mathbf{X}}[-\log P(\mathbf{X},\mathbf{Z}|\theta)] \\ =& \mathbb{E}[(\vec{z}_1-\vec{\mu}_0)^* \cdot \mathbf{Q}_0^{-1} \cdot (\vec{z}_1-\vec{\mu}_0)] + \mathbb{E}[\sum_{n=1}^{\mathrm{T}-1}(\vec{z}-\mathbf{A}\cdot\vec{z}_n)^* \cdot Q^{-1} \cdot (\vec{z}_{n+1}-\mathbf{A}\cdot\vec{z}_n)] \\ &+ \mathbb{E}[\sum_{n=1}^{\mathrm{T}}(\vec{x}_n-\mathbf{C}\cdot\vec{z}_n)^* \cdot R^{-1} \cdot (\vec{x}_n-\mathbf{C}\cdot\vec{z}_n)] + \log|\mathbf{Q}_0| + (\mathrm{T}-1)\log|Q| + \mathrm{T}\log|R| \end{aligned} \quad (5.5)$$

5.A.1 Learning parameters: Complex-Fit M-step

We will derive the results for Eq. (5.11-5.16). For the completeness, we will repeat several equations already described in the main content.

The M-step is derived by taking derivatives (see Definition 5.3) of the objective function and equating them to zeros.

$$\frac{\partial}{\partial \vec{\mu}_0}\mathcal{L} = 0 \tag{5.32}$$

$$\frac{\partial}{\partial \overline{\vec{\mu}_0}}\mathcal{L} = 0 \tag{5.33}$$

$$\tag{5.34}$$

where

$$\frac{\partial}{\partial \vec{\mu}_0}\mathcal{L} = -(\mathbb{E}[z_1] - \vec{\mu}_0)^* \cdot \mathbf{Q}_0^{-1} \tag{5.35}$$

$$\frac{\partial}{\partial \overline{\vec{\mu}_0}}\mathcal{L} = -(\mathbb{E}[z_1] - \vec{\mu}_0)^T \cdot (\mathbf{Q}_0^{-1})^T \tag{5.36}$$

It follows that

$$\vec{\mu}_0 = \mathbb{E}[\vec{z}_1] \tag{5.37}$$

By taking partial derivative w.r.t. $\mathbf{Q}_0$ and $\overline{\mathbf{Q}_0}$ and equating to zeros,

$$\frac{\partial}{\partial \mathbf{Q}_0}\mathcal{L} = 0 \tag{5.38}$$

$$\frac{\partial}{\partial \overline{\mathbf{Q}_0}}\mathcal{L} = 0 \tag{5.39}$$

where

$$\frac{\partial}{\partial \mathbf{Q}_0}\mathcal{L} = (\mathbf{Q}_0^T)^{-1} - (\mathbf{Q}_0^T)^{-1} \cdot \mathbb{E}[\overline{(\vec{z}_1 - \vec{\mu}_0)} \cdot (\vec{z}_1 - \vec{\mu}_0)^T] \cdot (\mathbf{Q}_0^T)^{-1} \tag{5.40}$$

It follows that

$$\mathbf{Q}_0 = \mathbb{E}[\vec{z}_1 \cdot \vec{z}_1^*] - \vec{\mu}_0 \cdot \vec{\mu}_0^* \tag{5.41}$$

By taking partial derivative w.r.t. $\mathbf{A}$ and $\overline{\mathbf{A}}$ and equating to zeros,

$$\frac{\partial}{\partial \mathbf{A}}\mathcal{L} = 0 \tag{5.42}$$

$$\frac{\partial}{\partial \overline{\mathbf{A}}}\mathcal{L} = 0 \tag{5.43}$$

where

$$\frac{\partial}{\partial \mathbf{A}}\mathcal{L} = -\mathbb{E}[\sum_{n=1}^{\mathrm{T}-1} (\mathbf{Q}^T)^{-1} \cdot \overline{(\vec{z}_{n+1} - \mathbf{A} \cdot \vec{z}_n)} \cdot \vec{z}_n^T] \tag{5.44}$$

$$\frac{\partial}{\partial \overline{\mathbf{A}}}\mathcal{L} = -\mathbb{E}[\sum_{n=1}^{\mathrm{T}-1} \mathbf{Q}^{-1} \cdot (\vec{z}_{n+1} - \mathbf{A} \cdot \vec{z}_n)^T \cdot \overline{\vec{z}_n}] \tag{5.45}$$

It follows that

$$\mathbf{A} = (\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[\vec{z}_{n+1} \cdot \vec{z}_n^*]) \cdot (\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*])^{-1} \tag{5.46}$$

By taking partial derivative w.r.t. $\mathbf{Q}$ and $\overline{\mathbf{Q}}$ and equating to zeros,

$$\frac{\partial}{\partial \mathbf{Q}} \mathcal{L} = 0 \tag{5.47}$$

$$\frac{\partial}{\partial \overline{\mathbf{Q}}} \mathcal{L} = 0 \tag{5.48}$$

where

$$\frac{\partial}{\partial \mathbf{Q}} \mathcal{L} = (\mathrm{T}-1)(\mathbf{Q}^T)^{-1} - (\mathbf{Q}^T)^{-1} \cdot \left(\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[(\overline{\vec{z}_{n+1}} - \overline{\mathbf{A}} \cdot \overline{\vec{z}_n}) \cdot (\vec{z}_{n+1} - \mathbf{A} \cdot \vec{z}_n)^T] \right) \cdot (Q^T)^{-1} \tag{5.49}$$

It follows that,

$$\mathbf{Q} = \frac{1}{\mathrm{T}-1} \sum_{n=1}^{\mathrm{T}-1} \left(\mathbb{E}[\vec{z}_{n+1} \cdot \vec{z}_{n+1}^*] - \mathbb{E}[\vec{z}_{n+1} \cdot \vec{z}_n^*] \cdot \mathbf{A}^* - \mathbf{A} \cdot \mathbb{E}[\vec{z}_n \cdot \vec{z}_{n+1}^*] + \mathbf{A} \cdot \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*] \cdot \mathbf{A}^* \right) \tag{5.50}$$

By taking partial derivative w.r.t. $\mathbf{C}$ and $\overline{\mathbf{C}}$ and equating to zeros,

$$\frac{\partial}{\partial \mathbf{C}} \mathcal{L} = 0 \tag{5.51}$$

$$\frac{\partial}{\partial \overline{\mathbf{C}}} \mathcal{L} = 0 \tag{5.52}$$

where

$$\frac{\partial}{\partial \mathbf{C}} \mathcal{L} = (\mathbf{R}^T)^{-1} \cdot \sum_{n=1}^{\mathrm{T}} (\overline{\mathbf{C}} \cdot \mathbb{E}[\overline{\vec{z}_n} \cdot \vec{z}_n^T] - \overline{\vec{x}_n} \cdot (\mathbb{E}[\vec{z}_n])^T) \tag{5.53}$$

$$\frac{\partial}{\partial \overline{\mathbf{C}}} \mathcal{L} = (\mathbf{R})^{-1} \cdot \sum_{n=1}^{\mathrm{T}} (\mathbf{C} \cdot \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*] - \vec{x}_n \cdot (\mathbb{E}[\vec{z}_n])^*) \tag{5.54}$$

It follows that

$$\mathbf{C} = (\sum_{n=1}^{\mathrm{T}} \vec{x}_n \cdot \mathbb{E}[\vec{z}_n^*]) \cdot (\sum_{n=1}^{\mathrm{T}} \mathbb{E}[\vec{z}_n \cdot \vec{z}_n^*])^{-1} \tag{5.55}$$

By taking the derivatives w.r.t. $\mathbf{R}$ and $\overline{\mathbf{R}}$, and equating to zeros,

$$\frac{\partial}{\partial \mathbf{R}} \mathcal{L} = 0 \tag{5.56}$$

$$\frac{\partial}{\partial \overline{\mathbf{R}}} \mathcal{L} = 0 \tag{5.57}$$

where

$$\frac{\partial}{\partial \mathbf{R}}\mathcal{L} = \mathrm{T}\cdot(\mathbf{R}^T)^{-1} - (\mathbf{R}^T)^{-1}\cdot\left(\sum_{n=1}^{\mathrm{T}}\mathbb{E}[(\overline{\vec{x}_n} - \overline{\mathbf{C}}\cdot\overline{\vec{z}_n})\cdot(\vec{x}_n - \mathbf{C}\cdot\vec{z}_n)^T]\right)\cdot(R^T)^{-1} \tag{5.58}$$

$$\tag{5.59}$$

It follows that

$$\mathbf{R} = \frac{1}{\mathrm{T}}\sum_{n=1}^{\mathrm{T}}\big(\vec{x}_n\cdot\vec{x}_n^* - \vec{x}_n\cdot\mathbb{E}[\vec{z}_n^*]\cdot\mathbf{C}^* - \mathbf{C}\cdot\mathbb{E}[\vec{z}_n]\cdot\vec{x}_n^* + \mathbf{C}\cdot\mathbb{E}[\vec{z}_n\cdot\vec{z}_n^*]\cdot\mathbf{C}^*\big) \tag{5.60}$$

5.A.2 Inference about the latent variables

The above M-step requires computation of the sufficient statistics on the posterior distribution. During the E-step, we will compute mean and covariance of the marginal and joint posterior ditrbutions $P(\vec{z}_n|\mathbf{X})$ and $P(\vec{z}_n, \vec{z}_{n+1}|\mathbf{X})$. The E-step computes the posteriors in with the forward-backward substeps (corresponding to Kalman filtering and smoothing in the traditional LDS). The forward step computes the partial posterior $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n$, and the backward pass computes the full posterior distributions. We will show that all these posteriors are complex normal distributions and will give an algorithm to find their means and variances.

The forward step computes the partial posterior $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n$ from the beginning $\vec{z}_1$ to the tail of the chain $\vec{z}_{\mathrm{T}}$. Throughout the following, we assume that

$$\vec{u}_n = \mathbb{E}[\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n] \tag{5.61}$$

$$\mathbf{U}_n = \mathrm{Var}(\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n) \tag{5.62}$$

Lemma 5.5. *For every $n = 1, 2, \ldots, \mathrm{T}$, the partial posterior distribution $P(\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n)$ follow complex Normal distribution.*

Starting from $n = 1$,

$$\begin{aligned} \vec{z}_1 &\sim \mathcal{CN}(\vec{\mu}_0, \mathbf{Q}_0) \\ \vec{x}_1|\vec{z}_1 &\sim \mathcal{CN}(\mathbf{C}\cdot\vec{z}_1, \mathbf{R}) \\ \text{from Lemma 5.3} \Longrightarrow & \\ \vec{z}_1|\vec{x}_1 &\sim \mathcal{CN}(\vec{u}_1, \mathbf{U}_1) \end{aligned}$$

where

$$\begin{aligned} \vec{u}_1 &= \vec{\mu}_0 + \mathbf{K}_1\cdot(\vec{x}_1 - \mathbf{C}\cdot\vec{\mu}_0) \\ \mathbf{U}_1 &= (I - \mathbf{K}_1\cdot\mathbf{C})\cdot\mathbf{Q}_0 \end{aligned}$$

and $\mathbf{K}_1$ is the complex-valued "Kalman" gain matrix,

$$\mathbf{K}_1 = \mathbf{Q}_0\cdot\mathbf{C}^*\cdot(\mathbf{R} + \mathbf{C}\cdot\mathbf{Q}_0\cdot\mathbf{C}^*)^{-1}$$

Now we have confirmed that the partial posterior for $n = 1$ follows complex normal distribution, we will base our induction from n to $n + 1$.

$$\begin{aligned}
\vec{z}_n|\vec{x}_1, \ldots, \vec{x}_n &\sim \mathcal{CN}(\vec{u}_n, \mathbf{U}_n) && (5.63)\\
\vec{z}_{n+1}|\vec{z}_n, \vec{x}_1, \ldots, \vec{x}_n &\sim \mathcal{CN}(\mathbf{A} \cdot \vec{z}_n, \mathbf{Q}) && (5.64)\\
\vec{z}_{n+1} \perp x_1, \ldots, \vec{x}_n|\vec{z}_n & && (5.65)\\
\text{from Lemma 5.2} \Longrightarrow & && (5.66)\\
\vec{z}_{n+1}|\vec{x}_1, \ldots, \vec{x}_n &\sim \mathcal{CN}(\mathbf{A} \cdot \vec{u}_n, \mathbf{A} \cdot \mathbf{U}_n \cdot \mathbf{A}^* + \mathbf{Q}) && (5.67)
\end{aligned}$$

Together with

$$\vec{x}_{n+1}|\vec{z}_{n+1} \sim \mathcal{CN}(\mathbf{C} \cdot \vec{z}_{n+1}, \mathbf{R})$$

from Lemma 5.3, it follows

$$\vec{z}_{n+1}|\vec{x}_1, \ldots, \vec{x}_{n+1} \sim \mathcal{CN}(\vec{u}_{n+1}, \mathbf{U}_{n+1}) \tag{5.68}$$

where

$$\begin{aligned}
\vec{u}_{n+1} &= \mathbf{A} \cdot \vec{u}_n + \mathbf{K}_{n+1} \cdot (\vec{x}_{n+1} - \mathbf{C} \cdot \mathbf{A} \cdot \vec{u}_n) && (5.69)\\
\mathbf{U}_{n+1} &= (I - \mathbf{K}_{n+1} \cdot \mathbf{C}) \cdot \mathbf{P}_{n+1} && (5.70)
\end{aligned}$$

and we define,

$$\begin{aligned}
\mathbf{P}_{n+1} &= \mathbf{A} \cdot \mathbf{U}_n \cdot \mathbf{A}^* + \mathbf{Q} && (5.71)\\
\mathbf{K}_{n+1} &= \mathbf{P}_{n+1} \cdot \mathbf{C}^* \cdot (\mathbf{R} + \mathbf{C} \cdot \mathbf{P}_{n+1} \cdot \mathbf{C}^*)^{-1} && (5.72)
\end{aligned}$$

The backward step computes the posterior $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_\mathrm{T}$ from the tail $\vec{z}_\mathrm{T}$ to the head of the chain $\vec{z}_1$. Throughout the backward step, we assume,

$$\begin{aligned}
\vec{v}_n &= \mathbb{E}[\vec{z}_n|\vec{x}_1 \ldots \vec{x}_n] && (5.73)\\
\mathbf{V}_n &= \mathrm{Var}(\vec{z}_n|\vec{x}_1 \ldots \vec{x}_\mathrm{T}) && (5.74)
\end{aligned}$$

Obviously, $\vec{v}_\mathrm{T} = \vec{u}_\mathrm{T}$ and $\mathbf{V}_\mathrm{T} = \mathbf{U}_\mathrm{T}$.

Lemma 5.6. *For every $n = 1, 2, \ldots, \mathrm{T}$, the posterior distribution $\vec{z}_n|\vec{x}_1 \ldots \vec{x}_\mathrm{T}$ follows complex Normal distribution.*

We will prove it based on the backward induction from $n + 1$ to n. From the posterior distribution lemma 5.3, in conjunction with the facts of (5.63),(5.64),

$$\vec{z}_n|\vec{z}_{n+1}, \vec{x}_1, \ldots, \vec{x}_n \sim \mathcal{CN}(\vec{u}_n + \mathbf{J}_{n+1}(\vec{z}_{n+1} - \mathbf{A} \cdot \vec{u}_n), (I - \mathbf{J}_{n+1} \cdot \mathbf{A}) \cdot \mathbf{U}_n)$$

where

$$\mathbf{J}_{n+1} = \mathbf{U}_n \cdot \mathbf{A}^* \cdot (\mathbf{A} \cdot \mathbf{U}_n \cdot \mathbf{A}^* + \mathbf{Q})^{-1} = \mathbf{U}_n \cdot \mathbf{A}^* \cdot \mathbf{P}_{n+1}^{-1}$$

Due to the Markov property,

$$P(\vec{z}_n|\vec{z}_{n+1}, \vec{x}_1, \ldots, \vec{x}_\mathrm{T}) = P(\vec{z}_n|\vec{z}_{n+1}, \vec{x}_1, \ldots, \vec{x}_n) \tag{5.75}$$

Together with the assumption

$$\vec{z}_{n+1}|\vec{x}_1, \ldots, \vec{x}_{\mathrm{T}} = \mathcal{CN}(\vec{v}_{n+1}, \mathbf{V}_{n+1}) \tag{5.76}$$

The two random variables conditioned on the data sequence $\mathbf{X}$ will fall in the linear Gaussian case as defined in Lemma. 5.1 and Lemma. 5.2. It follows that the joint distribution will be complex normal distribution, with

$$\vec{v}_n = \vec{u}_n + \vec{J}_{n+1} \cdot (\vec{v}_{n+1} - \mathbf{A} \cdot \vec{u}_n) \tag{5.77}$$

$$\mathbf{V}_n = \mathbf{U}_n + \mathbf{J}_{n+1} \cdot (\mathbf{V}_{n+1} - \mathbf{P}_{n+1}) \cdot \mathbf{J}^*_{n+1} \tag{5.78}$$

$$\mathrm{Cov}(\vec{z}_n, \vec{z}_{n+1}) = \mathbf{J}_{n+1} \cdot \mathbf{V}_{n+1} \tag{5.79}$$

By definition, it follows that

$$\mathbb{E}[\vec{z}_n \cdot \vec{z}^*_n] = \mathbf{V}_n + \vec{v}_n \cdot \vec{v}^*_n \tag{5.80}$$

$$\mathbb{E}[\vec{z}_n \cdot \vec{z}^*_{n+1}] = \mathbf{J}_{n+1} \cdot \mathbf{V}_{n+1} + \vec{v}_n \cdot \vec{v}^*_{n+1} \tag{5.81}$$

Part II

Parallel and Distributed Algorithms

Chapter 6

Parallel Learning for Linear Dynamical Systems

Multi-core processors with ever increasing number of cores per chip are becoming prevalent in modern parallel computing. Our goal is to make use of the multi-core as well as multi-processor architectures to speed up data mining algorithms. Specifically, we present a parallel algorithm for approximate learning of Linear Dynamical Systems (LDS), also known as Kalman Filters (KF). LDSs are widely used in time series analysis such as motion capture modeling and visual tracking etc. We propose CAS-LDS, a novel method to handle the data dependencies due to the chain structure of hidden variables in LDS, so as to parallelize the EM-based parameter learning algorithm. We implement the algorithm using OpenMP on both a supercomputer and a quad-core commercial desktop. The experimental results show that parallel algorithms using CAS-LDS achieve comparable accuracy and almost linear speedups over the serial version. In addition, CAS-LDS can be generalized to other models with similar linear structures such as Hidden Markov Models (HMM), which will be introduced in next chapter.

6.1 Introduction

In the previous chapters, we have already witnessed numerous applications of time series, including motion capture [Li et al., 2008], visual tracking, speech recognition, quantitative studies of financial markets, network intrusion detection, forecasting, etc. Mining and forecasting are popular operations relevant to time series analysis. Two typical statistical models for such problems are hidden Markov models (HMM) and linear dynamical systems (LDS, also known as Kalman filters). Both assume linear transitions on hidden (i.e. 'latent') variables which are considered discrete for HMM and continuous for LDS. The hidden states or variables in both models can be inferred through a forward-backward procedure involving dynamic programming. However, the maximum likelihood estimation of model parameters is difficult, requiring the well-known Expectation-Maximization (EM) method [Bishop, 2006]. The EM algorithm for learning of LDS/HMM iterates between computing conditional expectations of hidden variables through the forward-backward procedure (E-step) and updating model parameters to maximize its likelihood (M-step). Although EM algorithm generally produces good results, the EM iterations may take long to converge. Meanwhile, the computation time of E-step is linear in the length of the time series but cubic in the dimensionality of observations, which results in poor scaling on high dimensional data. For example, our experimental results show that on a 93-dimensional dataset of length over 300, the EM algorithm would take over one second to compute each iteration and over ten minutes to converge on a high-end multi-core commercial computer. Such capacity may not be able to fit modern computation-intensive applications with large amounts of data or real-time constraints. While there are efforts to speed up the forward-backward procedure with moderate assumptions such as sparsity or existence of low-dimensional approximation, we will focus on taking advantage of the quickly developing parallel processing technologies to achieve dramatic speedup.

Traditionally, the EM algorithm for LDS running on a multi-core computer only takes up a single core with limited processing power, and the current state-of-the-art dynamic parallelization techniques such as speculative execution [Colohan et al., 2006] benefit little to the straightforward EM algorithm due to the nontrivial data dependencies in LDS. As the number of cores on a single chip keeps increasing, soon we may be able to build machines with even a thousand cores, e.g. an energy efficient, 80-core chip not much larger than the size of a finger nail was released by Intel researchers in early 2007 [Intel, 2007]. This chapter is along the line to investigate the following question: how much speed up could we obtain for machine learning algorithms on multi-core? There are already several papers on distributed computation for data mining operations. For example, "cascade SVMs" were proposed to parallelize Support Vector Machines [Graf et al., 2005]. Other articles use Google's map-reduce techniques [Dean and Ghemawat, 2008] on multi-core machines to design efficient parallel learning algorithms for a set of standard machine learning algorithms/models such as naïve Bayes and PCA, achieving almost linear speedup [Chu et al., 2006, Ranger et al., 2007]. However, these methods do not apply to HMM or LDS directly. In essence, their techniques are similar to dot-product-like parallelism, by using divide-and-conquer on independent sub models; these do not work for models with complicated data dependencies such as HMM and LDS. [1]

In this chapter, we propose the Cut-And-Stitch method (CAS), which avoids the data-dependency problems. We show that CAS can quickly and accurately learn an LDS in parallel, as demonstrated on two popular architectures for high performance computing. The basic idea of our algorithm is to (a) *Cut* both the chain of hidden variables as well as the observed variables into smaller blocks, (b) perform intra-block computation, and (c) *Stitch* the local results seamlessly by summarizing sufficient statistics and updating model parameters and an additional set of block-specific parameters. The algorithm would iterate over 4 steps, where the most time-consuming E-step in EM as well as the two newly introduced steps could be parallelized with little synchronization overhead. Furthermore, this approximation of global models by local sub-models sacrifices only a little accuracy, due to the chain structure of LDS (also HMM), as shown in our experiments, which was our first goal. On the other hand, it yields almost linear speedup, which was our second main goal.

The rest of the chapter is organized as follows. We will present our proposed Cut-And-Stitch method in Section 6.3. Then we describe the programming interface and implementation issues in Section 6.4. We present experimental results Section 6.5.

[1] Or exactly, models with large diameters. The diameter of a model is the length of longest acyclic path in its graphical representation. For example, the diameter of the LDS in Figure 2.2 is N.

6.2 Previous work on parallel data mining

Data mining and parallel programming receives increasing interest. Parthasarathy et al. [Buehrer et al., 2007] develop parallel algorithms for mining terabytes of data for frequent itemsets, demonstrating a near-linear scale-up on up to 48 nodes.

Reinhardt and Karypis [Reinhardt and Karypis, 2007] used OpenMP to parallelize the discovery of frequent patterns in large graphs, showing excellent speedup of up to 30 processors.

Cong et al. [Cong et al., 2005] develop the Par-CSP algorithm that detects closed sequential patterns on a distributed memory system, and report good scale-up on a 64-node Linux cluster.

Graf et al. [Graf et al., 2005] developed a parallel algorithm to learn SVM through cascade SVM. Collobert et al. [Collobert et al., 2002] proposed a method to learn a mixture of SVM in parallel. Both of them adopted the idea of splitting dataset into small subsets, training SVM on each, and then combining those SVMs. Chang et al. [Chang et al., 2007] proposed PSVM to train SVMs on distributed computers through approximate factorization of the kernel matrix.

There is an attempt to use Google's Map-Reduce [Dean and Ghemawat, 2008] to parallelize a set of learning algorithm such as naïve-Bayes, PCA, linear regression and other similar algorithms [Chu et al., 2006, Ranger et al., 2007]. Their framework requires the summation form (like dot-product) in the learning algorithm, and hence could distribute independent calculations to many processors and then summarize them together. Therefore the same techniques could hardly be used to learn long sequential graphical models such as Hidden Markov Models and Linear Dynamical Systems.

6.3 Cut-And-Stitch for LDS

In the standard EM learning algorithm described in Section 2.2.5, the chain structure of the LDS enforces the data dependencies in both the forward computation from $\vec{z}_n$ (e.g. $\mathbb{E}[\vec{z}_n \mid \mathcal{Y}; \theta]$) to $\vec{z}_{n+1}$ and the backward computation from $\vec{z}_{n+1}$ to $\vec{z}_n$ In this section, we will present ideas on overcoming such dependencies and describe the details of Cut-And-Stitch parallel learning algorithm.

6.3.1 Intuition and Preliminaries

Our guiding principle to reduce the data dependencies is to divide LDS into smaller, independent parts. Given a data sequence $\mathcal{Y}$ and k processors with shared memory, we could cut the sequence into k subsequences of equal sizes, and then assign one processor to each subsequence. Each processor will learn the parameters, say $\theta_1, \ldots, \theta_k$, associated with its subsequence, using the basic, sequential EM algorithm. In order to obtain a consistent set of parameters for the whole sequence, we use a non-trivial method to summarize all the sub-models rather than simply averaging. Since each subsequence is treated independently, our algorithm will obtain near k-fold speedup. The main design challenges are: (a) how to minimize the overhead in synchronization and summarization, and (b) how to retain the accuracy of the learning algorithm. Our Cut-And-Stitch method (or CAS) is targeting both challenges.

Given a sequence of observed values $\mathcal{Y}$ with length of T, the learning goal is to best fit the parameters $\theta = (\mu_0, \Gamma, F, \Lambda, G, \Sigma)$. The Cut-And-Stitch (CAS) algorithm consists of two alternating steps: the *Cut* step and the *Stitch* step. In the *Cut* step, the Markov chain of hidden variables and corresponding observations

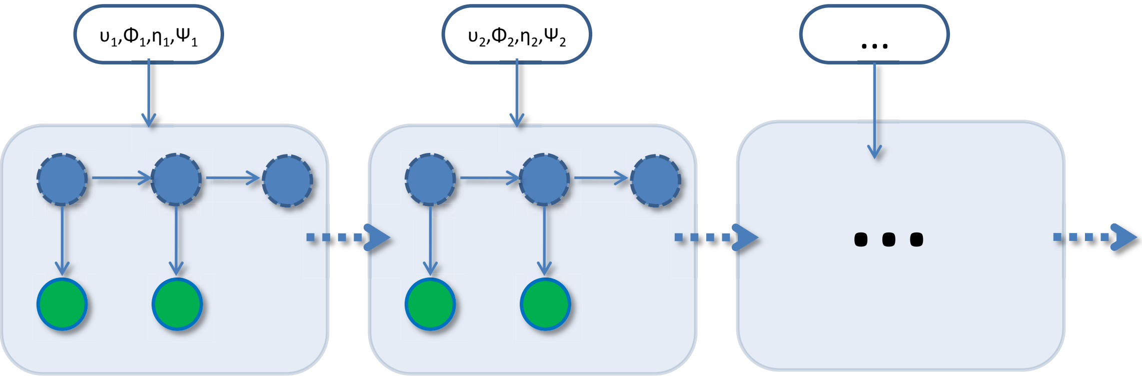


Figure 6.1: Graphical illustration of dividing LDS into blocks in the *Cut* step. Note *Cut* introduces additional parameters for each block.

are divided into smaller blocks, and each processor performs the local computation for each block. More importantly, it computes the initial beliefs (marginal expectation of hidden variables) for its block, based on the neighboring blocks, and then it computes the improved beliefs for its block, independently. In the *Stitch* step, each processor computes summary statistics for its block, and then the parameters of LDS are updated globally to maximize the EM learning objective function (also known as the *expected complete log-likelihood*). Besides, local parameters for each block are also updated to reflect changes in the global model. The CAS algorithm iterates between *Cut* and *Stitch* until convergence.

6.3.2 Cut step

The objective of *Cut* step is to compute the marginal posterior distribution of $\vec{z}_n$, conditioned on the observations $\vec{y}_1, \ldots, \vec{y}_\mathrm{T}$ given the current estimated parameter θ: $P(\vec{z}_n|\vec{y}_1, \ldots, \vec{y}_\mathrm{T}; \theta)$. Given the number of processors k and the observation sequence, we first divide the hidden Markov chain into k blocks: B_1,. . . , B_k, with each block containing the hidden variables $\vec{z}$, the observations $\vec{y}$, and four extra parameters υ, Φ, η, Ψ. The sub-model for i-th block B_i is described as follows (see Figure 6.1):

$$P(\vec{z}_{i,1}) = \mathcal{N}(\upsilon_i, \Phi_i) \tag{6.1}$$

$$P(\vec{z}_{i,j+1}|\vec{z}_{i,j}) = \mathcal{N}(\mathbf{F}\vec{z}_{i,j}, \Lambda) \tag{6.2}$$

$$P(\vec{z}'_{i,T}|\vec{z}_{i,T}) = \mathcal{N}(\mathbf{F}\vec{z}_{i,T}, \Lambda) \tag{6.3}$$

$$P(\vec{y}_{i,j}|\vec{z}_{i,j}) = \mathcal{N}(\mathbf{G}\vec{z}_{i,j}, \Sigma) \tag{6.4}$$

where the block size $T = \frac{\mathrm{T}}{k}$ and $j = 1 \ldots T$ indicating j-th variables in i-th block ($\vec{z}_{i,j} = \vec{z}_{(i-1)*T+j}$ and $\vec{y}_{i,j} = \vec{y}_{(i-1)*T+j}$). η_i, Ψ_i could be viewed as messages passed from next block, through the introduction of an extra hidden variable $\vec{z}'_{i,T}$.

$$P(\vec{z}'_{i,T}) = \mathcal{N}(\eta_i, \Psi_i) \tag{6.5}$$

Intuitively, the *Cut* tries to approximate the global LDS model by local sub-models, and then compute the marginal posterior with the sub-models. The blocks are both logical and computational, meaning that most computation about each logical block resides on one processor. In order to simultaneously and accurately compute all blocks on each processor, the block parameters should be well chosen with respect to the other blocks. We will describe the parameter estimation later but here we first describe the criteria. From the Markov properties of the LDS model, the marginal posterior of $\vec{z}_{i,j}$ conditioned on $\mathcal{Y}$ is independent of any observed $\vec{y}$ outside the block B_i, as long as the block parameters satisfy:

$$P(\vec{z}_{i,1}|\vec{y}_1,\ldots,\vec{y}_{i-1,T}) = \mathcal{N}(\upsilon_i, \Phi_i) \tag{6.6}$$

$$P(\vec{z}_{i+1,1}|\vec{y}_1,\ldots,\vec{y}_{\mathrm{T}}) = \mathcal{N}(\eta_i, \Psi_i) \tag{6.7}$$

Therefore, we could derive a local belief propagation algorithm to compute the marginal posterior $P(\vec{z}_{i,j}|\vec{y}_{i,1}\ldots\vec{y}_{i,T};\upsilon_i,\Phi_i,\eta_i,\Psi_i,\theta)$. Both computation for the forward passing and the backward passing can reside in one processor without interfering with other processors except possibly in the beginning. The local forward pass computes the posterior up to current time tick within one block $P(\vec{z}_{i,j}|\vec{y}_{i,1}\ldots\vec{y}_{i,j})$, while the local backward pass calculates the whole posterior $P(\vec{z}_{i,j}|\vec{y}_{i,1}\ldots\vec{y}_{i,T})$ (to save space, we omit the parameters). Using the properties of linear Gaussian conditional distribution and Markov properties (Chap.2 &8 in [Bishop, 2006]), one can easily infer that both posteriors are Gaussian distributions, denoted as:

$$P(\vec{z}_{i,j}|\vec{y}_{i,1}\ldots\vec{y}_{i,j}) = \mathcal{N}(\mu_{i,j}, \mathbf{V}_{i,j}) \tag{6.8}$$

$$P(\vec{z}_{i,j}|\vec{y}_{i,1}\ldots\vec{y}_{i,T}) = \mathcal{N}(\hat{\mu}_{i,j}, \hat{\mathbf{V}}_{i,j}) \tag{6.9}$$

We can obtain the following forward-backward propagation equations from Eq (6.1-6.5) by substituting Eq (6.6-6.9) and expanding.

$$\mathbf{P}_{i,j-1} = \mathbf{F}\mathbf{V}_{i,j-1}\mathbf{F}^T + \Lambda \tag{6.10}$$

$$\mathbf{K}_{i,j} = \mathbf{P}_{i,j-1}\mathbf{G}^T(\mathbf{G}\mathbf{P}_{i,j-1}\mathbf{G}^T + \Sigma)^{-1} \tag{6.11}$$

$$\vec{\mu}_{i,j} = \mathbf{F}\vec{\mu}_{i,j-1} + \mathbf{K}_{i,j}(\vec{y}_{i,j} - \mathbf{G}\mathbf{F}\vec{\mu}_{i,j-1}) \tag{6.12}$$

$$\mathbf{V}_{i,j} = (\mathbf{I} - \mathbf{K}_{i,j})\mathbf{P}_{i,j-1} \tag{6.13}$$

The initial values are given by:

$$\mathbf{K}_{i,1} = \boldsymbol{\Phi}_i\mathbf{G}^T(\mathbf{G}\boldsymbol{\Phi}_i\mathbf{G}^T + \Sigma)^{-1} \tag{6.14}$$

$$\vec{\mu}_{i,1} = \vec{\upsilon}_i + \mathbf{K}_{i,1}(\vec{y}_{i,1} - \mathbf{G}\vec{\upsilon}_i) \tag{6.15}$$

$$\mathbf{V}_{i,1} = (\mathbf{I} - \mathbf{K}_{i,1})\boldsymbol{\Phi}_i \tag{6.16}$$

The backward passing equations are:

$$\mathbf{J}_{i,j} = \mathbf{V}_{i,j}\mathbf{F}^T(\mathbf{P}_{i,j})^{-1} \tag{6.17}$$

$$\hat{\vec{\mu}}_{i,j} = \vec{\mu}_{i,j} + \mathbf{J}_{i,j}(\hat{\vec{\mu}}_{i,j+1} - \mathbf{F}\vec{\mu}_{i,j}) \tag{6.18}$$

$$\hat{\mathbf{V}}_{i,j} = \mathbf{V}_{i,j} + \mathbf{J}_{i,j}(\hat{\mathbf{V}}_{i,j+1} - \mathbf{P}_{i,j})\mathbf{J}_{i,j}^T \tag{6.19}$$

The initial values are given by:

$$\mathbf{J}_{i,T} = \mathbf{V}_{i,T}\mathbf{F}^T(\mathbf{F}\mathbf{V}_{i,T}\mathbf{F}^T + \Lambda)^{-1} \tag{6.20}$$

$$\hat{\vec{\mu}}_{i,T} = \vec{\mu}_{i,T} + \mathbf{J}_{i,T}(\eta_i - \mathbf{F}\vec{\mu}_{i,T}) \tag{6.21}$$

$$\hat{\mathbf{V}}_{i,T} = \mathbf{V}_{i,T} + \mathbf{J}_{i,T}(\Psi_i - \mathbf{F}\mathbf{V}_{i,T}\mathbf{F}^T - \Lambda)\mathbf{J}_{i,T}^T \tag{6.22}$$

Except for the last block:

$$\hat{\vec{\mu}}_{k,T} = \vec{\mu}_{i,T} \tag{6.23}$$

$$\hat{\mathbf{V}}_{k,T} = \mathbf{V}_{i,T} \tag{6.24}$$

6.3.3 Stitch step

In the *Stitch* step, we estimate the block parameters, collect the statistics and compute the most suitable LDS parameters for the whole sequence. The parameters $\theta = (\mu_0, \Gamma, F, \Lambda, G, \Sigma)$ is updated by maximizing over the expected complete log-likelihood function:

$$Q(\theta^{new}, \theta^{old}) = \mathbb{E}_{\theta^{old}}[\log p(\vec{y}_1 \ldots \vec{y}_N, \vec{z}_1 \ldots \vec{z}_N | \theta^{new})] \tag{6.25}$$

Now taking the derivatives of Eq 6.25 and zeroing out give the updating equations (Eq (6.32-6.37)). The maximization is similar to the M-step in EM algorithm of LDS, except that it should be computed in a distributed manner with the available k processors. The solution depends on the statistics over the hidden variables, which are easy to compute from the forward-backward propagation described in *Cut*.

$$\mathbb{E}[\vec{z}_{i,j}] = \hat{\vec{\mu}}_{i,j} \tag{6.26}$$

$$\mathbb{E}[\vec{z}_{i,j}\vec{z}^T_{i,j-1}] = \mathbf{J}_{i,j-1}\hat{\mathbf{V}}_{i,j} + \hat{\vec{\mu}}_{i,j}\hat{\vec{\mu}}^T_{i,j-1} \tag{6.27}$$

$$\mathbb{E}[\vec{z}_{i,j}\vec{z}^T_{i,j}] = \hat{\mathbf{V}}_{i,j} + \hat{\vec{\mu}}_{i,j}\hat{\vec{\mu}}^T_{i,j} \tag{6.28}$$

where the expectations are taken over the posterior marginal distribution $p(\vec{z}_n | \vec{y}_1, \ldots, \vec{y}_N)$. The next step is to collect the sufficient statistics of each block on every processor.

$$\tau_i = \sum_{j=1}^{T} y_{i,j}\mathbb{E}[\vec{z}^T_{i,j}] \tag{6.29}$$

$$\xi_i = \mathbb{E}[\vec{z}_{i,1}\vec{z}^T_{i-1,T}] + \sum_{j=2}^{T} \mathbb{E}[\vec{z}_{i,j}\vec{z}^T_{i,j-1}] \tag{6.30}$$

$$\zeta_i = \sum_{j=1}^{T} \mathbb{E}[\vec{z}_{i,j}\vec{z}^T_{i,j}] \tag{6.31}$$

To ensure its correct execution, statistics collecting should be run after all of the processors finish their *Cut* step, enabled through the *synchronization* among processors. With the local statistics for each block,

$$\vec{\mu}_0^{new} = \hat{\vec{\mu}}_{1,1} \tag{6.32}$$

$$\mathbf{\Gamma}_0^{new} = \hat{\mathbf{V}}_{1,1} \tag{6.33}$$

$$\mathbf{F}^{new} = \left(\sum_{i=1}^{k} \xi_i\right)\left(\sum_{i=1}^{k} \zeta_i - \mathbb{E}[\vec{z}_N\vec{z}^T_N]\right)^{-1} \tag{6.34}$$

$$\mathbf{\Lambda}^{new} = \frac{1}{N-1}\left(\sum_{i=1}^{k}(\zeta_i - \mathbf{F}^{new}\xi_i^T - \xi_i(\mathbf{F}^{new})^T) + \mathbf{F}^{new}(\sum_{i=1}^{k}\zeta_i - \mathbb{E}[\vec{z}_N\vec{z}^T_N])(\mathbf{F}^{new})^T - \mathbb{E}[\vec{z}_{1,1}\vec{z}^T_{1,1}]\right) \tag{6.35}$$

$$\mathbf{G}^{new} = \left(\sum_{i=1}^{k} \tau_i\right)\left(\sum_{i=1}^{k} \zeta_i\right)^{-1} \tag{6.36}$$

$$\mathbf{\Sigma}^{new} = \frac{1}{N}\left(Cov(\mathcal{Y}) + \sum_{i=1}^{k}(-\mathbf{G}^{new}\tau_i^T - \tau_i(\mathbf{G}^{new})^T + \mathbf{G}^{new}\zeta_i(\mathbf{G}^{new})^T)\right) \tag{6.37}$$

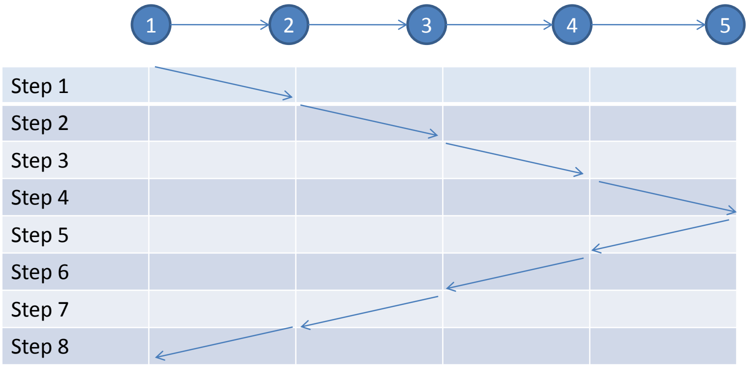


Figure 6.2: Graphical illustration of forward-backward algorithms. A sequential algorithm requires 8 passing steps, with consecutive depending on the previous.

where $Cov(\mathcal{Y})$ is the covariance of the observation sequences and could be precomputed.

$$Cov(\mathcal{Y}) = \sum_{n=1}^{N} \vec{y}_n \vec{y}_n^T$$

As we estimate the block parameters with the messages from the neighboring blocks, we could reconnect the blocks. Recall the conditions in Eq (6.6-6.7), we could approximately estimate the block parameters with the following equations.

$$\upsilon_i = \mathbf{F}\mu_{i-1,T} \tag{6.38}$$

$$\Phi_i = \mathbf{F}\mathbf{V}_{i,T}\mathbf{F}^T + \Lambda \tag{6.39}$$

$$\eta_i = \hat{\vec{\mu}}_{i+1,1} \tag{6.40}$$

$$\Psi_i = \hat{\mathbf{V}}_{i+1,1} \tag{6.41}$$

Except for the first block (no need to compute η_k and Ψ_k for the last block):

$$\upsilon_1 = \mu_0 \tag{6.42}$$

$$\Phi_1 = \Gamma \tag{6.43}$$

The EM algorithm is an iterative learning procedure and the message passing in E step is repeated until convergence. Here is the work flow of our algorithm: first, store the messages/beliefs from the previous iteration; second, passing the messages one step further based on the previous iteration.In, this way, the message passed is NOT dependent on the message of previous node but on the previous iteration. So we could pass the message as the iteration goes. Figure 6.3 shows the parallel algorithm in a EM setting. Note the passing of the messages will continue as the iteration repeats.

In summary, the parallel learning algorithm works in the following two steps, which could be further divided into four sub-steps:

Cut divides and builds small sub-models (blocks), and then each processor *estimate* (E) in parallel posterior marginal distribution in Eq (6.26-6.28), which includes *forward* and *backward* propagation of beliefs.

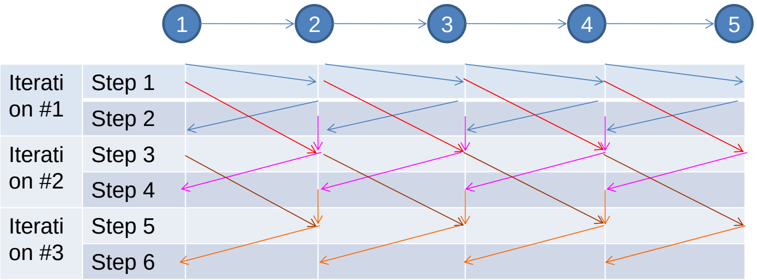


Figure 6.3: Graphical illustration of the parallel forward-backward algorithms in EM iterations. A parallel algorithm requires 2 passing steps on 4 cores/processors, with consecutive depending on the stored previous beliefs.

Stitch estimates the parameters through *collecting* (C) local statistics of hidden variables in each block Eq (6.29-6.31), taking the *maximization* (M) of the expected log-likelihood over the parameters Eq (6.32-6.37), and connecting the blocks by *re-estimate* (R) the block parameters Eq (6.38-6.43).

To extract the most parallelism, any of the above equations independent of each other could be computed in parallel. Computation of the local statistics in Eq (6.29-6.31) is done in parallel on k processors. Until all local statistics are computed, we use one processor to calculate the parameter using Eq (6.32-10.40). Upon the completion of computing the model parameters, every processor computes its own block parameters in Eq (6.38-6.43). To ensure the correct execution, *Stitch* step should run after all of the processors finish their *Cut* step, which is enabled through the synchronization among processors. Furthermore, we also use synchronization to ensure *Maximization* part after *Collecting* and *Re-estimate* after *Maximization*. An interesting finding is that our method includes the sequential version of the learning algorithm as a special case. Note if the number of processors is 1, the Cut-And-Stitch algorithm falls back to the conventional EM algorithm sequentially running on single processor.

6.3.4 Warm-up step

In the first iteration of the algorithm, there are undefined initial values of block parameters υ,Φ,η and Ψ, needed by the forward and backward propagations in *Cut*. A simple approach would be to assign random initial values, but this may lead to poor performance. We propose and use an alternative method: we run a sequential forward-backward pass on the whole observation, estimate parameters, i.e. we execute the *Cut* step with one processor, and the *Stitch* step with k processors. After that, we begin normal iterations of Cut-And-Stitch with k processors. We refer to this step as the *warm-up* step. Although we sacrifice some speedup, the resulting method converges faster and is more accurate. Figure 6.4 illustrates the time line of the whole algorithm on four CPUs.

6.4 Implementation

We will first discuss properties of our proposed CLDS method and what it implies for the requirements of the computer architecture:

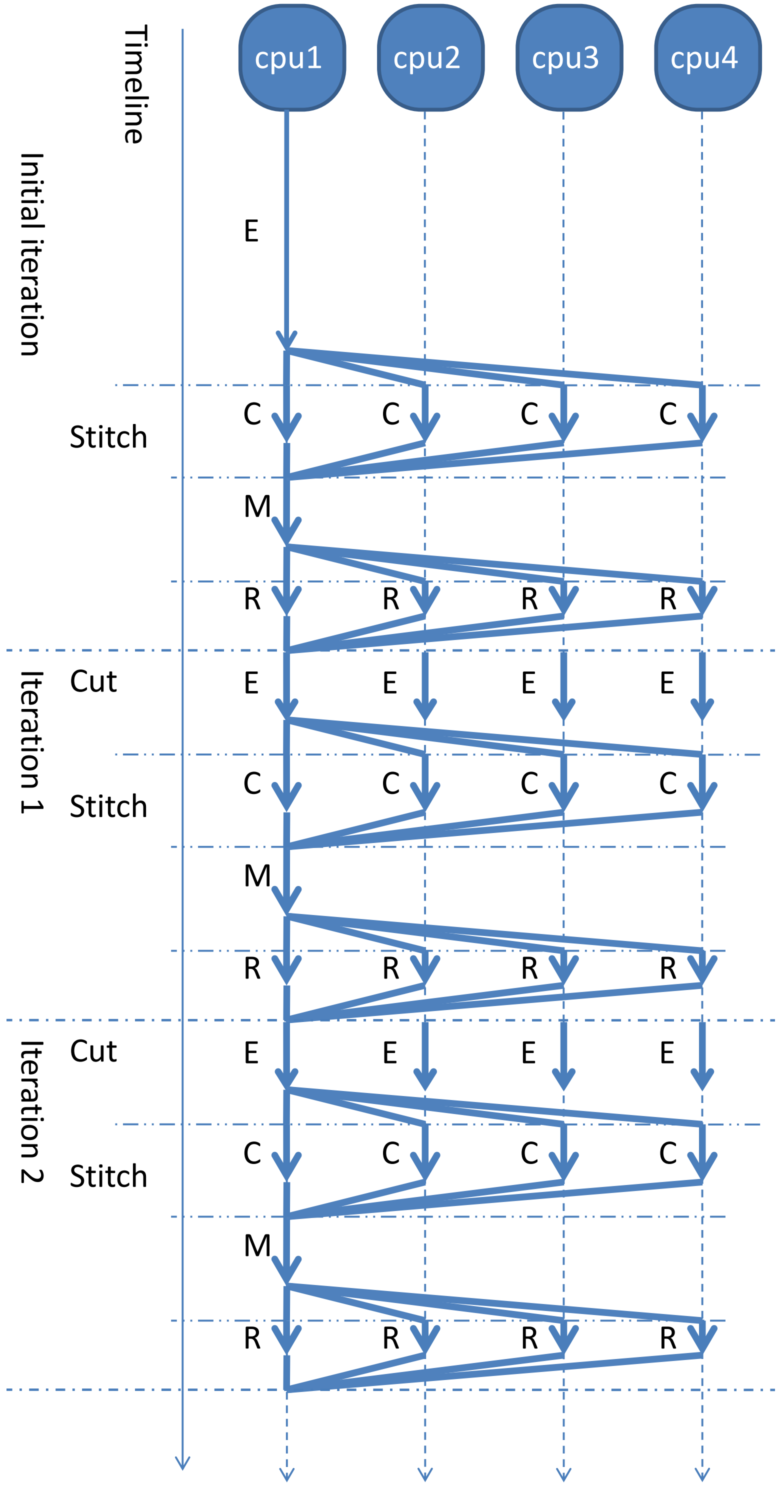


Figure 6.4: Graphical illustration of Cut-And-Stitch algorithm on 4 CPUs. Arrows indicates the computation on each CPU. Tilting lines indicate the necessary synchronization and data transfer between the CPUs and main memory. Tasks labeled with "E" indicate the (parallel) *estimation* of the posterior marginal distribution, including the forward-backward propagation of beliefs within each block as shown in Figure 6.1. (C) indicates the *collection* of local statistics of the hidden variables in each block; (M) indicates the *maximization* of the expected log-likelihood over the parameters, and then it *re-estimates* (R) the block parameters.

- **Symmetric**: The *Cut* step creates a set of equally-sized blocks assigned to each processor. Since the amount of computation depends on the size of the block, our method achieves good load balancing on symmetric processors.
- **Shared Memory**: The *Stitch* step involves summarizing sufficient statistics collected from each processor. This step can be done more efficiently in shared memory, rather than in distributed memory.
- **Local Cache**: In order to reduce the impact of the bottleneck of processor-to-memory communication, local caches are necessary to keep data for each block.

The current Symmetric MultiProcessing (SMP) technologies provide opportunities to match all of these assumptions. We implement our parallel learning algorithm for LDS using OpenMP, a multi-programming interface that supports shared memory on many architectures, including both commercial desktops and supercomputer clusters. Our choice of the multi-processor API is based on the fact that OpenMP is flexible and fast, while the code generation for the parallel version is decoupled from the details of the learning algorithm. We use the OpenMP to create multiple threads, share the workload and synchronize the threads among different processors. Note that OpenMP needs compiler support to translate parallel directives to run-time multi-threading. And it also includes its own library routines (e.g. timing) and environment variables (e.g. the number of running processors).

The algorithm is implemented in C++. Several issues on configuring OpenMP for the learning algorithm are listed as follows:

- **Variable Sharing** Conditional expectation in the E-step are stored in global variables of OpenMP, visible to every processor. There are also several intermediate matrices and vectors results for which only local copies need to be kept; they are temporary variables that belong to only one processor. This also saves the computational cost by preserving locality and reducing cache miss rate.
- **Dynamic or Static Scheduling** What is a good strategy to assign blocks to processors? Usually there are two choices: static and dynamic. Static scheduling will fix processor to always operate on the same codes while dynamic scheduling takes an on-demand approach. We pick the static scheduling approach (i.e. fix the block-processor mapping), for the following reasons: (a) the computation is logically block-wise and in a regular fashion and (b) we have performance gains by exploiting the temporal locality when we always asociate the same processor with the same block. Furthermore, in our implementation, we improve the M-step by using four processors to calculate model parameters in Eq (6.32-6.37): two for Eq (6.32-6.33), one for Eq (6.34-6.35) and one for Eq (6.36-6.37).
- **Synchronization** As described earlier, the *Stitch* step of the learning algorithm should happen only after the *Cut* step has completed, and the order of stages inside *Stitch* should be *collecting*, *maximization* and *re-estimate*. We put barriers after each step/stage to synchronize the threads and keep them in the same pace. Each iteration would include four barriers, as shown in Figure 6.4.

6.5 Evaluation

To evaluate the effectiveness and usefulness of our proposed Cut-And-Stitch method in practical applications, we tested our implementation on SMPs and did experiments on real data. Our goal is to answer the following questions:

Table 6.1: Wall-clock time for the case of a walking motion (#22) on multi-processor/multi-core (in seconds), and the average of normalized running time on 58 motions (serial time= 1).

# of Procs	time (sec.)	avg. of norm. time
1(serial)	3942	1
2	1974	0.5
4	998	0.256
8	510	0.134
16	277	0.0703
32	171	0.0438
64	117	0.0342
128	115	0.0335

- Speedup: how would the performance change as the number of processors/cores increase?
- Quality: how accurate is the parallel algorithm, compared to serial one?

We will first describe the experimental setup and the dataset we used.

6.5.1 Dataset and Experimental Setup

We run the experiments on a supercomputer as well as on a commercial desktop, both of which are typical SMPs.

- The supercomputer is an SGI Altix system[2], at National Center for Supercomputing Applications (NCSA). The cluster consists of 512 1.6GHz Itanium2 processors, 3TB of total memory and 9MB of L3 cache per processor. It is configured with an Intel C++ compiler supporting OpenMP.
- The test desktop machine has two Intel Xeon dual-core 3.0GHz CPUs (a total of four cores), 16G memory, running Linux (Fedora Core 7) and GCC 4.1.2 (supporting OpenMP).

We used a 17MB motion dataset from CMU Motion Capture Database [3]. It consists of 58 walking, running and jumping motions, each with 93 bone positions in body local coordinates. The motions span several hundred frames long (100∼500). We use our method to learn the transition dynamics and projection matrix of each motion, using H=15 hidden dimensions.

6.5.2 Speedup

We did experiment on all of the 58 motions with various number of processors on both machine. The speedup for k processors is defined as

$$S_k = \frac{\text{running time with a single processor}}{\text{running time with } k \text{ processors}}$$

According to Amdahl's law, the theoretical limit of speedup is

$$S_k \leq \frac{1}{(1-p)+\frac{p}{k}} < k$$

[2] cobalt.ncsa.uiuc.edu

[3] http://mocap.cs.cmu.edu/

Table 6.2: Rough estimation of the number of arithmetic operations $(+, -, \times, /)$ in E, C, M, R sub steps of CAS-LDS. Each type of operation is equally weighted, and only the largest portions in each step are kept.

	#of operation
E	$N \cdot (m^3 + H \cdot m^2 + m \cdot H^2 + 8H^3)$
C	$N \cdot H^3$
M	$2k \cdot H^2 + 4H^3 + k \cdot m \cdot H + 2m \cdot H^2 + m^2 \cdot H$
R	$2k \cdot H^3$

where p is the proportion of the part that could run in parallel, and $(1-p)$ is the part remains serial. To determine the speedup limit, we provide an analysis of the complexity of our algorithm by counting the basic arithmetic operations. Assume that the matrix multiplication takes cubic time, the inverse uses Gaussian elimination, there is no overhead in synchronization, and there is no memory contention. Table 6.2 lists a rough estimate of the number of basic arithmetic operations in the *Cut* and *Stitch* steps with E, C, M, and R sub steps. As we mentioned in Section 6.3, the E,C,R sub steps can run on k processors in parallel, while the M step in principle, has to be performed serially on a single processor (or up to four processors with a finer breakdown of the computation).

In our experiment, T is around 100-500, $m = 93$, $H = 15$, thus p is approximately $99.81\% \sim 99.96\%$.

Figure 6.5 shows the wall clock time and speedup on the supercomputer with a maximum of 128 processors. Figure 6.6 shows the wall clock time and speedup on the multi-core desktop (maximum 4 cores). We also include the theoretical limit from Amdahl's law. Table 6.1 lists the running time on the motion set. In order to compute the average running time, we normalized the wall clock time relative to the serial one, defined as

$$t_{norm} = \frac{t_k}{t_1} = \frac{1}{S_k}$$

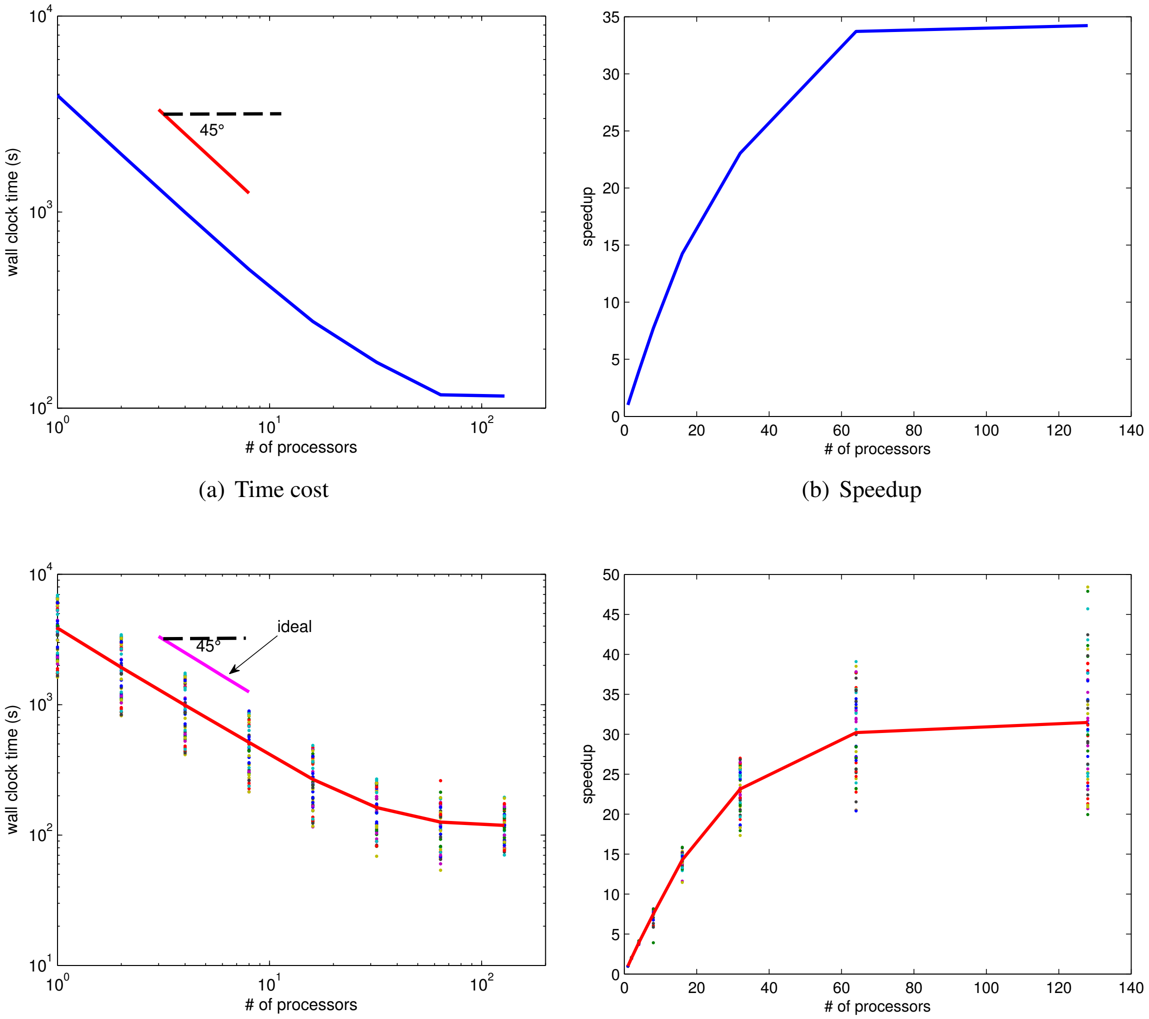
where t_k is wall clock time with k processors.

The performance results show almost linear speedup as we increase the number of processors, which is very promising. Taking a closer look, it is near linear speedup up to 64 processors. The speedup for 128 processors is slightly below linear. A possible explanation is that we may hit the bus bandwidth between processors and memory, and the synchronization overhead increases dramatically with a hundred processors.

6.5.3 Quality

In order to evaluate the quality of our parallel algorithm, we run our algorithm on a different number of processors and compare the error against the serial version (EM algorithm on single processor). Due to the non-identifiability problem, the model parameters for different run might be different, thus we could not directly compute the error on the model parameters. Since both the serial EM learning algorithm and the parallel one tries to maximize the data log-likelihood, we define the error as the relative difference between log-likelihood of the two, where data log-likelihood is computed from the E step of the EM algorithm.

$$error_k = \frac{l(\mathcal{Y}; \hat{\theta}_1) - l(\mathcal{Y}; \hat{\theta}_k)}{l(\mathcal{Y}; \hat{\theta}_1)} \times 100\%$$



(a) Time cost

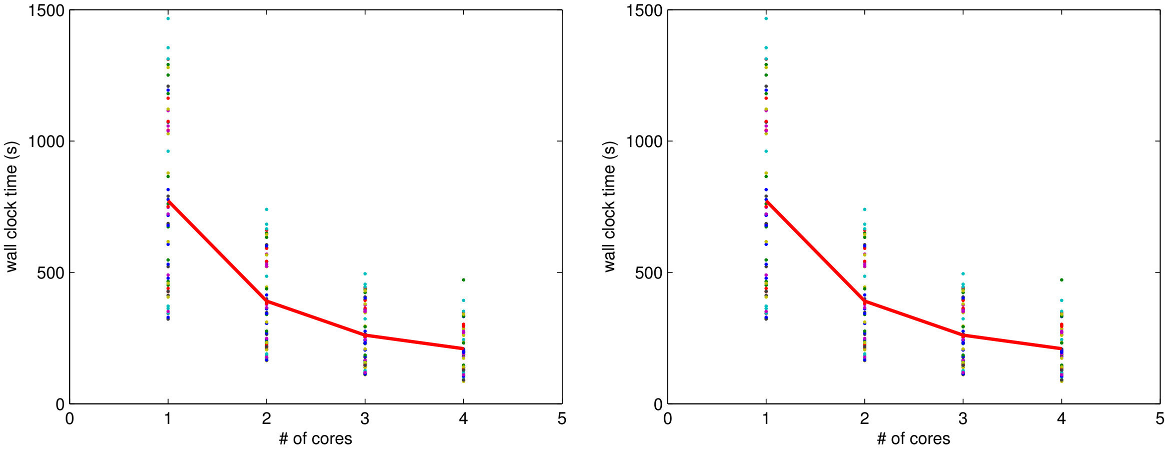
(b) Speedup

(c) Average time cost

(d) Average speedup

Figure 6.5: Performance of Cut-And-Stitch on multi-processor supercomputer, running on the 58 motions. The Sequential version is on one processor, identical to the EM algorithm. (a) Running time for a sample motion (subject 16 #22, walking, 307 frames) in log-log scales; (b) Speedup for walking motion(subject 16 #22) compared with the sequential algorithm; (c) Average running time (red line) for all motions in log-log scales. (d) Average speedup for all motions, versus number of processors k.

where $\mathcal{Y}$ is the motion data sequence, $\hat{\theta}_k$ are parameters learned with k processors and $l(\cdot)$ is the log-likelihood function. The error from the experiments is very tiny, with a maximum 0.3% and mean 0.17%, and no clear evidence of increasing error with more processors. In some cases, the parallel algorithm even found higher (0.074%) likelihood than the serial EM. Note there are limitations of the log-likelihood criteria, namely higher likelihood does not necessarily indicate better fitting, since it might get over-fitting. The error curve shows the quality of parallel is almost identical to the serial one.



(a) Time cost

(b) Average speedup

Figure 6.6: Performance of Cut-And-Stitch on multi-core desktop, running on the 58 motions. The Sequential version is on one processor, identical to the EM algorithm. (a) running time for all motions in log-log scales; (b) average speedup for the 58 motions, versus number of cores k.

6.5.4 Case study

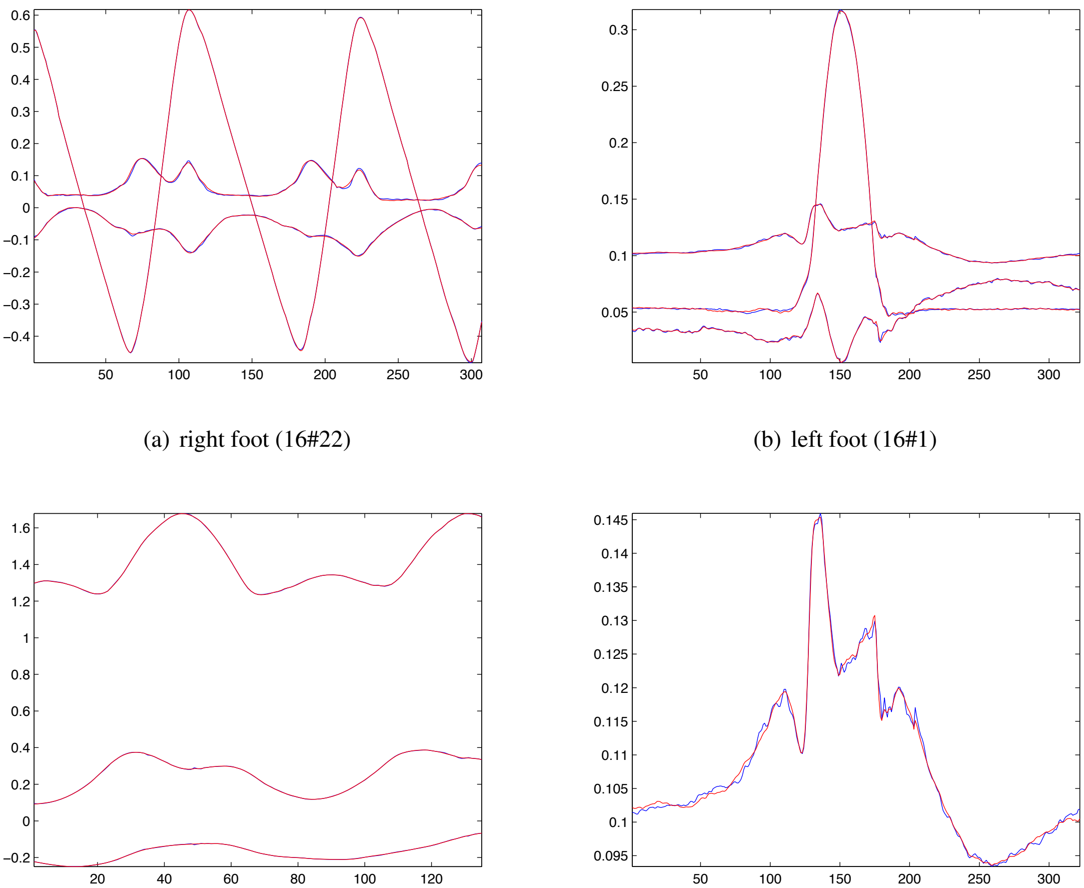
In order to show the visual quality of the parallel learning algorithm, we observe a case study on two different sample motions: walking motion (Subject 16 #22, with 307 frames), jumping motion (Subject 16 #1, with 322 frames), and running motion (Subject 16 #45, with 135 frames). We run the Cut-And-Stitch algorithm with 4 cores to learn model parameters on the multi-core machine, and then use these parameters to estimate the hidden states and reconstruct the original motion sequence. The test criteria is the reconstruction error (NRE) normalized to the variance, defined as

$$NRE = \sqrt{\frac{\sum_{i=1}^{N} ||y_i - \hat{y}_i||^2}{\sum_{i=1}^{N} ||y_i - \sum_{j=1}^{N} y_j/N||^2}} \times 100\%$$

where y_i is the observation for i-th frame and $\hat{y}_i$ is the reconstructed with model parameters from 4-core computation. Table 6.3 shows the reconstruction error: both parallel and serial achieve very small error and are similar to each other. Figure 6.7 and Figure 6.8 show the reconstructed sequences of the feet coordinates. Note our reconstruction (red lines) is very close to the original signal (blue lines).

Table 6.3: Normalized Reconstruction Error

method	Walking	Jumping	Running
Serial	1.929%	1.139%	0.988%
Parallel(4-core)	1.926%	1.140%	0.985%



(a) right foot (16#22)

(b) left foot (16#1)

(c) right foot (16#45)

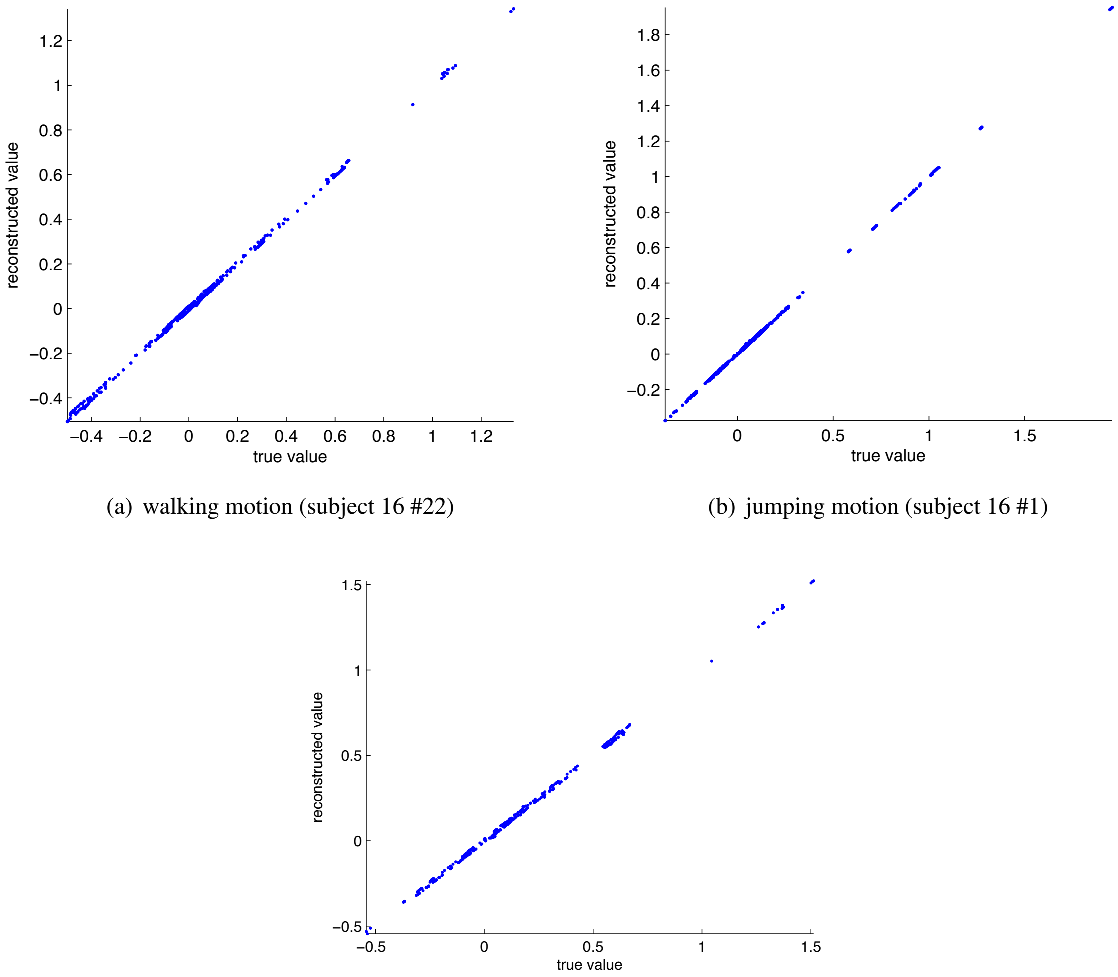
(d) left foot, x (16#1)

Figure 6.7: Visual effects: the reconstructed x, y, z coordinates using learned parameters on 4 cores. Horizontal axis is frame index (time tick). (a) right foot coordinates (x,y,z) for the walking motion (subject 16 #22). (b) left foot coordinates for the jumping motion (subject 16 #1). (c) right foot coordinates for the running motion (subject 16 #45). (d) magnification of the x coordinate (the upper curve in (b)). Note that the reconstructed sequences (red lines) are so close to the original signals (blue lines), that the plots looks like a set of purple lines; this illustrates the high accuracy of Cut-And-Stitch.

6.6 Summary

In this chapter, we explore the problem of parallelizing the learning algorithm for Linear Dynamical Systems (LDS) on symmetric multiprocessor architectures. The main contributions are as follows:

- We propose an approximate parallel learning algorithm for Linear Dynamic Systems, and implement it using the OpenMP API on shared memory machines.
- We performed experiments on a large collection of 58×93 real motion capture sequences spanning 17 MB. CAS-LDS showed near-linear speedup on typical settings (a commercial multi-core desktop, as well as a super computer). We showed that our reconstruction error is almost identical to the



(a) walking motion (subject 16 #22)

(b) jumping motion (subject 16 #1)

(c) running motion (subject 16 #45)

Figure 6.8: Scatter plot: reconstructed value versus true value. For clarity, we only show the 500 *worst* reconstructions - even then, the points are very close on the 'ideal', 45 degree line.

serial algorithm.

Next chapter will discuss the extension to models with similar chain structure such as HMMs.

Chapter 7

Parallelizing Learning Hidden Markov Models

7.1 Introduction

Markov chain models are often used in capturing the temporal behavior of a system. Hidden Markov chain models are those with random variables unobserved in Markov chains. Both linear dynamical systems (LDS) and hidden Markov models (HMM) fall into this framework. The chain of hidden variables, for example, could represent the (unknown) functions of a genetic sequences modeled by HMM, or velocities and accelerations of rockets by Kalman filters. In the following, we will first introduce the general framework of hidden Markov chain models, and describe traditional algorithms for learning those models respectively. Table 3.1 lists the symbols and annotations used in both LDS and HMM.

In hidden Markov chain models, a sequence of observations $\mathbf{Y}$ $(= \vec{y}_1, \ldots, \vec{y}_{\mathrm{T}})$ are drawn from an emission probability distribution $P(\vec{y}_n|\vec{z}_n)$, and hidden variables $\vec{z}_n$ from a Markov chain with the transition

Table 7.1: Symbols and annotations for HMM

Symbol	Definition
$\mathbf{Y}$	observation sequence $(= \{\vec{y}_1, \ldots, \vec{y}_{\mathrm{T}}\})$
$\mathbf{Z}$	the hidden variables $(= \{\vec{z}_1, \ldots, \vec{z}_{\mathrm{T}}\})$
T	the duration of the observation
$\mathbf{M}$	number of discrete values of observation variables
$\mathbf{V}$	possible values for observation variables $(= \{\vec{v}_1, \ldots, \vec{v}_M\})$
$\mathbf{K}$	number of discrete values of hidden variables
$\mathbf{S}$	possible values for hidden variables $(= \{\vec{s}_1, \ldots, \vec{s}_K\})$
$\mathbf{A}$	the transition matrix, $K \times K$
$\mathbf{B}$	the project matrix from hidden to observation, $K \times M$
Π	the initialization vector, $(= \{\pi_1, \ldots, \pi_K\})$

probability distribution $P(\vec{z}_{n+1})|\vec{z}_n$. The joint pdf of the model is as follows:

$$P(\mathbf{Y}, \mathbf{Z}) = P(\vec{z}_1) \prod_{n=2}^{\mathrm{T}} P(\vec{z}_{n+1}|\vec{z}_n) \prod_{n=1} P(\vec{y}_n|\vec{z}_n) \tag{7.1}$$

7.1.1 Hidden Markov Model

Hidden Markov Model (HMM) shares the same graphical model as LDS. However, hidden variables $\mathcal{Z}$ for HMM are discrete and the transitions between them follow the multinomial distributions. The observation $\mathcal{Y}$ can be either discrete or continuous. We will describe the discrete case, however, the learning algorithm is similar.

Assume each observation variable $\vec{y}_n$ of a Hidden Markov Model has M possible values $(v_1, v_2, ..., v_M)$, each hidden variable $\vec{z}_n$ has K possible values $(s_1, s_2, ..., s_K)$. Then parameter set of the HMM λ includes the transition matrix $\mathbf{A}_{pq}$ $(K \times K)$, observation matrix $\mathbf{B}_p(v_r)$ $(K \times M)$ and initialization vector π_i (K $\times$ 1). The data in the model flows according to the subsequent equations:

$$P(\vec{z}_1 = s_p) = \pi_p \tag{7.2}$$

$$P(\vec{z}_n = s_q|\vec{z}_{n-1} = s_p) = \mathbf{A}_{pq} \tag{7.3}$$

$$P(\vec{y}_n = v_r|\vec{z}_n = s_p) = \mathbf{B}_p(v_r) \tag{7.4}$$

The training problem for HMM is as follows: given observations $\mathcal{Y}$, find an optimal λ that maximize the data likelihood. With no tractable direct solution, Training problem can be solved by an EM algorithm as well, which specifically is known as the Baum-Welch algorithm [Baum et al., 1970].

7.2 Cut-And-Stitch for HMM

In order to solve the training problem in Hidden Markov Model, we used EM algorithm to iteratively update the parameter set λ. At each iteration, we calculate $\alpha_n(p)$ from $\vec{z}_1$ to $\vec{z}_N$ (forward computation), and $\beta_n(p)$ from $\vec{z}_N$ back to $\vec{z}_1$ (backward computation). We also define auxiliary variables $\gamma_n(p)$ and $\xi_n(p, q)$ to help us update the HMM model. The definitions of α, β, γ and ξ are shown from Eq (7.5-7.8).

$$\alpha_n(p) = P(\vec{y}_1, ...\vec{y}_n, \vec{z}_n = s_p|\lambda) \tag{7.5}$$

$$\beta_n(p) = P(\vec{y}_{n+1}, ..., \vec{y}_N|\vec{z}_n = s_p, \lambda) \tag{7.6}$$

$$\gamma_n(p) = P(\vec{z}_n = s_p|\vec{y}_1, ..., \vec{y}_N, \lambda) \tag{7.7}$$

$$\xi_n(p, q) = P(\vec{z}_n = s_p, \vec{z}_{n+1} = s_q|\vec{y}_1, ..., \vec{y}_N, \lambda) \tag{7.8}$$

In the *Cut* step with k processors, we again split the Hidden Markov Model into k blocks: $B_1, ..., B_k$. We still use the notation $\vec{z}_{i,j}$ and $\vec{y}_{i,j}$ to indicate j-th variable in the i-th block. Same notations also apply to other intermediate variables like $\alpha_{i,j}(p)$. In order to propagate information between adjacent blocks, we define two set of parameters $\delta_i(p)$ and $\kappa_i(p)$ for each block B_i, where:

$$\delta_i(p) = \alpha_{i-1,T}(p) \tag{7.9}$$
$$\kappa_i(p) = \beta_{i+1,1}(p) \tag{7.10}$$

Then local HMM blocks can update themselves according to Eq (7.11-7.19).

$$\alpha_{1,1}(p) = \pi_p \mathbf{B}_p(\vec{y}_{1,1}) \tag{7.11}$$
$$\alpha_{i,1}(p) = \mathbf{B}_p(\vec{y}_{i,1})\Sigma_{q=1}^{K}\delta_i(q)\mathbf{A}_{qp} \tag{7.12}$$
$$\alpha_{i,j}(p) = \mathbf{B}_p(\vec{y}_{i,j})\Sigma_{q=1}^{K}\alpha_{i,j-1}(q)\mathbf{A}_{qp} \tag{7.13}$$
$$\beta_{k,T}(p) = 1 \tag{7.14}$$
$$\beta_{i,T}(p) = \Sigma_{q=1}^{K}\kappa_i(q)\mathbf{A}_{pq}\mathbf{B}_q(\vec{y}_{i+1,1}) \tag{7.15}$$
$$\beta_{i,j}(p) = \Sigma_{q=1}^{K}\beta_{i,j+1}(q)\mathbf{A}_{pq}\mathbf{B}_q(\vec{y}_{i,j+1}) \tag{7.16}$$
$$\gamma_{i,j}(p) = \frac{\alpha_{i,j}(p)\beta_{i,j}(p)}{\Sigma_{q=1}^{K}\alpha_{i,j}(q)\beta_{i,j}(q)} \tag{7.17}$$
$$\xi_{i,j}(p,q) = \frac{\gamma_{i,j}(p)\mathbf{A}_{pq}\mathbf{B}_q(\vec{y}_{i,j+1})\beta_{i,j+1}(q)}{\beta_{i,j}(p)} \tag{7.18}$$
$$\xi_{i,T}(p,q) = \frac{\gamma_{i,T}(p)\mathbf{A}_{pq}\mathbf{B}_q(\vec{y}_{i+1,1})\kappa_i(q)}{\beta_{i,T}(p)} \tag{7.19}$$

In the *Stitch* step, each block $\mathbf{B}_i$ first collect necessary statistics:

$$\tau_i(p,q) = \Sigma_{j=1}^{T}\xi_{i,j}(p,q) \tag{7.20}$$
$$\zeta_i(p,q) = \Sigma_{l=1}^{K}\Sigma_{j=1}^{T}\xi_{i,j}(p,l) \tag{7.21}$$
$$\eta_i(p,v_r) = \Sigma_{j=1,\vec{y}_{i,j}=v_r}^{T}\gamma_{i,j}(p) \tag{7.22}$$
$$\varphi_i(p,v_r) = \Sigma_{j=1}^{T}\gamma_{i,j}(p) \tag{7.23}$$

Except for the last block:

$$\tau_k(p,q) \quad = \quad \Sigma_{j=1}^{T-1}\xi_{k,j}(p,q) \tag{7.24}$$
$$\zeta_k(p,q) \quad = \quad \Sigma_{l=1}^{K}\Sigma_{j=1}^{T-1}\xi_{k,j}(p,l) \tag{7.25}$$

Subsequently, all blocks work together to update HMM model λ at Eq (7.26-7.28). δ_i and κ_i are also updated here according to Eq (7.9-7.10).

$$\pi_p^{new} \quad = \quad \gamma_{1,1}(p) \tag{7.26}$$
$$\mathbf{A}_{pq}^{new} \quad = \quad \frac{\Sigma_{i=1}^{k}\tau_i(p,q)}{\Sigma_{i=1}^{k}\zeta_i(p,q)} \tag{7.27}$$
$$\mathbf{B}_p(v_r)^{new} \quad = \quad \frac{\Sigma_{i=1}^{k}\eta_i(p,v_r)}{\Sigma_{i=1}^{k}\varphi_i(p,v_r)} \tag{7.28}$$

7.2.1 Warm-Up Step

In the first iteration of the algorithm, there are undefined initial values of block parameters υ,Φ,η and Ψ, needed by the forward and backward propagations in *Cut*. A simple approach would be to assign random initial values, but this may lead to poor performance. We propose and use an alternative method: we run a sequential forward-backward pass on the whole observation, estimate parameters, i.e. we execute the *Cut* step with one processor, and the *Stitch* step with k processors. After that, we begin normal iterations of CAS-HMM with k processors. We refer to this step as the *warm-up* step. Although we sacrifice some speedup, the resulting method converges faster and is more accurate. Figure 7.1 illustrates the time line of the whole algorithm on four CPUs.

In summary, the CAS-HMM algorithms (CAS-LDS and CAS-HMM) work in the following two steps, which could be further divided into four sub-steps:

Cut divides and builds small sub-models (blocks), and then each processor *estimate* (E) in parallel posterior marginal distribution in Eq (7.11-7.19), which includes *forward* and *backward* propagation of beliefs.

Stitch estimates the parameters through *collecting* (C) local statistics of hidden variables in each block Eq (6.29-6.31)and Eq (7.20-7.25), taking the *maximization* (M) of the expected log-likelihood over the parameters Eq (6.32-6.37) and Eq (7.26-7.28), and connecting the blocks by *re-estimate* (R) the block parameters Eq (6.38-6.43) and Eq (7.9-7.10).

7.3 Evaluation

o evaluate the effectiveness and usefulness of our proposed CAS-HMM method in practical applications, we tested our implementation on SMPs. Our goal is to answer the following questions:

- Speedup: how would the performance change as the number of processors/cores increase?
- Quality: while the parallel algorithm is faster than serial algorithm, are we giving up any precision on derived model?

We will first describe the experimental setup and the dataset we used.

7.3.1 Dataset and Experimental Setup

We run the experiments on a variety of typical SMPs, two supercomputers and a commercial desktop.

M1 The first supercomputer is an SGI Altix system[1], at National Center for Supercomputing Applications (NCSA). The cluster consists of 512 1.6GHz Itanium2 processors, 3TB of total memory and 9MB of L3 cache per processor. It is configured with an Intel C++ compiler supporting OpenMP. We use this supercomputer to test our LDS algorithm.

M2 The first supercomputer is an SGI Altix system[2], at Pittsburgh Supercomputing Center (PSC). The cluster consists of 384 1.66GHz Itanium2 Montvale 9130M dual-core processors (a total of 768

[1] cobalt.ncsa.uiuc.edu

[2] http://www.psc.edu/machines/sgi/altix/pople.php

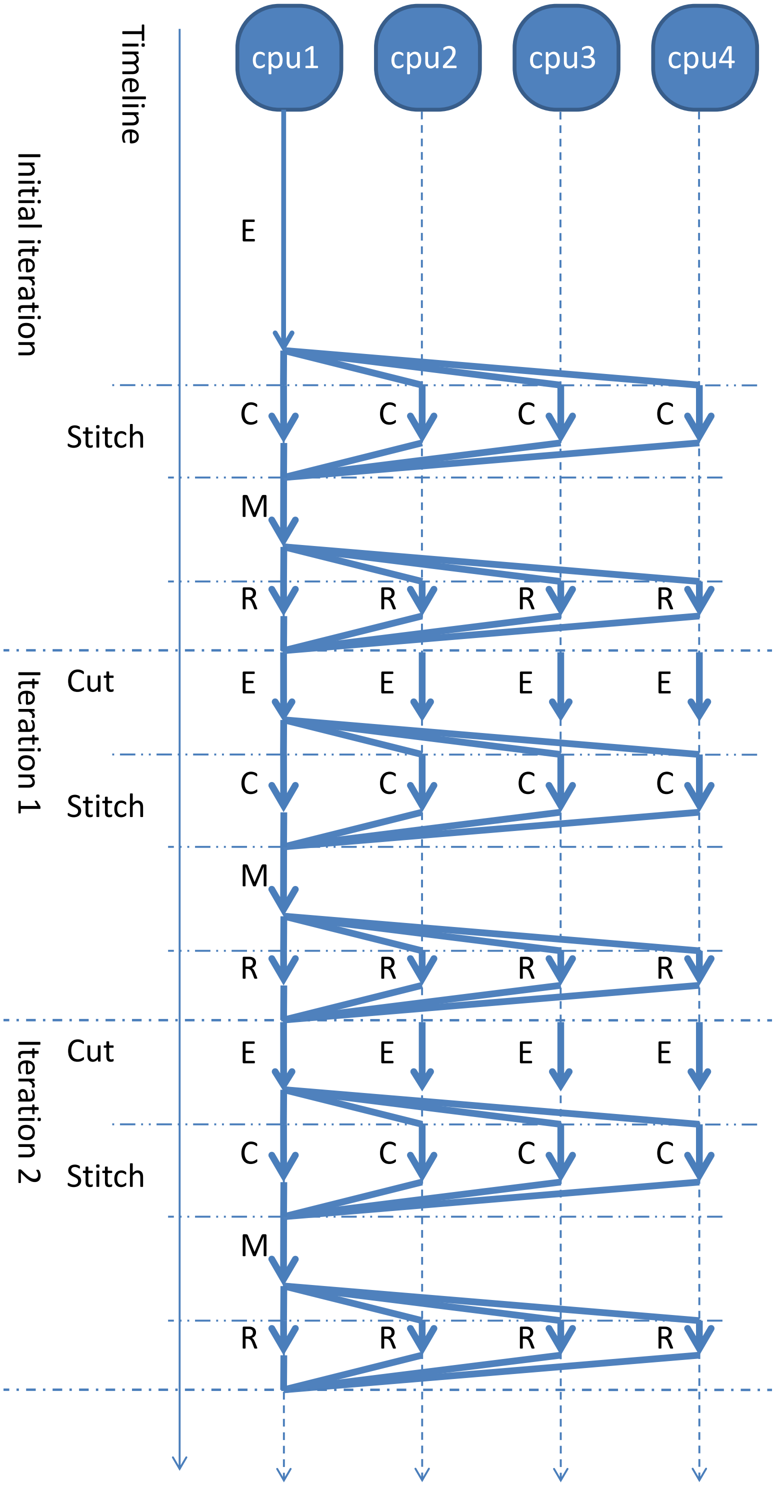


Figure 7.1: Graphical illustration of CAS-HMM algorithm on 4 CPUs. The workflow is same as Figure 6.4. Arrows indicates the computation on each CPU. Tilting lines indicate the necessary synchronization and data transfer between the CPUs and main memory. Tasks labeled with "E" indicate the (parallel) *estimation* of the posterior marginal distribution, including the forward-backward propagation of beliefs within each block as shown in Figure 6.1. (C) indicates the *collection* of local statistics of the hidden variables in each block; (M) indicates the *maximization* of the expected log-likelihood over the parameters, and then it *re-estimates* (R) the block parameters.

Table 7.2: Count of arithmetic operations $(+, -, \times, /)$ in E, C, M, R sub steps of CAS-HMM in HMM. Each type of operation is equally weighted, and only the largest portions in each step are kept.

	#of operations of HMM
E	$9 \cdot N \cdot K^2$
C	$2K \cdot N \cdot (K+M)$
M	$2k \cdot K \cdot (K+M)$
R	$4k \cdot K^2$

cores), 1.5TB of total memory and 8MB of L3 cache per processor. It is configured with SuSE Linux and Intel compiler. We use this supercomputer to test our HMM algorithm.

M3 The test desktop machine has two Intel Xeon dual-core 3.0GHz CPUs (a total of four cores), 16G memory, running Linux (Fedora Core 7) and GCC 4.1.2 (supporting OpenMP). We use this supercomputer to test both of our LDS and HMM algorithm.

For HMM, we used a synthetic dataset, with the observation sequences randomly generated. The data has $K = 100$ hidden states and $R = 50$ different observation values. The duration of the sequence is N=1536.

7.3.2 Speedup

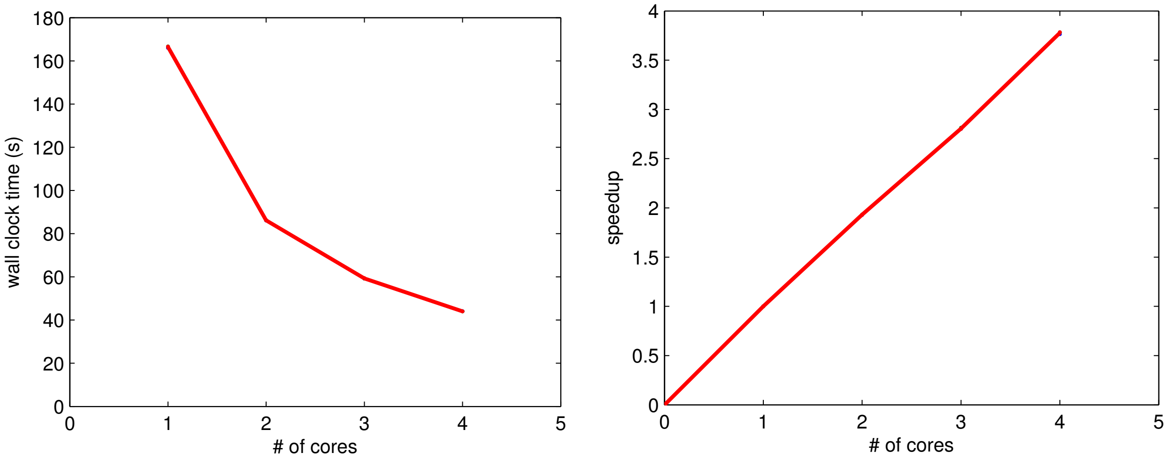
The algorithmic complexity of our parallel HMM implementation is shown in Table 7.2. Figure 7.2 and Figure 7.3 show the wall clock time and speedup on multi-core desktop and PSC supercomputer respectively. Comparing to LDS with similar model size, each iteration of HMM implementation would take much less time, so the overhead of parallel framework stands out earlier: the speedup for HMM starts to getting less impressive when we use about 16 processors.

However, for some Hidden Markov Model applications such as bioinformatics, sequence length (N) could be much larger: for example, each DNA sequence might contain thousands, millions or even more *b*ase pairs(pairs of nucleotides A,T,G,C). We envision that our CAS-HMM method would exhibit better speedup towards those problems.

7.3.3 Quality

In order to evaluate the quality of our parallel algorithm, we run our algorithm on a different number of processors and compare the error against the serial version (EM algorithm on single processor). Due to the non-identifiability problem, the model parameters for different run might be different, thus we could not directly compute the error on the model parameters. Since both the serial EM learning algorithm and the parallel one tries to maximize the data log-likelihood, we define the error as the relative difference between log-likelihood of the two, where data log-likelihood is computed from the E step of the EM algorithm.

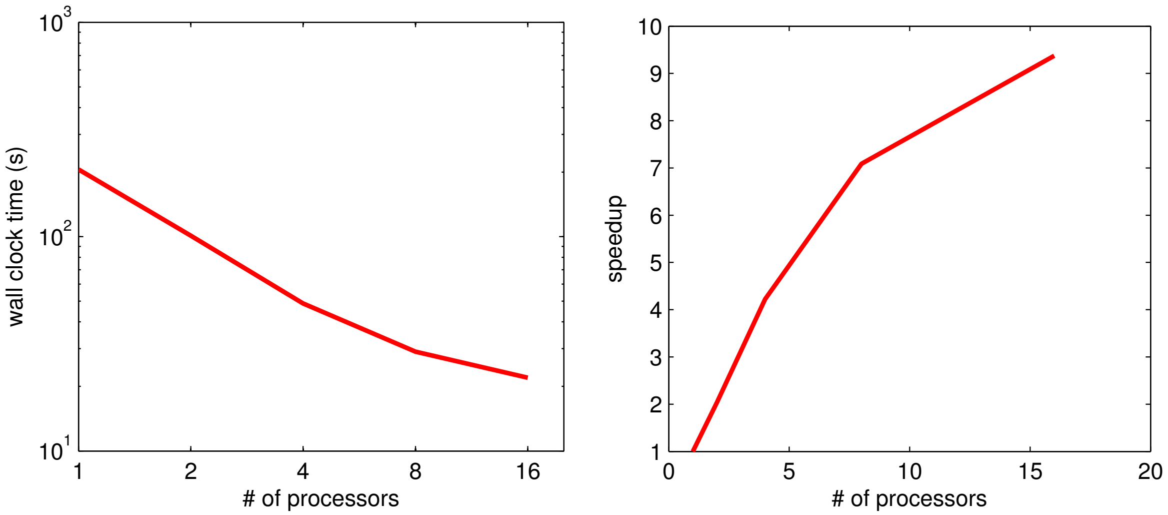
$$error_k = \frac{l(\mathcal{Y};\hat{\theta}_1) - l(\mathcal{Y};\hat{\theta}_k)}{l(\mathcal{Y};\hat{\theta}_1)} \times 100\%$$



(a) Running time

(b) Speedup

Figure 7.2: Performance of CAS-HMM on multi-core desktop.



(a) Running time

(b) Speedup

Figure 7.3: Performance of CAS-HMM on PSC supercomputer.

where $\mathcal{Y}$ is the motion data sequence, $\hat{\theta}_k$ are parameters learned with k processors and $l(\cdot)$ is the log-likelihood function. The error to both of our LDS and HMM experiments are very tiny: error of LDS algorithm has a maximum of 0.5% and mean of 0.17%; error of HMM algorithm has a maximum of 1.7% and mean of 1.2%. Furthermore, there is no clear positive correlation between error and the number of processors. In some cases, the parallel algorithm even found higher (0.074%) likelihood than the serial algorithm. Note there are limitations of the log-likelihood criteria, namely higher likelihood does not necessarily indicate better fitting, since it might get over-fitting. The error curve shows the quality of parallel is almost identical to the serial one.

7.4 Summary

In this chapter, we present a parallel algorithm for learning Hidden Markov Models (HMM) on symmetric multiprocessor architectures. The main contributions are as follows:

- We propose CAS-HMM, an approximate parallel learning algorithm for Hidden Markov Models, and implement it using the OpenMP API on shared memory machines.
- We performed experiments on synthetic datasets. CAS-HMM showed near-linear speedup on typical settings (a commercial multi-core desktop, as well as a super computer). We showed that our reconstruction error is almost identical to the serial algorithm.

Chapter 8

Distributed Algorithms for Mining Web-click Sequences

Given a large stream of users clicking on web sites, how can we find trends, patterns and anomalies? In this chapter, we present a novel method, WindMine, and its fine-tuning sibling, WindMine-part, to find patterns and anomalies in such datasets. Our approach has the following advantages: (a) it is effective in discovering meaningful "building blocks" and patterns such as the lunch-break trend and anomalies, (b) it automatically determines suitable window sizes, and (c) it is fast, with its wall clock time linear on the duration of sequences. Moreover, it can be made sub-quadratic on the number of sequences (WindMine-part), with little loss of accuracy.

In the later part of the chapter, we will examine the effectiveness and scalability by performing experiments on 67 GB of real data (one billion clicks for 30 days). Our proposed WindMine does produce concise, informative and interesting patterns. We also show that WindMine-part can be easily implemented in a parallel or distributed setting, and that, even in a single-machine setting, it can be an order of magnitude faster (up to 70 times) than the plain version.

8.1 Introduction

Many real applications generate log data at different time stamps, such as web click logs and network packet logs. At every time stamp, we might observe a set of logs, each consisting of a set of events, or time stamped tuples. In many applications the logging rate has increased greatly with the advancement of hardware and storage technology. One big challenge when analyzing these logs is to handle such large volumes of data at a very high logging rate. For example, a search web could generate millions of logging entries every minute, with information of users and URLs. As an illustration, we will use the web-click data as a running target scenario, however, our proposed method will work for general datasets as we demonstrate experimentally.

There has been much recent work on summarization and pattern discovery for web-click data. We formulate the web-click data as a collection of time stamped entries, i.e. $\langle$user-id, url, timestamp$\rangle$. The goal is to find anomalies, patterns, and periodicity for such datasets in a systematic and scalable way. Analyzing click sequences can help practitioners in many fields: (a) ISPs would like to undertake provisioning, capacity planning and abuse detection by analyzing historical traffic data; (b) web masters and web-site owners would like to detect intrusions or target designed advertisements by investigating the user-click patterns.

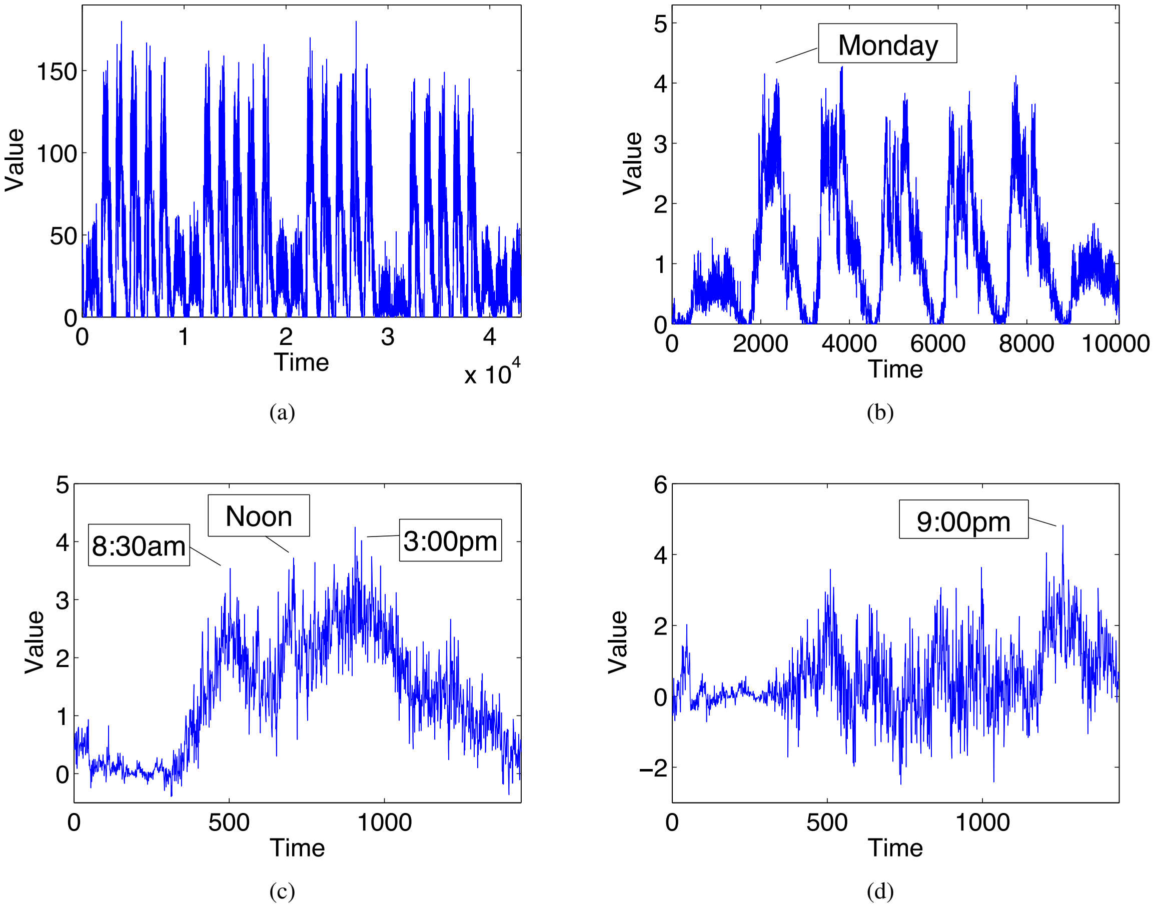


Figure 8.1: Illustration of trend discovery. (a) Original web-click sequence (access count from a business news site). (b) Weekly trend, which shows high activity on weekdays for business purposes. (c) Weekday trend, which increases from morning to night and reaches peaks at 8:30 am, noon, and 3:00 pm. (d) Weekend trend, which is different from the weekday trend pattern.

A common approach to analyzing the web-click tuples is to view them as multiple event sequences, one for each common URL. For example, one event sequence could be $\{\langle Alice, 1\rangle, \langle Bob, 2\rangle, \dots\}$, i.e., Alice hits url$_1$ at time 1 sec., and Bob at 2 sec. Instead of studying this at the individual click level, our approach is designed to find patterns at the aggregation level to allow us to detect common behavior or repeated trends. Formally, these event sequences are aggregated into multiple time series for $m$websites (or URLs). Each of them counts the number of hits (clicks) per $\Delta t = 1$ minute and has an aligned duration of T. Given such a dataset with multiple time series, we would like to develop a method to find interesting patterns and anomalies. For example, the leftmost plot in Figure 8.1 shows the web-click records from a business news site. The desired patterns for this particular data include (a) the adaptive cycles (e.g., at a daily or a weekly level); (b) the spikes in the morning and at lunch time that some sequences exhibit; and (c) the fact that these spikes are only found on week-days. For a few number of sequences (e.g. $m = 5$), a human could eye-ball them, and derive the above patterns. The mining task is even more challenging if the number of sequences increases tremendously. How can we accomplish this task automatically for thousands or even million of sequences? We will show later that our proposed system can automatically identify these patterns all at once, and how it solves scalably.

Contributions

Our main contribution is the proposal of WindMine, which is a novel method for finding patterns in a large collection of click-sequence data. WindMine automatically detects daily periodicity (unsurprisingly), huge lunch-time spikes for news-sites as shown in Figure 8.1 (reasonable, in retrospect), as well as additional, surprising patterns. Additional contributions are as follows:

1. *Effective*: We apply WindMine to several real datasets, spanning 67 GB. Our method finds both expected and surprising patterns, completely on its own.
2. *Adaptive*: We propose a criterion that allows us to choose the best window size of trend patterns from the sequence dataset. The choice of window sizes is data-driven and fully automatic.
3. *Scalable*: The careful design of WindMine makes it linear on the number of time-ticks in terms of wall clock time. In fact, it is readily parallelizable, which means that it can also scale well with a large number of sites m.

The rest of the chpater is organized as follows: Section 8.2 discusses related work. Sections 8.3 presents our proposed method, and Section 8.4 introduces some useful applications of our method, and evaluates our algorithms based on extensive experiments.

8.2 Related Work

There are several pieces of work related to our approach, including (a) dimensionality reduction; (b) time series indexing; (c) pattern/trend discovery and outlier detection.

Dimensionality reduction for time-series data: Singular value decomposition (SVD) and principal component analysis (PCA) [Jolliffe, 2002, Wall et al., 2003] are commonly used tools to discover hidden variables and low rank patterns from high dimensional data. In contrast to the traditional SVD for batch data, Yi et al. [Yi et al., 2000] proposed an online autoregressive model to handle multiple sequences. Gilbert et al. [Gilbert et al., 2001] used wavelets to compress the data into a fixed amount of memory by keeping track of the largest Haar wavelet coefficients. Papadimitriou et al. [Papadimitriou et al., 2005] proposed the SPIRIT method to discover linearly correlated patterns for data streams, where the main idea was to calculate the SVD incrementally. Sun et al. [Sun et al., 2006a] took a step forward by extending SPIRIT to handle data from distributed sources, so that each node or sensor device could calculate local patterns/hidden variables, which are then summarized later at a center node. Our proposed WindMine is related to a scheme for partitioning the data, calculating them individually and finally integrating them in the end. Moreover, our method could detect patterns and anomalies even more effectively.

Indexing and representation: Our work is also related to the theories and methods for time-series representation [Mehta et al., 2006, Lin et al., 2003, Shieh and Keogh, 2008], and indexing [Keogh, 2002, Sakurai et al., 2005b, Keogh et al., 2004, Fujiwara et al., 2008]. Various methods have been proposed for representing time-series data using shapes, including velocity and shape information for segmenting trajectory [Mehta et al., 2006]; symbolic aggregate approximation (SAX) [Lin et al., 2003] and its generalized version for indexing massive amounts of data (iSAX) [Shieh and Keogh, 2008]. Keogh [Keogh, 2002] proposed a search method for dynamic time warping (DTW). [Sakurai et al., 2005b] proposed the FTW method with successive approximations, refinements and additional optimizations, to accelerate "whole sequence" matching under the DTW distance. Keogh et al. used uniform scaling to create an index for large human motion databases [Keogh et al., 2004]. [Fujiwara et al., 2008] presented SPIRAL, a fast search method for HMM datasets. To reduce the search cost, the method efficiently prunes a significant number of search candidates by applying upper bounding approximations when estimating likelihood. Tensor analysis is yet another tool for modeling multiple streams. Related work includes scalable tensor decomposition [Kolda and Sun, 2008] and incremental tensor analysis [Sun et al., 2006c,b, 2008].

Pattern/trend discovery: Papadimitriou et al. [Papadimitriou and Yu, 2006] proposed an algorithm for discovering optimal local patterns, which concisely describe the multi-scale main trends. [Sakurai et al., 2005a] proposed BRAID, which efficiently finds lag correlations between multiple sequences. SPRING [Sakurai et al., 2007] efficiently and accurately detects similar subsequences without determining window size. Kalman filters are also used in tracking patterns for trajectory and time series data [Tao et al., 2004, Li et al., 2009]. Other remotely related work includes the classification and clustering of time-series data and outlier detection. Gao et al. [Gao et al., 2008] proposed an ensemble model to classify time-series data with skewed class distributions, by undersampling the dominating class and oversampling or repeating the rare class. Lee et al. [Lee et al., 2008] proposed the TRAOD algorithm for identifying outliers in a trajectory database. In their approach, they first partition the trajectories into small segments and then use both distance and density to detect abnormal sub-trajectories. This chapter mainly focuses on web-click mining as an application of our method, thus, our work is also related to topic discovery for web mining. There has been a large body of work on statistical topic models [Hofmann, 1999, Blei et al., 2003, Newman et al., 2006, Wei et al., 2007], which uses a multinomial word distribution to represent a topic. These techniques are also useful for web-click event analysis while our focus is to find local components/trends in multiple numerical sequences.

8.3 WindMine

8.3.1 Problem definition

Web-log data consist of tuples of the form *(user-id, url, timestamp)* . We turn them into sequences X_1, . . ., X_n, one for each URL of interest. We compute the number of hits per Δt= 1 minute (or second), and thus we have n sequences of duration T. One of the sequences, X, is a discrete sequence of numbers { $x_1, \ldots, x_t, \ldots, x_\mathrm{T}$ }, where x_T is the most recent value.

Our goal is to extract the main components of click sequences, to discover common trends, hidden patterns, and anomalies. As well as the components of the entire sequences, we focus on components of length w to capture local trends. We now define the problems we are trying to solve and some fundamental concepts.

Problem 8.1 (Local component analysis)**.** *Given n sequences of duration* T *and window size w, find the subsequence patterns of length w that represent the main components of the sequences.*

The window size w is given in Problem 8.1. However, with real data, w for the component analysis is not typically known in advance. Thus the solution has to handle subsequences of multiple window sizes. This gives rise to an important question: whenever the main components of the 'best' window size are extracted

from the sequences, we expect there to be many other components of multi-scale windows, which could potentially flood the user with useless information. How do we find the best window size automatically? The full problem that we want to solve is as follows:
Problem 8.2 (Choice of best window size)**.** *Given n sequences of duration T, find the best window size w and the subsequence patterns of length w that represent the main components of the sequences.*

An additional question relates to what we can do in the highly likely case that the users need an efficient solution while in practice they require high accuracy. Thus, our final challenge is to present a scalable algorithm for the component analysis.

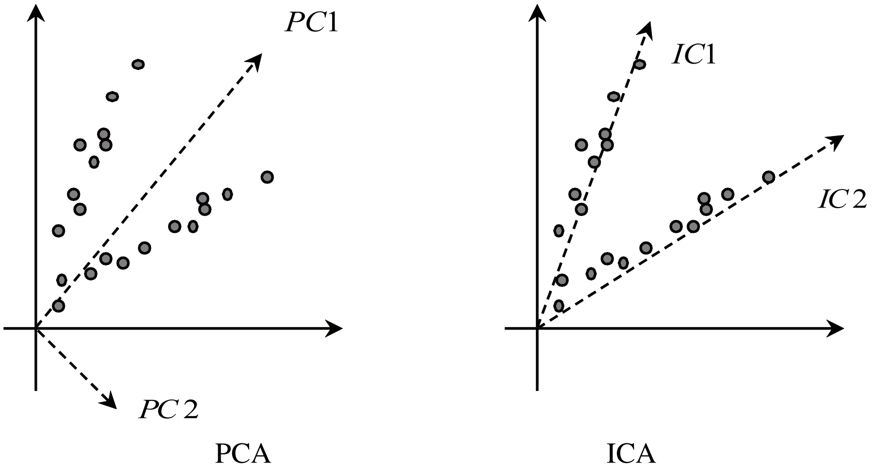
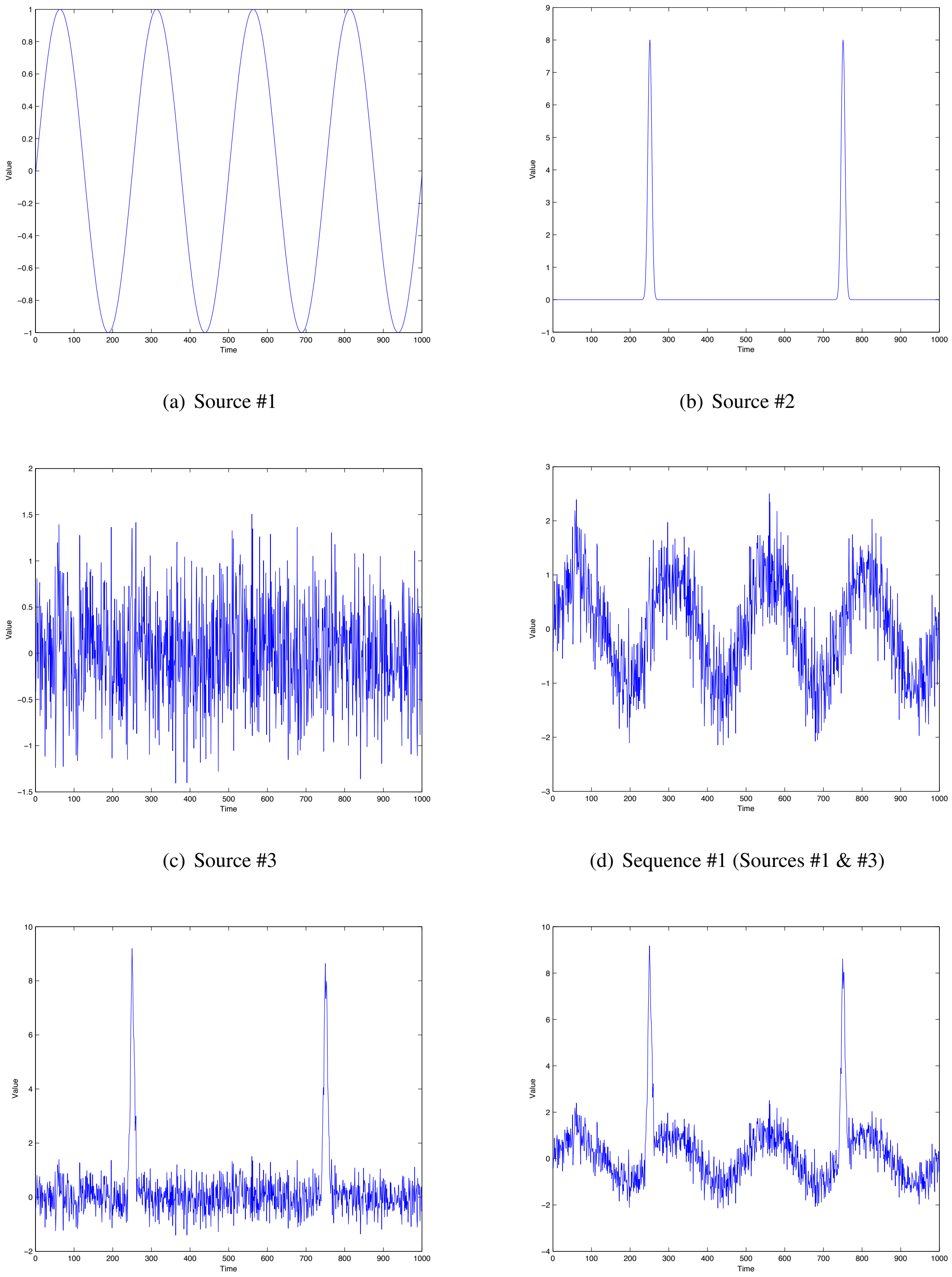


Figure 8.2: PCA versus ICA. Note that PCA vectors go through empty space; ICA/WindMine components snap on the natural lines (leading to sparse encoding).

8.3.2 Multi-scale local component analysis

For a few time sequences, a human could eye-ball them, and derive the above patterns. But, how can we accomplish this automatically for thousands of sequences? The first idea would be to perform principal component analysis (PCA) [Jolliffe, 2002], as employed in [Korn et al., 1997]. However, PCA and singular value decomposition (SVD) have pitfalls. Given a cloud of T-D points (sequences with T time-ticks), PCA will find the best line that goes through that cloud; and then the second best line (orthogonal to the first), and so on. Figure 8.2 highlights this pitfall: If the cloud of points looks like a pair of scissors, then the first principal component will go in the *empty* area marked ”PC1”, and the second principal component will be on the equally empty area that is perpendicular to the first PC.
Approach 8.1. *We introduce independent component analysis (ICA) for data mining of numerical sequences.*

Instead of using PCA, we propose employing ICA [Hyvärinen and Oja, 2000], also known as *blind source separation*. ICA will find the directions marked IC1 and IC2 in Figure 8.2 exactly, because it does not require orthogonality, but needs a stronger condition, namely, independence. Equivalently, this condition results in sparse encoding: the points of our initial cloud will have a sparse representation in the new set of (non-orthogonal) axes, that is, they will have several zeros.
Example 8.1. *Figure 8.3 shows a set of synthetic sequences and Figure 8.4 an example of component analysis. The sample dataset includes three sequences: (1) sinusoidal waves with white noise, (2) large*



(a) Source #1

(b) Source #2

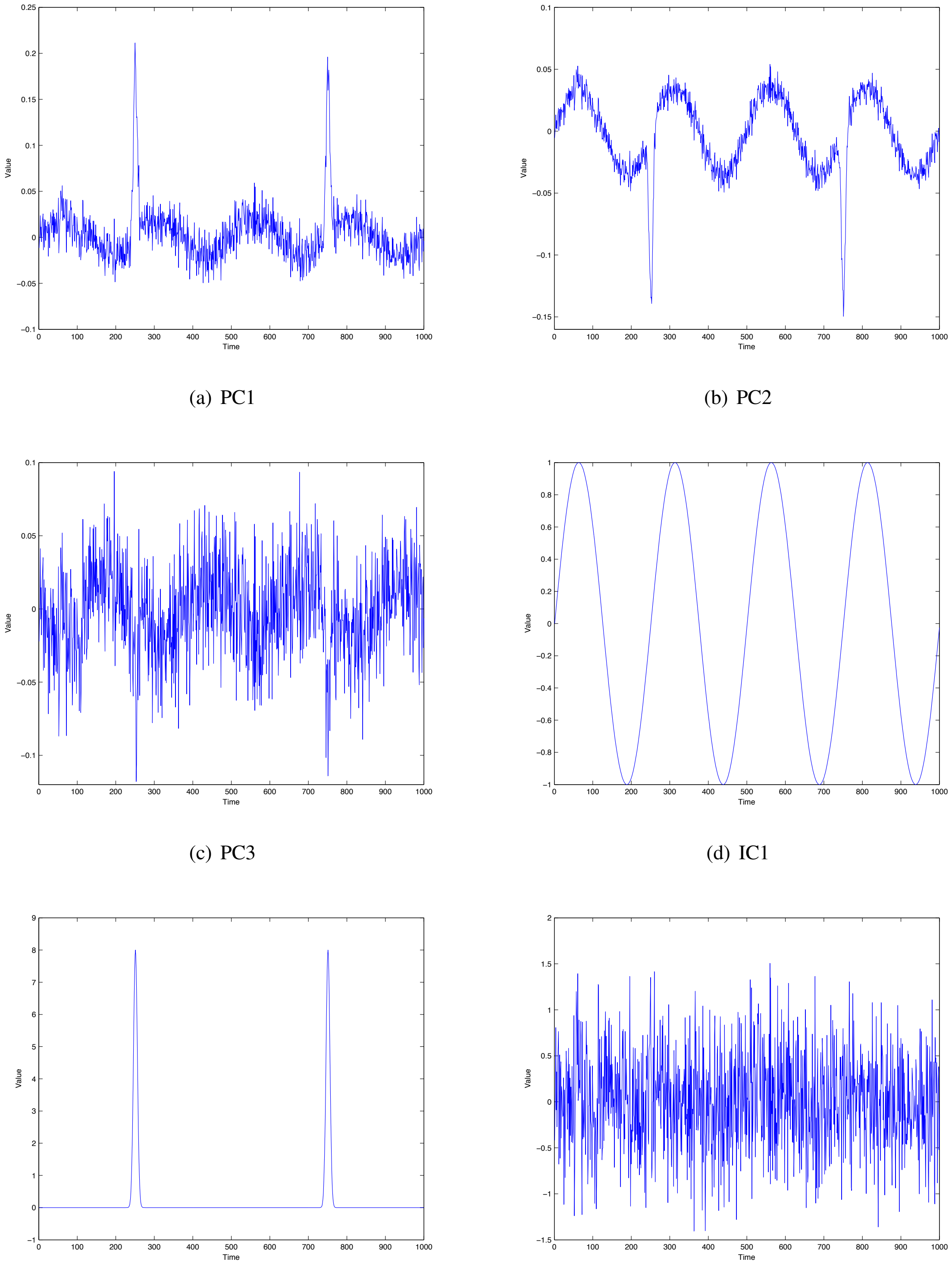
(c) Source #3

(d) Sequence #1 (Sources #1 & #3)

(e) Sequence #2 (Sources #2 & #3)

(f) Sequence #3 (Mix of all 3 sources)

Figure 8.3: An illustrative example. (a), (b) and (c): source signals (basis) to generate the observation sequences; (d), (e) and (f): data sequences that are linear combinations of the three sources.



(a) PC1

(b) PC2

(c) PC3

(d) IC1

(e) IC2

(f) IC3

Figure 8.4: Example of PCA and ICA components for data sequences in Figure 8.3. (a), (b) and (c): the components recovered by PCA; (d), (e) and (f): the components recovered by ICA. Notice how much more clear is the separation of sources that ICA achieves. PCA suffers from the ’PCA confusion’ phenomenon.

spikes with noise, and (3) a combined sequence. We compute three components each for PCA and ICA, from the three original sequences. Unlike PCA, which is confused by these components, ICA recognizes them successfully and separately.

In the preceding discussion we introduced ICA and showed how to analyze entire full length sequences to obtain their 'global' components. We then describe how to find the local components using ICA.
Approach 8.2. *We propose applying a short-window approach to ICA, which is a more powerful and flexible approach for component analysis.*
Definition 8.1 (Window matrix)**.** *Given a sequence $X = \{ x_1, \ldots, x_\mathrm{T} \}$ and a window size w, the window matrix of X, $\hat{X}$, is a $\lceil \mathrm{T}/w \rceil \times w$ matrix, in which the i-th row is $\{ x_{(i-1)w+1}, \ldots, x_{iw} \}$.*

When we have m sequences, we can locally analyze their common independent components using the short-window approach. We propose WindMine for local component analysis.
Definition 8.2 (WindMine)**.** *Given n sequences of duration T, and a window size w, the local independent components are computed from the $M \times w$ window matrix of the n sequences, where $M = n \cdot \lceil \mathrm{T}/w \rceil$.*

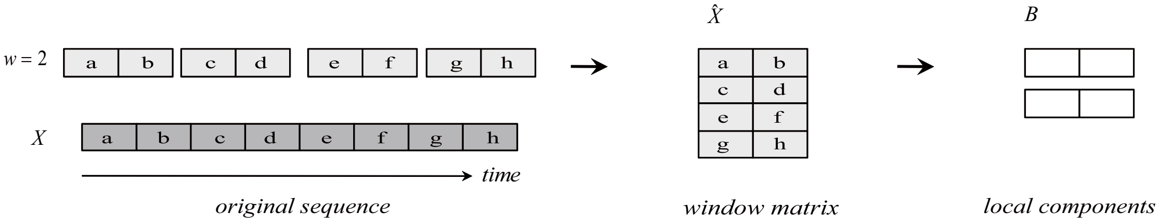


Figure 8.5: Illustration of WindMine for window size $w = 2$. It creates a window matrix of 4 disjoint windows, and then finds their two major trends/components.

The size of the local components typically depends on the given datasets. Our method, WindMine, handles multi-scale windows to analyze the properties of the sequences.
Approach 8.3. *We introduce a framework based on multi-scale windows to discover local components.*

Starting with the original sequences $\{ X_1, \ldots, X_n \}$, we divide each one into subsequences of length w, construct their window matrix $\hat{X}_w$, and then compute the local components from $\hat{X}_w$. We vary the window size w, and repeatedly extract the local components B_w with the mixing matrix A_w for various window sizes w.
Example 8.2. *Figure 8.5 illustrates multi-scale local component analysis using WindMine. The total duration of a sequence X is $n = 8$. We have four disjoint windows each of length $w = 2$, thus $\hat{X}_2$ is a 4×2 matrix. We extract two local components for $w = 2$ in this figure.*

8.3.3 CEM criterion: best window size selection

Thus far, we have assumed that the window size was given. The question we address here is how to estimate a good window size automatically when we have multiple sequences. We would like to obtain a criterion that will operate on the collection of subsequences, and dictate a good number of subsequence length w for the local component analysis. This criterion should exhibit a spike (or dip) at the "correct" value of w. Intuitively, our observation is that if there is a trend of length w that frequently appears in the given sequences, the computed local component is widely used to represent their window matrix $\hat{X}_w$. We want to find the "sweet spot" for w.

We therefore propose using the mixing matrix A_w to compute the criterion for selecting the window size. Notice that straightforward approaches are unsuitable, because they are greatly affected by specific, unpopular components. For example, if we summed up the weight values of each column of the mixing matrix and then chose the component with the highest value, the component would be used to represent a limited number of subsequences. Our goal boils down to the following question: *What function of w reaches an extreme value when we hit the 'optimal' number of window sizes w_{opt}?* It turns out that 'popular' (i.e., widely used) components are suitable for selection as local components that capture the local trends of the sequence set.

Approach 8.4. *We introduce a criterion for window size selection, which we compute from the entropy of the weight values of each component in the mixing matrix.*

We propose a criterion for estimating the optimal number of w for a given sequence set. The idea is to compute the probability histogram of the weight parameters of each component in the mixing matrix, and then compute the entropy of each component.

The details are as follows: For a window size w, we provide the mixing matrix $A_w = [a_{i,j}]$ ($i = 1,\ldots,M$; $j = 1,\ldots,k$) of given sequences, where k is the number of components and M is the number of subsequences. In addition, we normalize the weight values for each subsequence.

$$a'_{i,j} = a_{i,j} / \sum_j a^2_{i,j}. \tag{8.1}$$

We then compute the probability histogram $P_j = \{p_{1,j}, \ldots, p_{M,j}\}$ for the j-th component.

$$p_{i,j} = \|a'_{i,j}\| / \sum_i \|a'_{i,j}\|. \tag{8.2}$$

Intuitively, P_j shows the size of the j-th component's contribution to each subsequence. Since we need the most popular component among k components, we propose using the entropy of the probability histogram for each component.

Therefore, our proposed criterion, which we call component entropy maximization (CEM), or the CEM score, is given by

$$C_{w,j} = -\frac{1}{\sqrt{w}} \sum_i p_{i,j} \log p_{i,j}, \tag{8.3}$$

where $C_{w,j}$ is the CEM score of the j-th component for the window size w. We want the best local component of length w that maximizes the CEM score, that is, $C_w = \max_j C_{w,j}$.

Once we obtain C_w for every window size, the final step is to choose w_{opt}. Thus, we propose

$$w_{opt} = arg \max_w C_w. \tag{8.4}$$

8.3.4 Scalable algorithm: WindMine-part

In this subsection we tackle an important and challenging question, namely how do we efficiently extract the best local component from large sequence sets? In Section 8.3.2 we present our first approach for multi-scale local component analysis. We call this approach WindMine-plain[1].

[1] We use 'WindMine' as a general term for our method and its variants.

Algorithm 8.1: WindMine-part (w, $\{X_1, \ldots, X_n\}$)

```
for each sequence $X_i$ do
    Divide $X_i$ by $\lceil n/w \rceil$ subsequences ;
    Append the subsequences to the window matrix $\hat{X}$;
end
for level $h = 1$ to $H$ do
    Initialize $\hat{X}_{new}$;
    Divide the subsequence set of $\hat{X}$ into $\lceil M/g \rceil$ groups;
    for group number $j = 1$ to $\lceil M/g \rceil$ do
        Create the $j$-th submatrix $S_j$ of $\hat{X}$;
        Compute the local components of $S_j$ with their mixing matrix $A$;
        Compute the CEM score of each component from $A$;
        Append the best local component(s) to $\hat{X}_{new}$;
    end
    $\hat{X}$ = $\hat{X}_{new}$;
    M = $\lceil M/g \rceil$;
end
Report the best local component(s) in $\hat{X}$;
```

Although important, this approach is insufficient to provide scalable processing. What can we do in the highly likely case that the users need an efficient solution for large datasets while in practice they require high accuracy? To reduce the time needed for local component analysis and overcome the scalability problem, we present a new algorithm, WindMine-part.

Approach 8.5. *We introduce a partitioning approach for analyzing a large number of subsequences hierarchically, which yields a dramatic reduction in the computation cost.*

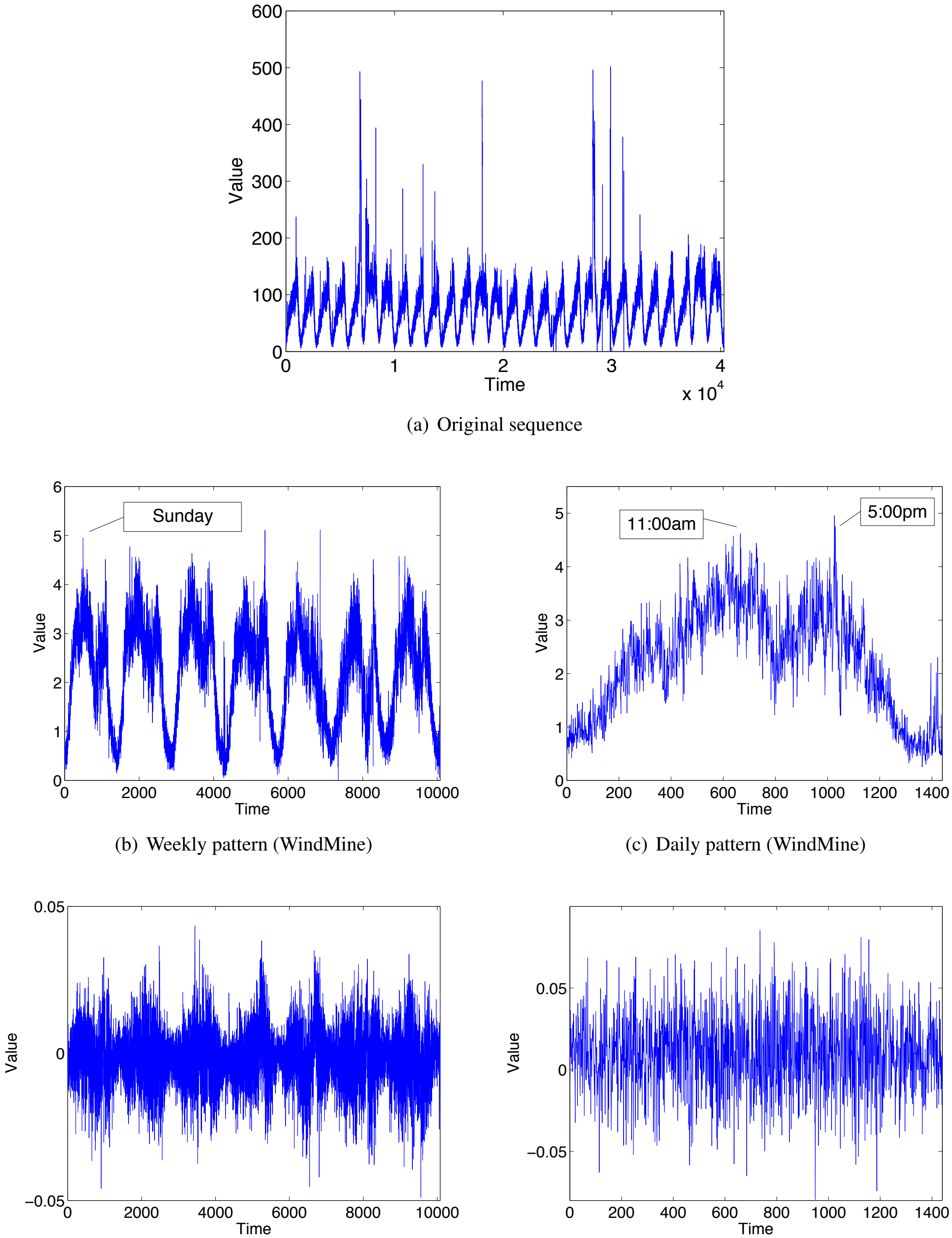
Specifically, instead of computing local components directly from the entire set of subsequences, we propose partitioning the original window matrix into submatrices, and then extracting local components each from the submatrices.

Definition 8.3 (Matrix partitioning)**.** *Given a window matrix $\hat{X}$, and an integer g for partitioning, the j-th submatrix of $\hat{X}$ is formed by taking rows from $(j-1)g+1$ to jg.*

Our partitioning approach is hierarchical, which means that we reuse the local components of the lower level for local component analysis on the current level.

Definition 8.4 (WindMine-part)**.** *Given a window matrix on the h-th level, we extract k local components from each submatrix that has g local components of the $(h-1)$-th level. Thus, the window matrix on the h-th level includes $M \cdot (k/g)^{h-1}$ local components (i.e., $M \cdot (k/g)^{h-1}$ rows).*

After extracting the local components from the original window matrix on the first level $h = 1$, we create a new window matrix from the components of $h = 1$ on the second level ($h = 2$), and then compute the local components of $h = 2$. We repeatedly iterate this procedure for the upper levels. Algorithm 8.1 provides a high-level description of WindMine-part.



(a) Original sequence

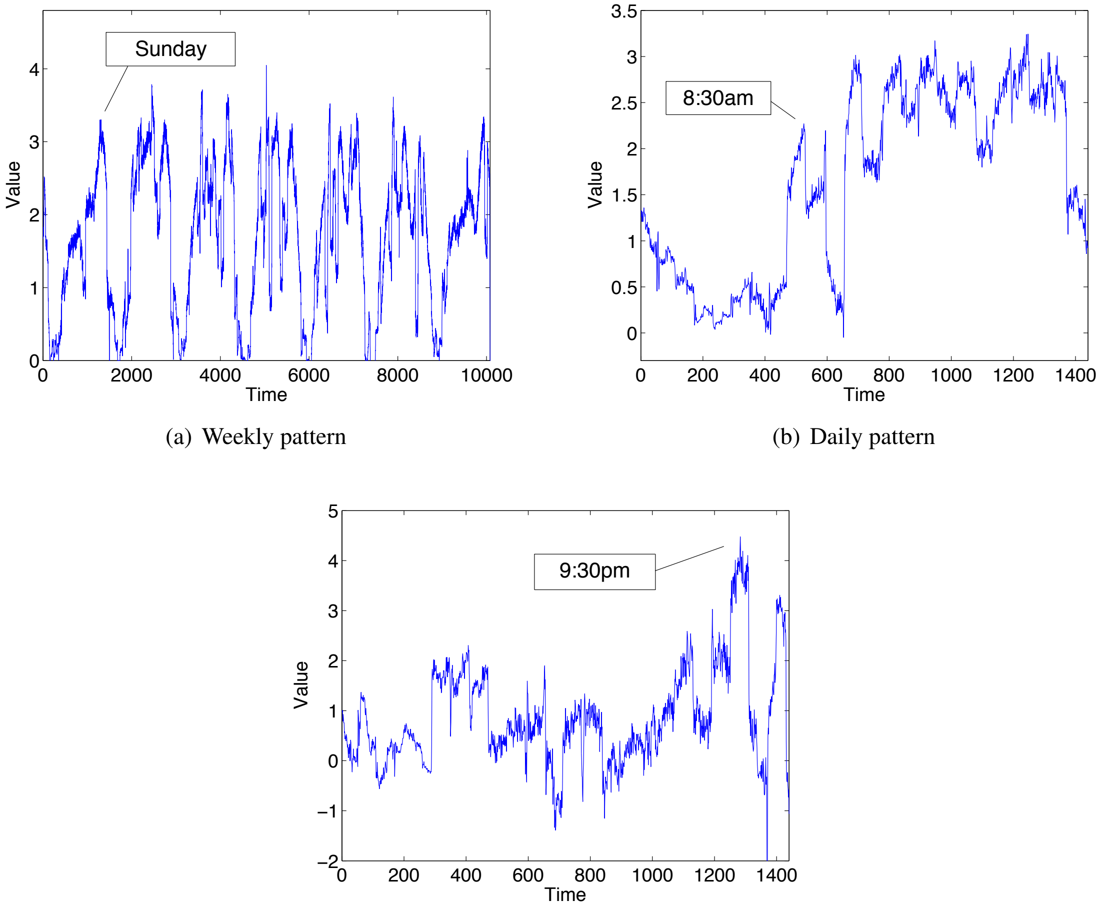
(b) Weekly pattern (WindMine)

(c) Daily pattern (WindMine)

(d) Weekly pattern (PCA)

(e) Weekly pattern (PCA)

Figure 8.6: Original sequence and weekly and daily components for `Ondemand TV`. (b) Note the daily periodicity, with NO distinction between weekdays and weekends. (c) The main daily pattern agrees with our intuition: peaks in the morning and larger peaks in the evening, with low activity during the night. In contrast, PCA discovers trends that are not as clear, which suffer from the 'PCA confusion' phenomenon.



(a) Weekly pattern

(b) Daily pattern

(c) Weekend pattern

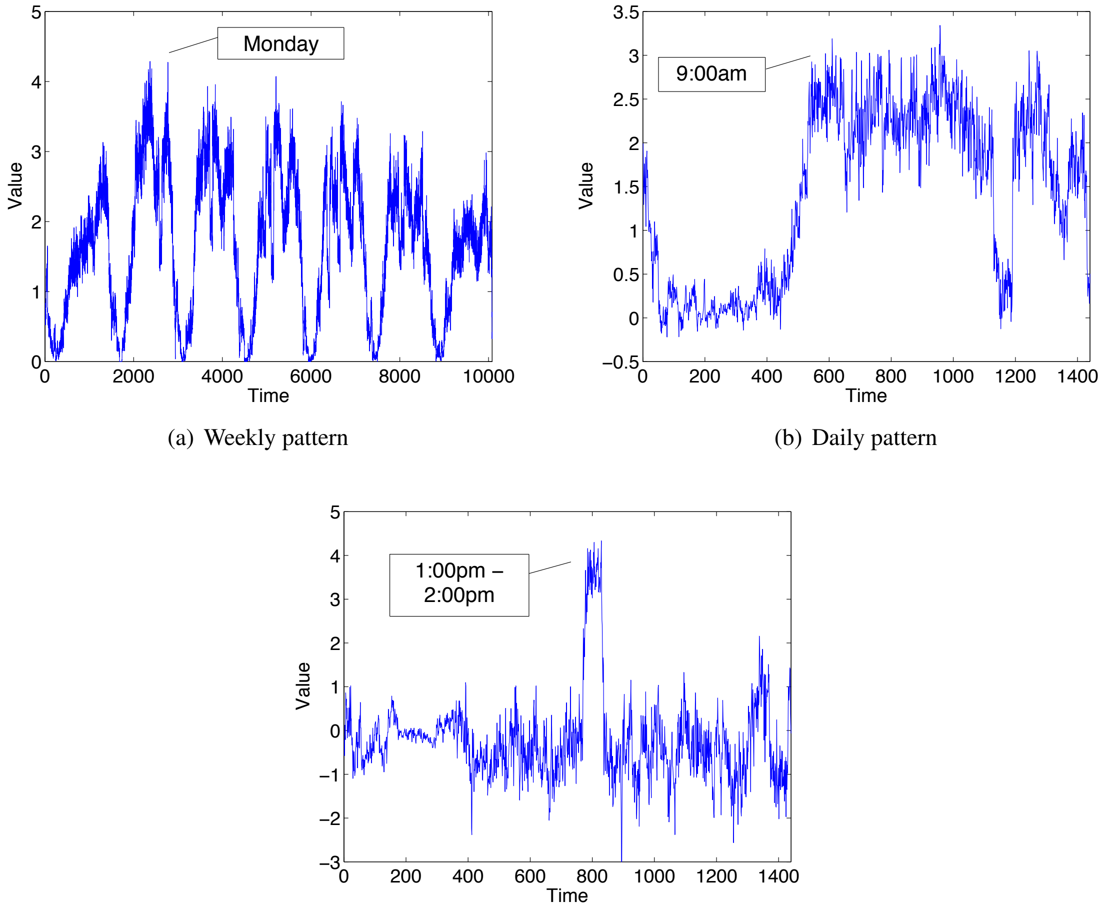
Figure 8.7: Frequently used components for the Q & A site of `WebClick`. (a) Major weekly trend/component, showing similar activity during all 7 days of the week. (b) Major daily trend - note the low activity during sleeping time, as well as the dip at dinner time. (c) Major weekday pattern - note the spike during lunch time.

8.4 Evaluation

To evaluate the effectiveness of WindMine, we carried out experiments on real datasets. We conducted our experiments on an Intel Core 2 Duo 1.86GHz with 4GB of memory, and running Linux. Note that all components/patterns presented in this section are generated by the scalable version, WindMine-part, while both versions provide useful results for the applications.

The experiments were designed to answer the following questions:

1. How successful is WindMine in local component analysis?
2. Does WindMine correctly find the best window size for mining locally patterns?
3. How does WindMine scale with the number of subsequences n in terms of computational time?



(a) Weekly pattern

(b) Daily pattern

(c) Weekday additional pattern

Figure 8.8: Frequently used components for the job-seeking site of `WebClick`. (a) Major weekly trend, showing high activity on weekdays. (b) Major daily pattern. (c) Daily pattern, which is mainly applicable to weekdays.

8.4.1 Effectiveness in mining Web-click sequences

In this subsection we describe some of the applications for which WindMine proves useful. We present case studies of real web-click datasets to demonstrate the effectiveness of our approach in discovering the common trends for each sequence.

`Ondemand TV` This dataset is from the `Ondemand TV` service of 13,231 programs that users viewed in a 6-month period (from May 14th to November 15th, 2007). The data record the use of `Ondemand TV` by 109,474 anonymous users. It contains a list of attributes (e.g., content ID, the date the user watched the content, and the ID of the user who watched the content).

Figure 8.6 (a) shows the original sequence on the `Ondemand TV` dataset. It exhibits a cyclic daily pattern. There are anomaly spikes each day at about lunch time. Figure 8.6 (b)-(c) show that WindMine successfully captures the weekly and daily patterns from the dataset. It can be easily used to capture information for arbitrary time scales. For comparison, Figure 8.6 (d)-(e) show the best local patterns using the PCA technique. As shown in these figures, it is not robust against noise and anomaly spikes, and it cannot produce good results.

`WebClick` This dataset consists of the web-click records from *www.goo.ne.jp*, obtained over one month (from April 1st to 30th, 2007). It contains one billion records with 67 GB of storage. Each record has 3 attributes: user ID (2,582,252 anonymous users), URL group ID (1,797 groups), and the time stamp of the click. There are various types of URLs, such as “blog”, “news”, “health”, and “kids”.

Figures 8.7, 8.8 and 8.9 show the effectiveness of our method. Specifically, Figures 8.7 and 8.8 show the local components of the Q & A site and job-seeking site. The left, middle and right columns in Figure 8.7 show the weekly, daily, and weekend patterns, respectively. Our method identifies the daily period, which increases from morning to night and reaches a peak. This trend appears strongly, especially on weekends. In contrast, Figure 8.8 describes “business” trends. Starting from Monday, the daily access decreases as the weekend approaches. At 9:00 am, workers arrive at their office, and they look at the job-seeking website during a short break. Additionally, the right figure shows that there is a large spike during the lunch break.

Figure 8.9 shows the local patterns of other websites. We can observe interesting daily trends according to various lifestyles.

(a) Dictionary: Figure 8.9 (a) shows the daily trend of the dictionary site. The access count increases from 8:00 am and decreases from 11:00 pm. We consider this site to be used for business purposes since this trend is strong on weekdays.

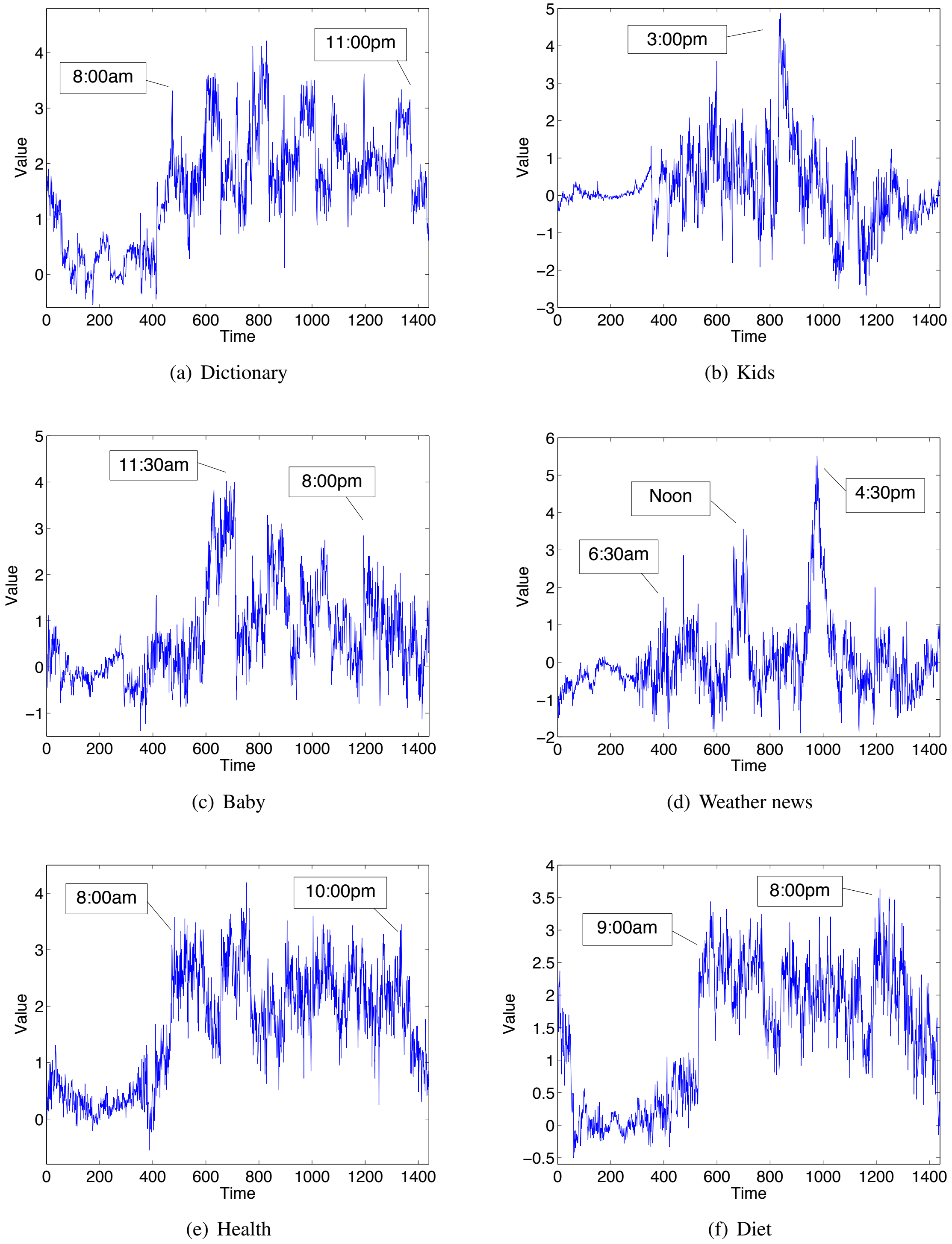
(b) Kids: Our method discovered a clear trend from an educational site for children. From this figure, we can recognize that they visit this site after school at 3:00 pm.

(c) Baby: This figure shows the daily pattern of the website as regards pregnancy and baby nursery resources. The access pattern shows the presence of several peaks until late evening, which is very different from the kids site. This is probably because the kids site is visited by elementary school children whereas the main users of the baby site will be their parents, rather than babies!

(d) Weather news: This website provides official weather observations, weather forecasts and climate information. We observed that the users typically check this site three times a day. We can recognize a pattern of behavior. They visit this site in the early morning and at noon before going outside. In the early evening, they check their local weather for the following day.

(e) Health: This is the main result of the healthcare site. The result shows that the users rarely visit website late in the evening, which is indeed good for their health.

(f) Diet: This is the main daily pattern of an on-line magazine site that provides information about diet, nutrition and fitness. The access count increases rapidly after meal times. We also observed that the count is still high in the middle of the night. We think that perhaps a healthy diet should include an earlier bed time.



(a) Dictionary

(b) Kids

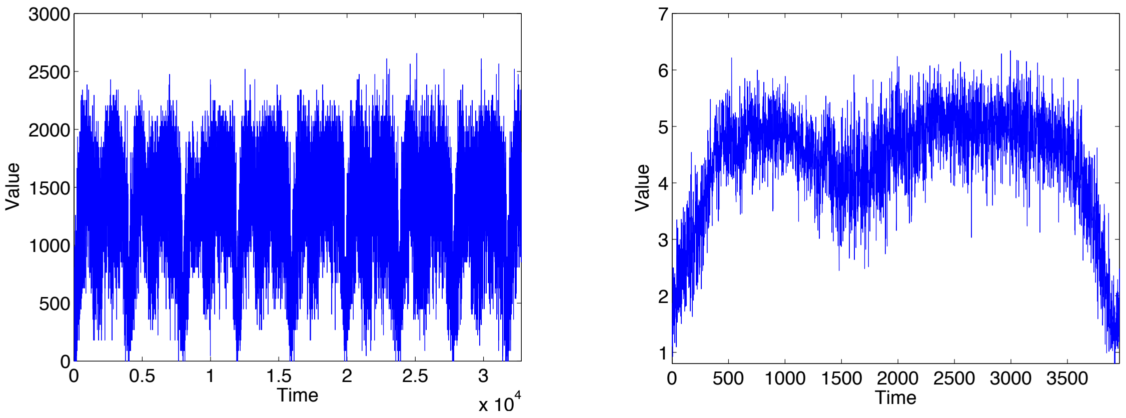
(c) Baby

(d) Weather news

(e) Health

(f) Diet

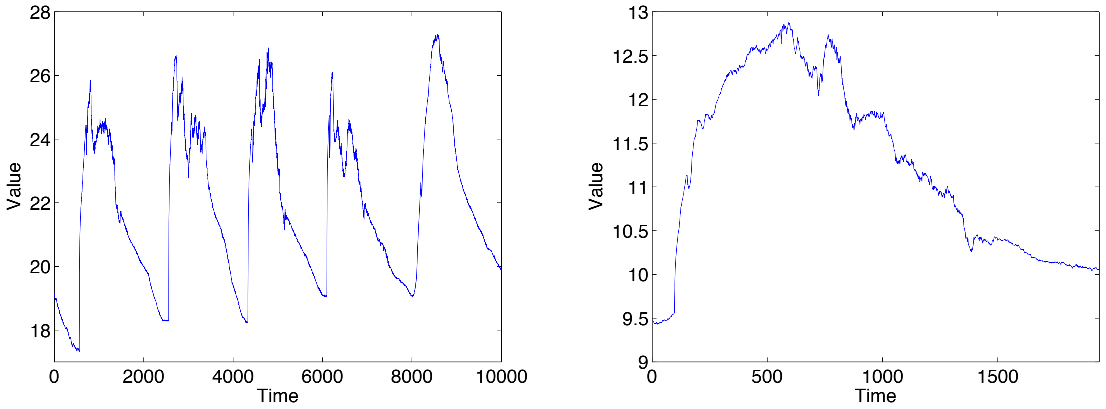
Figure 8.9: Daily components for the dictionary, kids, baby, weather news, health and diet sites of *WebClick*. WindMine discovers daily trends according to various lifestyles.



(a) Automobile

(b) Automobile pattern

Figure 8.10: Data sequence and the pattern by found WindMine for `Automobile`.



(a) Automobile

(b) Automobile pattern

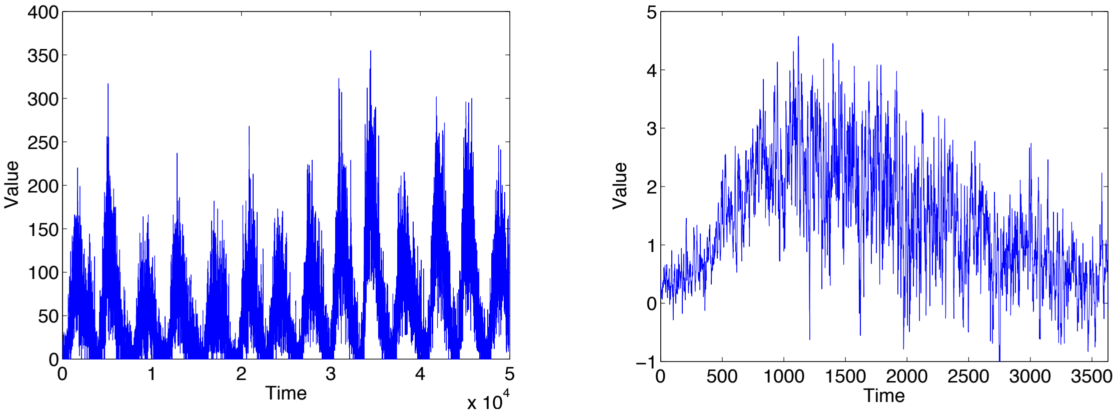
Figure 8.11: Data sequence and the pattern by found WindMine for `Temperature`.

8.4.2 Generalization: WindMine for other time series

We demonstrate the effectiveness of our approach in discovering the trends for other types of sequences.

`Automobile` This dataset consists of automobile traffic count for a large, west coast interstate. The left plot of Figure 8.10 (a) exhibits a clear daily periodicity. The main trend repeats at a window of approximately 4000 timestamps. Also, during each day there is another distinct pattern of morning and afternoon rush hours. However, these peaks have distinctly different shapes: the evening peak is more spread out, the morning peak is more concentrated and slightly sharper.

The right side of Figure 8.10 (a) shows the output of WindMine for the `Automobile` dataset. The common trend seen in the figure successfully captures the two peaks and also their approximate shape.



(a) Automobile

(b) Automobile pattern

Figure 8.12: Data sequence and the pattern by found WindMine for `Sunspot`.

`Temperature` We used temperature measurements (degrees Celsius) in the Critter data set, which comes from small sensors within several buildings. In this dataset there are some missing values, and it exhibits a cyclic pattern, with cycles lasting less than 2000 time ticks. This is the same dataset that was used in our previous study [Sakurai et al., 2005a].

Our method correctly captures the right window for the main trend and also an accurate picture of the typical daily pattern. As shown in Figure 8.11 (b), there are similar patterns that fluctuate significantly with the weather conditions (which range from 17 to 27 degrees). Actually, WindMine finds the daily trend when the temperature fluctuates between cool and hot.

Sunspots

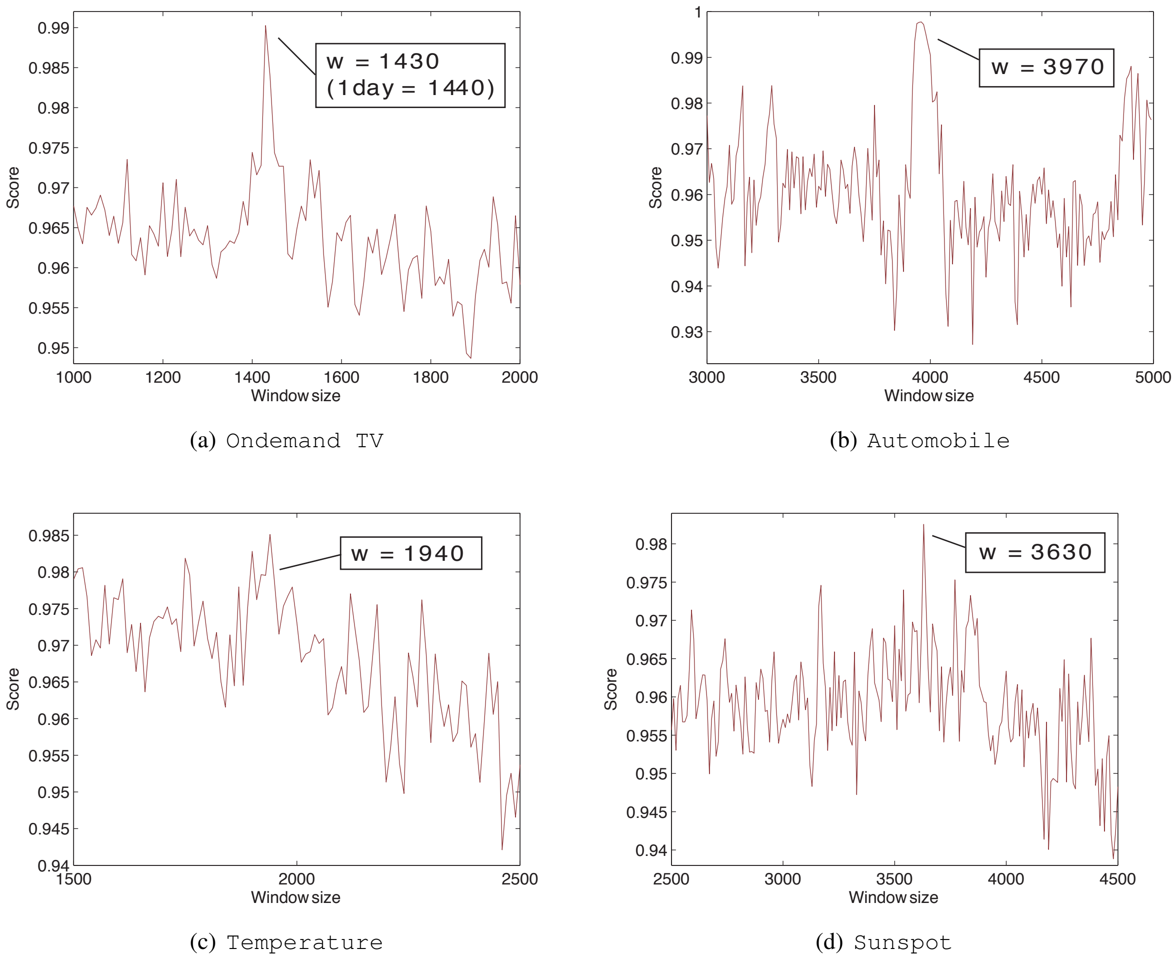
We know that sunspots appear in cycles. The left column of Figure 8.12 (c) indicates the number of sunspots per day. For example, during one 30-year period within the so-called "Maunder Minimum", only about 50 sunspots were observed, as opposed to the normal 40,000-50,000 spots. The average number of visible sunspots varies over time, increasing and decreasing in a regular cycle of between 9.5 and 11 years, averaging about 10.8 years [2].

WindMine can capture bursty sunspot periods and identify the common trends in the `Sunspot` dataset. The right column in Figure 8.12 (c) shows that our method provides an accurate picture of what typically happens within a cycle.

8.4.3 Choice of best window size

We evaluate the accuracy of the CEM criterion for window size selection. Figure 8.13 (a) presents the CEM score for *Ondemand TV*, for various window sizes. This figure shows that WindMine can determine the best window size by using the CEM criterion. As expected, our method indeed suggests that the

[2] http://csep10.phys.utk.edu/astr162/lect/sun/sscycle.html



(a) `Ondemand TV`

(b) `Automobile`

(c) `Temperature`

(d) `Sunspot`

Figure 8.13: CEM scores all data sets.

best window is daily periodicity. It identifies $w = 1430$ as the best window size, which is close to the one-day duration ($w = 1440$). Due to the window size estimation, we can discover the daily pattern for `Ondemand TV` (see Figure 8.6 (c)).

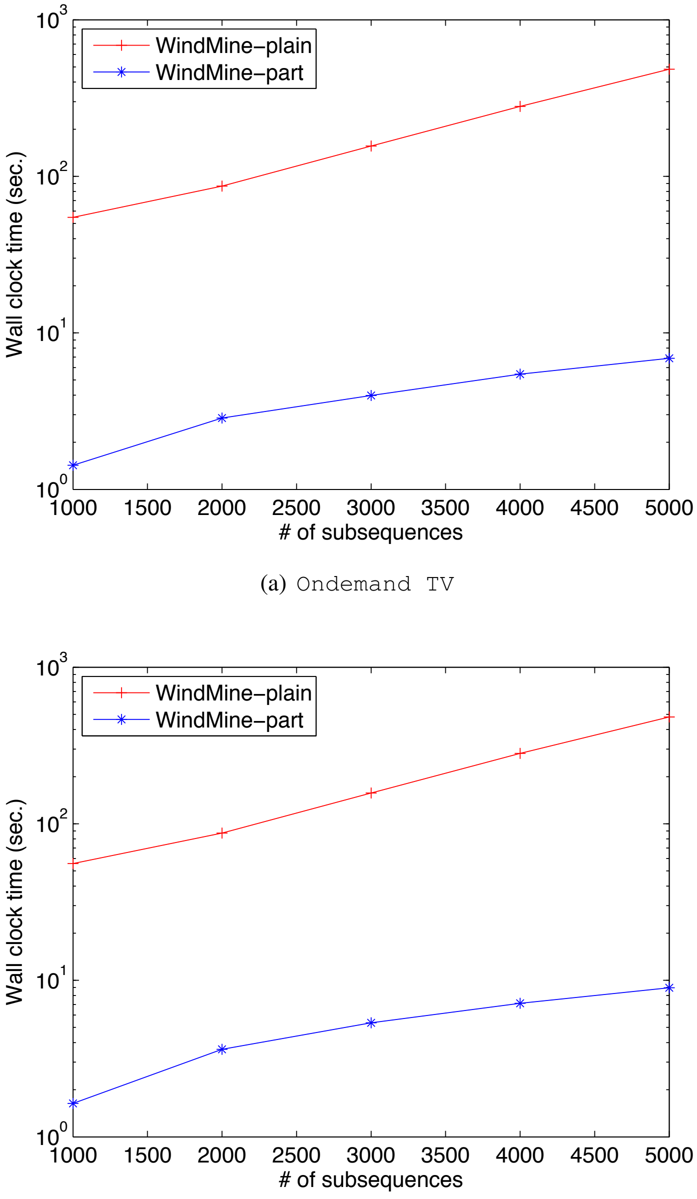
Figure 8.13 (b)-(d) show the CEM scores per window element for `Automobile`, `Temperature` and `Sunspot`, respectively. Note that the `Temperature` dataset includes missing values and the `Sunspot` dataset has time-varying periodicity. As shown in these figures, WindMine successfully detects the best window size for each dataset, which corresponds to the duration of the main trend (see the figures of the right in Figure 8.12).

8.4.4 Performance

We conducted experiments to evaluate the efficiency of our method. Figure 8.14 compares WindMine-plain and the scalable version, WindMine-part, in terms of computation time for different numbers of subsequences. The wall clock time is the processing time needed to capture the trends of subsequences.

Note that the vertical axis is logarithmic. We observed that WindMine-part achieves a dramatic reduction in computation time that can be up to 70 times faster than the plain method.

Figure 8.15 shows the wall clock time as a function of duration T. The plots were generated using `WebClick`. Although the run-time curves are not so smooth due to the convergence of ICA, they reveal the almost linear dependence of the duration T. As we expected, WindMine-part identifies the trends of subsequences much faster than WindMine-plain.



(a) `Ondemand TV`

(b) `WebClick`

Figure 8.14: Scalability: wall clock time vs. # of subsequences.

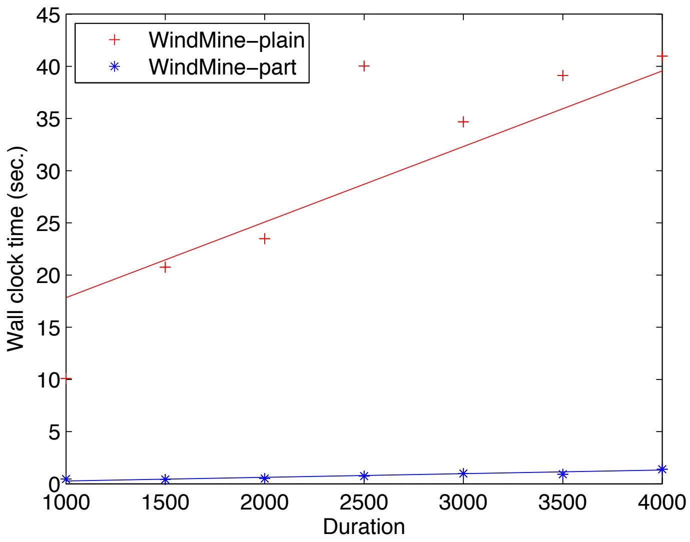


Figure 8.15: Scalability (`Ondemand TV`): wall clock time vs. duration.

8.5 Summary

In this chapter, we focused on the problem of fast, scalable pattern extraction and anomaly detection in large web-click sequences. The main contributions of this work are as follows:

1. We proposed WindMine, a scalable, parallelizable method for breaking sequences into a few, fundamental ingredients (e.g., spikes, cycles).
2. We described a partitioning version, which has the same accuracy, but scales *linearly* over the sequence duration, and near-linearly on the number of sequences.
3. We proposed a criterion that allows us to choose the best window sizes from the data sequences.
4. We applied WindMine to several real sets of sequences (web-click data, sensor measurements) and showed how to derive useful information (spikes, differentiation of weekdays from weekends). WindMine is fast and practical, and requires only a few minutes to process 67 GB of data on commodity hardware.

Part III

Domain Specific Algorithms and Case Studies

Chapter 9

Natural Human Motion Stitching

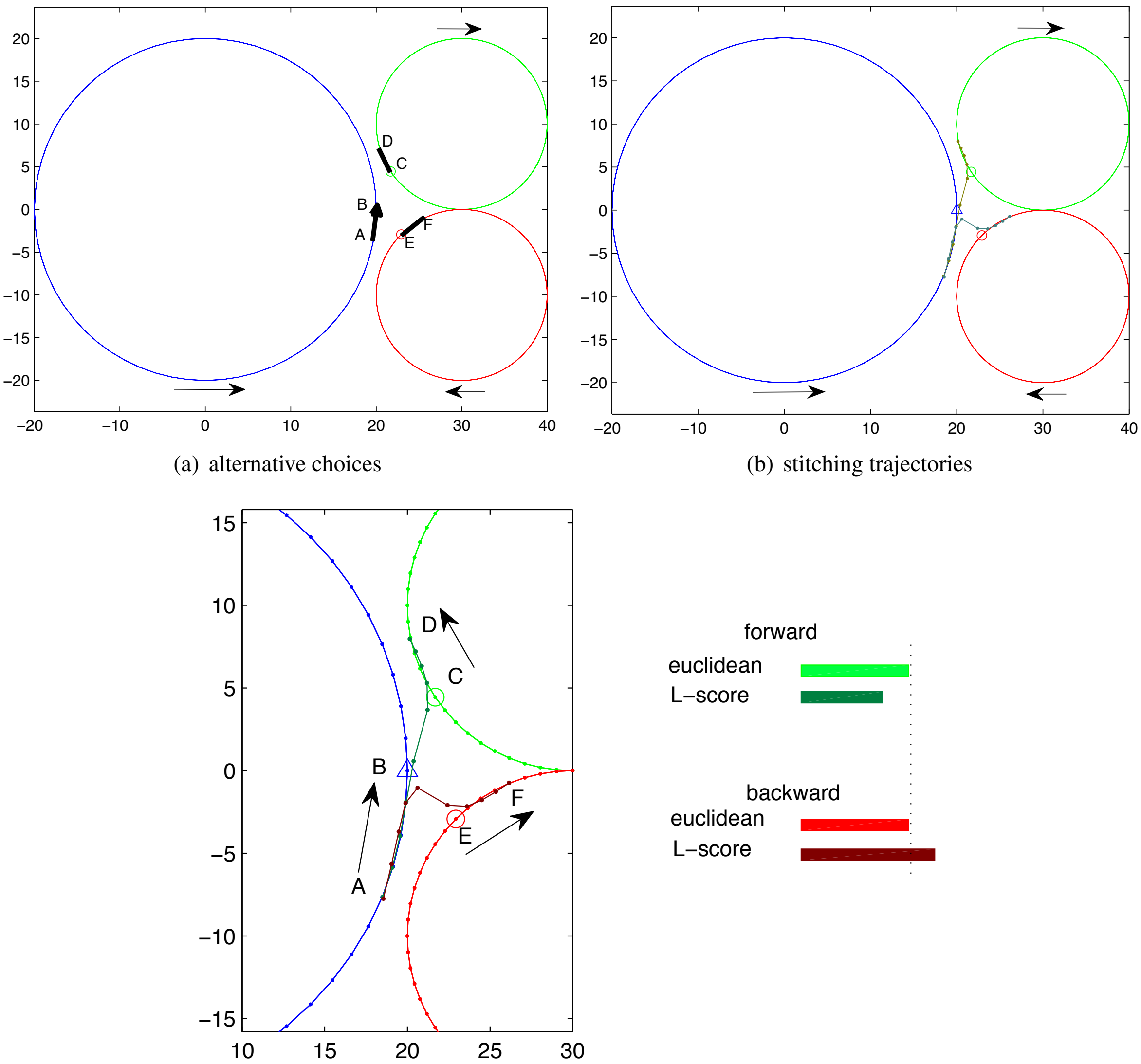
Given two motion-capture sequences that are to be stitched together, how can we assess the goodness of the stitching? The straightforward solution, Euclidean distance, permits counter-intuitive results because it ignores the effort required to actually make the stitch. In this chapter, we present an intuitive, first-principles approach, by computing the effort that is needed to do the transition (laziness-effort, or 'L-Score'). Our conjecture is that, the smaller the effort, the more natural the transition will seem to humans. Moreover, we propose the elastic L-Score which allows for elongated stitching, to make a transition as natural as possible. We present preliminary experiments on both artificial and real motions which show that our L-Score approach indeed agrees with human intuition, it chooses good stitching points, and generates natural transition paths.

9.1 Introduction

Human motion control and generation is an important research area and has many applications in the game and movie industries. The most important issue in this area is the generation of realistic character motion. One approach to this problem is to make use of motion capture data. With various sensing methods, one could capture the movement and posture of the human body, and build a large database of basic human movement, e.g. walking, running and jumping. Techniques exist which generate new motion from such a database by stitching together old motions. The success of these techniques depends, to a large extent, on a good choice of stitching distance function.

A good distance function is important for the generation of realistic character motion from motion capture databases. Given two captured human motion sequences that are to be stitched together, how can we assess the goodness of the stitching? In this paper, we propose a novel distance function to pick natural stitching points between human motions. To motivate our work, we demonstrate that a a straightforward, ad-hoc approach may lead to poor stitchings. For example, Figure 9.1 shows a problem case for the often-used *windowed Euclidean distance*[Wang and Bodenheimer, 2003]. Other ad-hoc metrics like time-warping and geodesic joint-angle distance [Wang and Bodenheimer, 2004] may suffer from similar issues, because none of them tries to capture the dynamics of the stitching as explicitly as our upcoming proposal does.

How do we capture the "naturalness" of a stitching? Our approach is to go to first principles, informally



(a) alternative choices

(b) stitching trajectories

(c) Magnified stitching

Figure 9.1: Motivating example: The stitching from (AB)-to-(CD) ("forward") seems more natural than the stitching (AB)-to-(EF) ("backward"). The right part shows the corresponding "stitchability" scores. However, the Euclidean distance does not capture the awkwardness of the actual stitching and assigns the same cost (about 47) to both. The "forward" stitching (AB)-to-(CD) has a smoother, more natural-looking trajectory(darker lines).

expressed in our following conjecture:

Conjecture 9.1 (Laziness)**.** *Between two similar trajectories, the one that looks more "natural" is the one that implies less effort/work.*

The rationale behind our conjecture is that humans and animals tend to minimize the work they spend during their motions, as captured in the "minimum jerk" [Flash and Hogan, 1985] and "minimum torque change" [Uno et al., 1989] hypotheses of motion, for example. Formally, we are focusing on the following problem (Figure 9.5):

Problem 9.1 (Stitching Naturalness)**. Given** *a query sequence* $\mathcal{Q}$ *of* $\mathrm{T_Q}$ *points in* m*-dimensional space with take-off point* $\vec{q}_a$*, and a data sequence* $\mathcal{X}$ *of* T *points of the same dimensionality with landing point* $\vec{x}_b$*,* **find** *a function to assess the goodness of the resulting stitched sequence, i.e.* $\vec{q}_1, \ldots, \vec{q}_a, \vec{x}_b, \ldots, \vec{x}_\mathrm{T}$.

The goal is that the "goodness" metric should be low if humans consider the stitching to be natural. Once we obtain a qualified distance function, we can either do a sequential scan or use database indexing techniques to perform a fast search over the whole motion capture database to find the best stitching motions [Lee et al., 2002].

The main contribution of this work is: a good stitching should require less work than a bad stitching. Additional contributions include the details of our approach, and specifically the following:

- We show how to estimate the hidden variables (velocities, accelerations), even in the presence of noise
- Our technique supports fast *elastic* version of stitching, which automatically computes the optimal number of frames to interject, so that the stitching looks as natural as possible.

The rest of this chapter is organized as follows. Section 9.2 will describe the motion capture system, body skeleton and generated data formats. Section 9.3 gives a survey of related work in this area. Section 9.4 defines the problem and our proposed L-Score and our proposed extension, the *elastic* stitching, where we automatically estimate the optimal number of frames to inject, to make a stitching even more natural-looking. Section 9.5 presents the results obtained on both artificial data and real motion capture data.

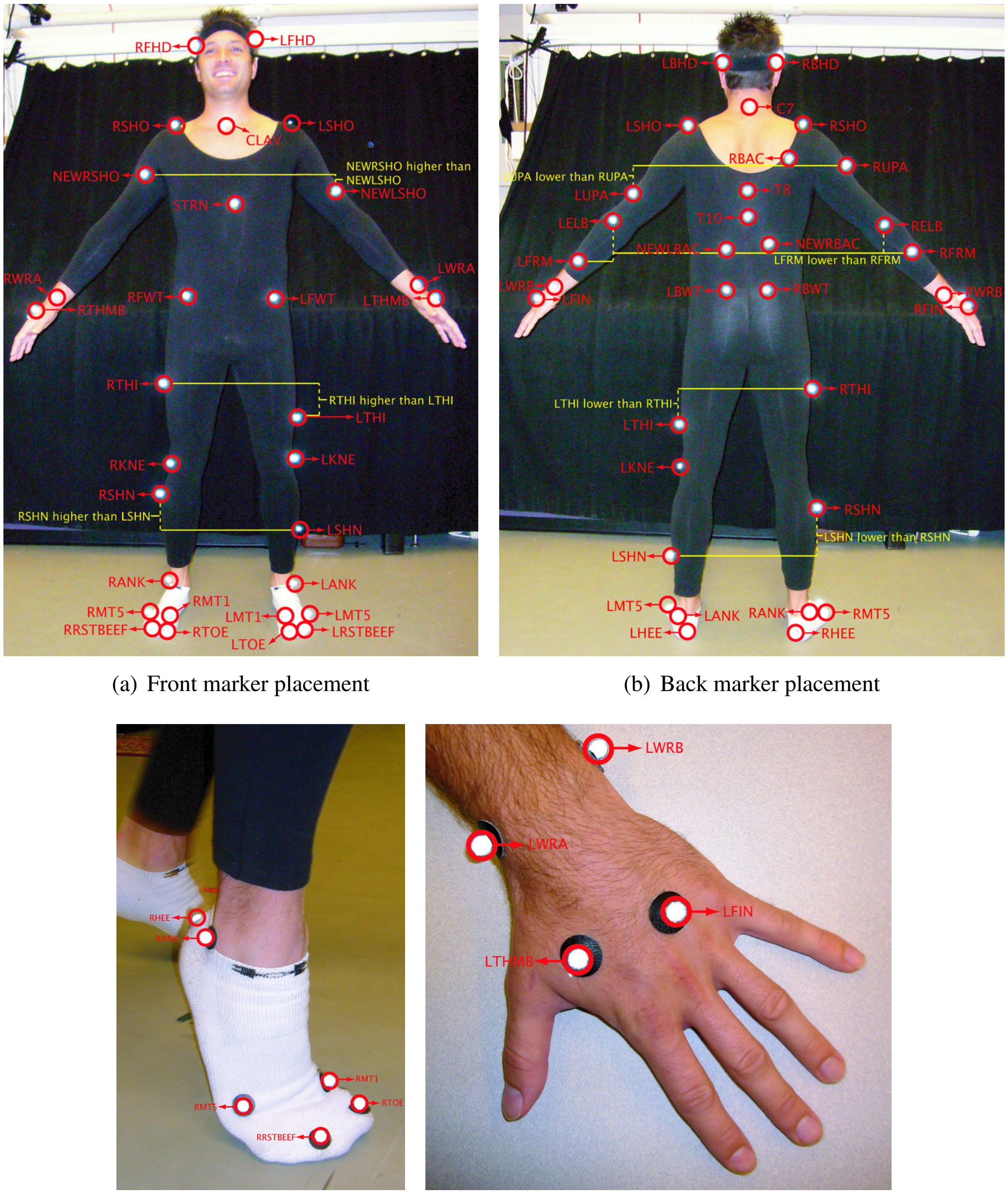
9.2 Motion Capture and Human Motion

There are several types of motion capture systems, following into two major categories: (a) optical systems and (b) non-optical system. In optical systems, markers are often attached to human body, face and fingers while multiple cameras are used to track the position of makers and human body. Markers can be made of many technology: such as passive markers, active markers, semi-passive markers, and even markerless version. For example, CMU mocap lab [1] uses passive optical systems with markers made from reflective materials. The recent entertainment system, Kinect, requires no markers and uses a set of normal cameras and a infra-red camera to capture the movement of a player. On the other hand, non-optical systems often directly measure the dynamics of body parts using inertial sensors or body joint angles using skeleton systems. Throughout this thesis, we will target optical motion capture systems with passive markers. We will solve a series of problems arising from real applications of motion capture. We have already seen missing value imputation problem in Chapter 3. In this chapter and next chapter, we will propose techniques for two more specific problems in modeling motion capture data.

Figure 9.2 exhibits a marker system attached on human actors. In our experiments, we use 41 passive markers. Each marker will be translated into xyz-coordinates, thus in total yielding 123 sequences of marker positions. The 123-dimensional data will be used in Chapter 10 when we utilize the body skeleton information. In addition, there are two other formats for motion capture data: one describes the animated skeleton in terms of bones and joints angles between them (AMC format); the other further describes the positions of ends of bones in terms of body local coordinates. In body local coordinates, all markers are positioned in reference to the center of body mass but projected on the ground. For example, the hip will

[1] http://mocap.cs.cmu.edu

be at (0,0,1) for a standing still actor. The AMC format has 62 dimensions, while the other version has 96 dimensions. Unless otherwise noted, we will mainly use the last version of the motion capture sequences. However, our techniques can be either directly or easily extended to other formats. Figure 9.3 shows sequences of joint angles for a walking motion. Figure 9.4 shows sequences of translated bone positions in body local coordinates for a Taichi performance. Note the correlation among the multiple sequences in both formats.



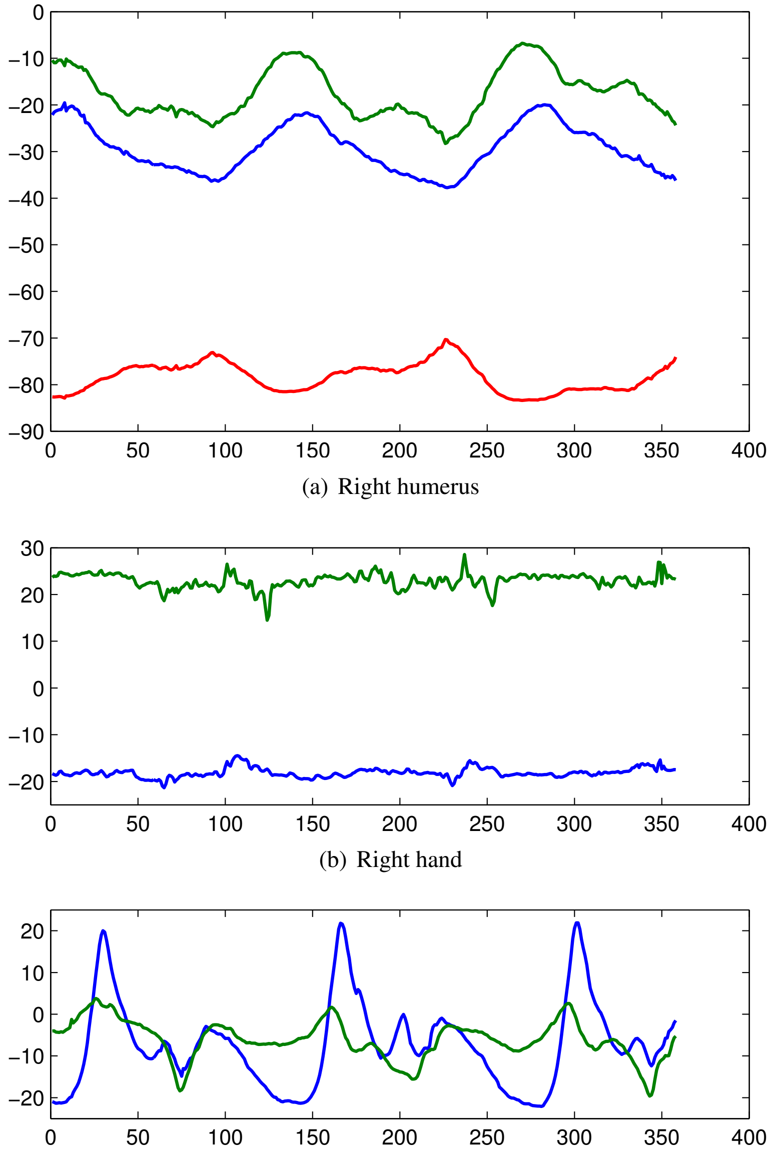
(a) Front marker placement

(b) Back marker placement

(c) Foot marker placement

(d) Hand marker placement

Figure 9.2: Placement of optical markers for motion capture systems. The figures are created at CMU mocap lab[2].



(a) Right humerus

(b) Right hand

(c) Right foot

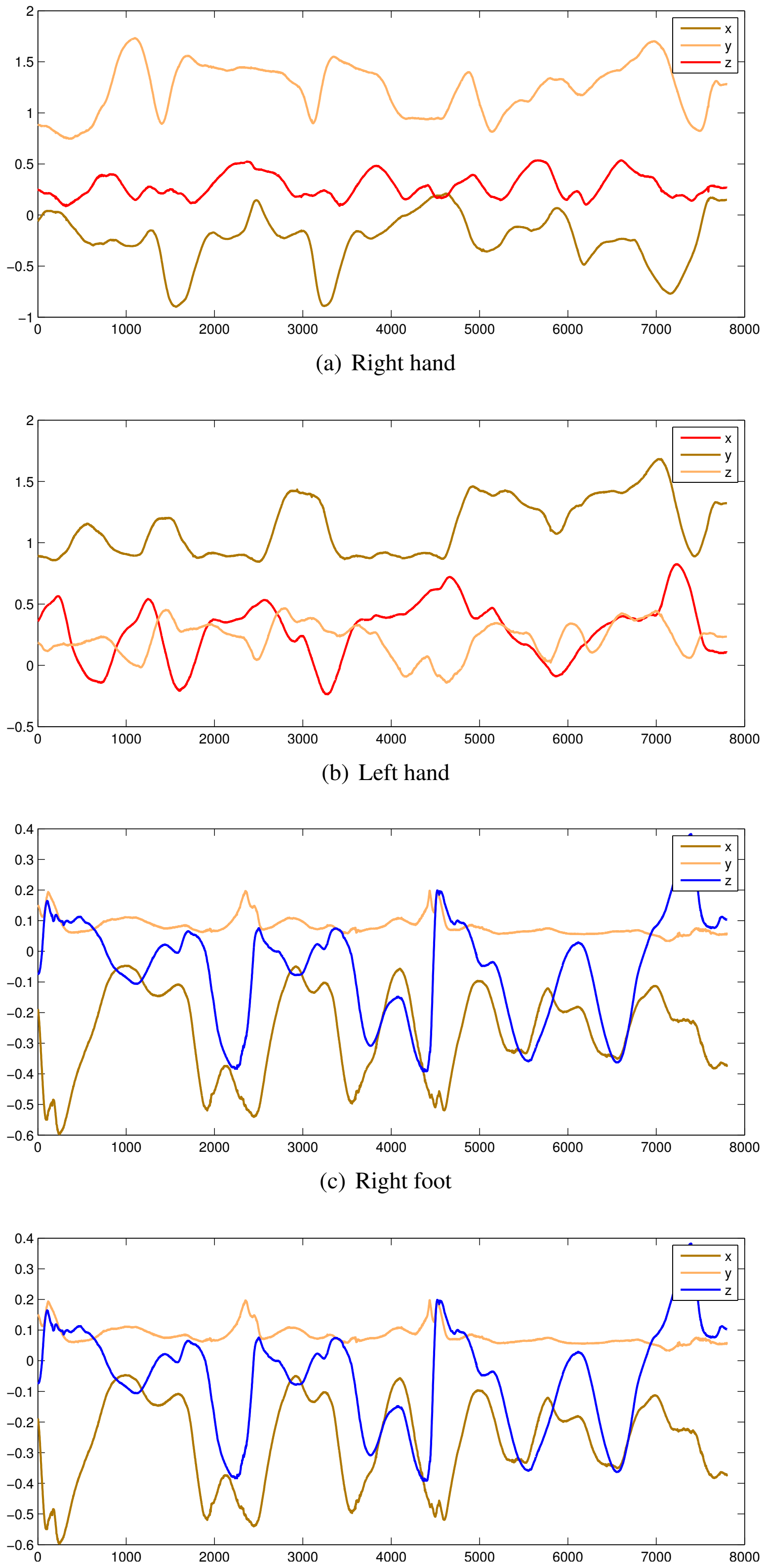
Figure 9.3: Joint angle sequences for a walking motion (subject#35.01).

9.3 Related work

There has been a great deal of research related to generation of motion transitions (e.g. creation of motion graphs) [Kovar et al., 2002, Lee et al., 2002, Arikan and Forsyth, 2002, Li et al., 2002]. However, the focus of these works is not on the distance metrics themselves.

Li et al [Li et al., 2002] use a statistical two level Markov approach to learn basic motion textons, and

[2] http://mocap.cs.cmu.edu



(a) Right hand

(b) Left hand

(c) Right foot

(d) Left foot

Figure 9.4: Sequences of bone positions for a Taichi motion (subject#10.04). Note the correlation between left foot and right foot, left hand and right hand, which will help fill in occlusions.

synthesize complex motions from them accordingly. In essence, they use the transition likelihood between the textons as the distance, and then generate new motions with respect to the likelihood.

Lee et al [Lee et al., 2002] describe methods to identify and efficiently search plausible transitions of motion segments from a motion capture database; additionally, they provide interactive interfaces to control avatar motion. They use a weighted euclidean distance on both position and joint angles to asses the motion transition quality. Arikan and Forsyth [Arikan and Forsyth, 2002] propose a randomized search approach to synthesize human motions under a given set of constraints. Their distance function is a normalized weighted euclidean distance on both position and velocity. Kovar et al [Kovar et al., 2002] use the motion graph to encode natural transitions between behaviors and can generate motion with good local quality by constraining the motion to be very close to captured examples. They also use a weighted euclidean distance on positions, though they pick the minimum over all possible linear transformations.

The euclidean distance was revisited by Wang and Bodenheimer [Wang and Bodenheimer, 2003], who calculated an optimal weighting for the metric used in [Lee et al., 2002] based on human-subjects experiments. In addition, Wang and Bodenheimer [Wang and Bodenheimer, 2004] proposed two heuristics for computing an optimal duration for a linear blend from a start motion to an end motion. They assume a start and end frame are given, and propose the following two metrics: (1) identify a blend duration to minimize average geodesic difference between poses during the blend, and (2) directly compute a blend length based on the velocity of the joint having the largest joint angle difference between the start and end point. In contrast, our elastic L-Score explicitly models the dynamics of the motion, and estimates a path based on these dynamics.

Rose et al. [Rose et al., 1996] and Liu and Cohen [Liu and Cohen, 1995] demonstrate techniques for computing minimal energy transitions between start and end frames, using a numerical optimization formulation that considers full body dynamics. The approach of Liu and Cohen is also ”elastic”, in the sense that duration of the transition is a parameter that can be tuned to generate a transition of minimum energy. However, our method differs in two aspects: (a) we focus on designing a distance function and (b) our method is much faster, requiring computation time that is linear on the number of frames involved. Speed is important for our target problem (see Problem 0, later), namely, the rapid identification of good transitions in a database of several motion-capture sequences.

However, none of the above distance functions takes the dynamics into account, which is the main contribution of this work.

Our work is also related to dynamics-based filters for nonlinear systems, including extended Kalman, unscented Kalman [Julier and Uhlmann, 1997], and particle filters. These filters try to track the whole body as one unified but highly nonlinear system. Such approaches have been applied with some success to human motion data (e.g. [Tak and Ko, 2005]) and human tracking problems (e.g. [Rosales and Sclaroff, 1998]). Unlike these approaches, we use a conceptual simpler Kalman filter to estimate the dynamics independently for each body bone. Methods such as in [Tak and Ko, 2005] could be put in our setting, with the expected cost in computation time.

9.4 Motion stitch and laziness score

We will first state a formal definition of our straw-man, the *Euclidean distance*(its weighted version) [Lee et al., 2002].

Table 9.1: Symbols and Definitions

Symbol	Definition
$\mathcal{X}$	a data sequence $(\vec{x}_1, ...\vec{x}_{\mathrm{T}})$ of T time-ticks in m dimensions
$\mathcal{Q}$	a query sequence, $(\vec{q}_1, ...\vec{q}_{\mathrm{T_Q}})$ of $\mathrm{T_Q}$ time-ticks and the same dimensionality
m	number of dimensions (e.g., angles, positions) - m=93 here
a, b	frame-number for the take-off point and the landing point
$L()$,$L_k()$	$L-Score$ and its extension for k injected frames
$L_*()$	our proposed *elastic* $L-Score$
$\mathrm{T_Q}$,T	duration of each sequence (about 300-1,000 frames)
Δt	time gap between consecutive data points - Δt = 1 frame, here

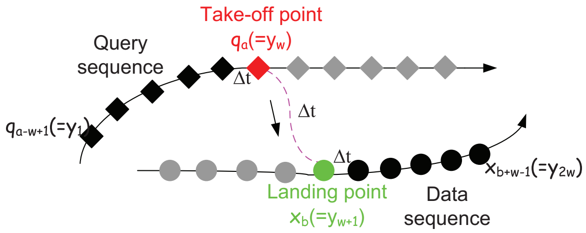


Figure 9.5: Illustration of Problem 9.1. The query trajectory (with diamonds) is to be stitched with the data trajectory (with circles) at the two indicated points: the red diamond indicates the take-off point q_a, and the green circle marks the landing point x_b. Grayed out points indicate points that we ignore in our stitching.

Definition 9.1. *Given two data sequences $\mathcal{Q}$ and $\mathcal{X}$ with the take-off point a and landing point b, the* windowed Euclidean distance *for motion stitching is defined as*

$$\mathrm{D_{eu}}(\mathcal{Q}, \mathcal{X}) = \sum_{j=-w}^{w} \|\vec{q}_{a+j} - \vec{x}_{b+j-1}\|^2 \tag{9.1}$$

which means that we compare the w frames before, and the w frames after the transition.

Here, we describe our L-Score for motion stitching. The idea is to exploit Conjecture 9.1, that humans tend to use as little work as possible and thus natural human motion transitions should be work-efficient. Our L-Score relies on a fast, easy to compute estimate of the effort required to make a stitch. To create the stitch, any on-the-shelf regression/fitting method (e.g. linear interpolation or spline) could be plugged into our L-Score method in principle. However, these methods need manual tuning(e.g. order of spline). We recommend the Kalman filter to estimate the motion dynamics for the following reasons (a) it has explicit (Newtonian) dynamic equations consistent with first principles (b) it could reduce noise as well. Kalman filters have already been applied to human motion data for retargeting [Tak and Ko, 2005] and computer puppetry [Shin et al., 2001].

9.4.1 Estimation of Dynamics

Given the query sequence $\mathcal{Q}$ and the data sequence $\mathcal{X}$ in m dimensions, we create a new stitching sequence (within a certain window size w) $\mathcal{Y} = \vec{y}_1, \ldots, \vec{y}_{2w} = \vec{q}_{a-w+1}, \ldots, \vec{q}_a, \vec{x}_b, \ldots, \vec{x}_{b+w-1}$ (Figure 9.5). To estimate the trajectory in the stitching process, we try to find the hidden dynamics (the true position, velocity, and acceleration) at each time tick, while eliminating the observation noise. Given the observed position at every time tick, we build the following Kalman filter(Eq. 9.2) for each dimension of the stitching sequence. For the following, we assume the data sequence is one dimensional.

$$\begin{aligned} \vec{z}_1 &= \mu_0 + \omega_0 \\ \vec{z}_{n+1} &= \mathbf{A}\vec{z}_n + \omega_n, \qquad \mathbf{A} = \begin{pmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \\ y_n &= \mathbf{C}\vec{z}_n + \epsilon_n \end{aligned} \tag{9.2}$$

where the hidden states consist of true position p_n, velocity v_n, acceleration a_n: $\vec{z}_n = (p_n, v_n, a_n)^T$. The transition matrix $\mathbf{A}$ is determined from Newtonian mechanics of a point mass, and the transmission matrix $\mathbf{C} = (1 \quad 0 \quad 0)$, with Gaussian noise terms $\omega_t \sim \mathcal{N}(0, \text{diag}(\gamma_1, \gamma_2, \gamma_3))$ and $\epsilon_t \sim \mathcal{N}(0, \sigma)$. We set the prior parameter $\mu_0 = (p_0, v_0, a_0)^T = (y_1, (y_2 - y_1)/\Delta t, (y_3 + y_1 - 2y_2)/\Delta t^2)^T$.

We use the forward-backward algorithm [Welch and Bishop, 2001] to achieve our goal: to estimate the expected value of the hidden states $\hat{\vec{z}}_n = \mathbb{E}[\vec{z}_n \mid \mathcal{Y}](n = 1, \ldots, 2w)$. For motion stitching data in m dimensional space, we build the Kalman estimation for each dimension and estimate position, velocity and acceleration separately.

9.4.2 L-Score

Now that the velocities and accelerations have been calculated, the next step is to calculate the effort during the stitching.

Given the estimated hidden states $\hat{\vec{z}}_n = (\hat{p}_n, \hat{v}_n, \hat{a}_n)^T (n = 1, \ldots, 2w)$, we can compute the energy spent as the product of force and displacement. Thus, we define the following L-Score = $L(\mathcal{Q}, a, \mathcal{X}, b, w)$ for motion stitching:

$$L(\mathcal{Q}, a, \mathcal{X}, b, w) = \sum_{n=1}^{2w-1} |(\hat{p}_{n+1} - \hat{p}_n) \cdot \hat{a}_n| \tag{9.3}$$

9.4.3 Generalization: Elastic L-Score

The L-Score we just presented approximates the effort of transition during motion stitching. As we mentioned, it assumes flying from the take-off frame of query motion directly to the landing frame of data sequence, without any intermediate “stops”. Thus, it may result in abrupt transitions. We propose to remedy it, by allowing the injection of some intermediate frames between take-off and landing. The question is, how should we choose a good number of such frames? It turns out that our earlier framework can be easily extended, and it can estimate not only the best k injected frames (when k is given), but it can also estimate the best value for k itself! To generalize even more, we make the number of *injected frames* adaptive to the motions to be stitched. Figure 9.6 illustrates our idea and terminologies. To estimate the proper number k_{opt} of injected frames, we resort to our ’laziness’ conjecture (conjecture 9.1), and we select the transition with the lowest effort. This is exactly the intuition behind our proposed *elastic* L-Score. Formally, we address the following problem:

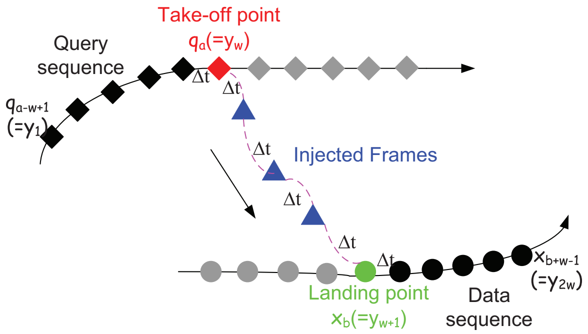


Figure 9.6: Illustration of “take-off”, “injected” and “landing” points. The trajectory of squares is to be stitched with the trajectory of circles at the two indicated points (red square for take-off, green circle for landing); the injected frames, shown as blue triangles, which are to be estimated. Grayed out points indicate points ignored in our stitching.

Problem 9.2 (Elastic Stitching)**. Given** *a query sequence $\mathcal{Q}$ of* T *points in m-dimensional space with the take-off point $\vec{q}_a$; a data sequence $\mathcal{X}$ of* T *points in the same dimensionality with landing point $\vec{x}_b$,* **find** *the suitable number k_{opt} as well as the best k_{opt} intermediate frames, such that the transition appears natural from the take-off point to the landing point through k_{opt} intermediate frames. Furthermore,* **find** *a function to assess the goodness of the resulting stitched sequence.*

We propose the *elastic*-L-Score method to solve this problem. The general procedure of the method works as follows. Given a query motion and a data sequence, generate a new sequence by connecting a few frames (w) from the query motion right before take-off (included), several artificial frames (k_{opt}, to be estimated), and a few frames (w) of target motion after landing (included). For the new sequence, we propose to use a variant of the Kalman filter, so that we can estimate the dynamics for each frame (actual, as well as injected), and then we compute the total L-Score as an approximation of the work. Once we get the L-Score, we could range over all possible numbers k of injected frames, and choose k_{opt} as the one with the minimum L-Score. The challenge here is to estimate the dynamics for injected frames; for the non-injected ones, we can use the Kalman equations of Section 2.2.5

Let’s start with the easier problem, where the number of injected frames k is given to us.

Given the query motion sequence $\mathcal{Q}$ with take-off frame at a, the data sequence $\mathcal{X}$ with landing frame at b, and the number of injected frames k, we create a new stitching sequence $\mathcal{Y}$: $\vec{y}_i = \vec{q}_{a-w+i}(i = 1, \ldots, w)$, and $\vec{y}_{w+i-1} = \vec{x}_{b+i-1}(i = 1, \ldots, w)$ (w is the window size). For the hidden states $\vec{z}$, we inject k hidden variables ($\vec{z'}_n = (p'_n, v'_n, a'_n)^T$, $n = 1 \ldots k$) in the middle to model the true position, velocity and acceleration for injected frames. Again, we use the Kalman filter but with missing observations, to model the dynamics of the system (a simplified version of DynaMMo, as shown in Figure 9.7). For the leading and ending w frames, the equations are the same as Eq. 9.2 except for z_{w+1}. In addition, we use the following equations.

$$\vec{z'}_1 = \mathbf{A}\vec{z}_w + \omega'_0 \tag{9.4}$$

$$\vec{z'}_{n+1} = \mathbf{A}\vec{z'}_n + \omega'_n \tag{9.5}$$

$$\vec{z}_{w+1} = \mathbf{A}\vec{z'}_k + \omega'_k \tag{9.6}$$

where ω'_n are Gaussian noises, with the variances:

$$\omega'_n \sim \mathcal{N}(0, \text{diag}(\gamma_1, \gamma_2, \gamma_3))$$

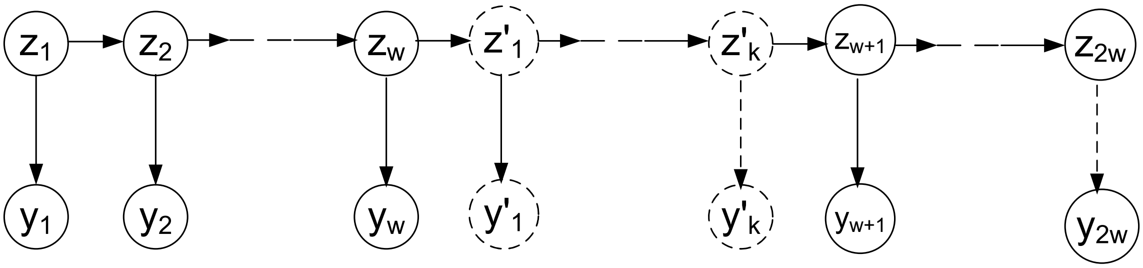


Figure 9.7: Graphical illustration of the Kalman filter with completely missing observations in the middle, $\vec{z}_1, \ldots, \vec{z}_{2w}$ indicate the hidden variables, $\vec{y}_1, \ldots, \vec{y}_w$ are the observations from query motion with take-off point at $\vec{y}_w$; variables $\vec{y}_{w+1}, \ldots, \vec{y}_{2w}$ stand for observations from data motion with landing point at $\vec{y}_{w+1}$. Variables $\vec{y'}_1, \ldots, \vec{y'}_k$ are the missing observations (injected frames), while $\vec{z'}_1, \ldots, \vec{z'}_k$ are the corresponding hidden variables. Arrows indicated Linear Gaussian probabilistic dependencies.

Again, the goal is to estimate the expected value of the hidden states, but now, with missing observations. That is, we want to estimate $\hat{\vec{z}}_n = \mathbb{E}[\vec{z}_n \mid \mathcal{Y}](n = 1, \ldots, 2w)$ and $\hat{\vec{z'}}_n = \mathbb{E}[\vec{z'}_n \mid \mathcal{Y}](n = 1, \ldots, k)$. Note that for $k = 0$, we have exactly Problem 9.1 of the previous section.

To solve the new set of equations we use a variant of the forward-backward algorithm to estimate the expectation of the posterior distribution $p(\vec{z}_n \mid \mathcal{Y})(n = 1, \ldots, 2w)$ and $p(\vec{z'}_n \mid \mathcal{Y})(n = 1, \ldots, k)$. The full details are omitted for brevity, see Section 2.2.5 for detail of the algorithm.

To assess the goodness of the motion stitching, we use the first principle: lower work leads to more natural motion transition. We could approximate the work used to make such transition from the estimated dynamics above. For the two motions to be stitched, we inject k frames in the middle and estimate the dynamics using the above method. To estimate the transition effort, we not only compute the work for real frames, but also compute the effort for injected frames. To justify this, we want to minimize the whole transition procedure, rather than minimize the effort for only one frame. We define the L-Score $L_k(\mathcal{Q}, a, \mathcal{X}, b, w)$ for fixed number of injections k, and the elastic L-Score $L_*()$ for the optimal number of injections k_{opt}.

$$\begin{aligned} L_k(\mathcal{Q}, a, \mathcal{X}, b, w) = & \sum_{n=1}^{w-1} |(\hat{p}_{n+1} - \hat{p}_n) \cdot \hat{a}_n| \\ & + |(\hat{p'}_1 - \hat{p}_w) \cdot \hat{a}_w| + \sum_{i=1}^{k-1} |(\hat{p'}_{i+1} - \hat{p'}_i) \cdot \hat{a'}_i| \\ & + |(\hat{p}_{w+1} - \hat{p'}_k) \cdot \hat{a'}_k| + \sum_{n=w+1}^{2w-1} |(\hat{p}_{n+1} - \hat{p}_n) \cdot \hat{a}_n| \end{aligned} \tag{9.7}$$

$$L_*(\mathcal{Q}, a, \mathcal{X}, b, w) = \min_{k \geq 0} \quad L_k(\mathcal{Q}, a, \mathcal{X}, b, w) \tag{9.8}$$

$$k_{opt} = \arg\min_{k \geq 0} \quad L_k(\mathcal{Q}, a, \mathcal{X}, b) \tag{9.9}$$

The elastic L-Score $L_*()$ not only gives an assessment of the stitching quality, but it also chooses the most suitable number k_{opt} of frames to inject - its goal is always to minimize the transition effort. Furthermore, once we decide the number k_{opt} of injected frames, we get a good transition trajectory for free: $\hat{p}_1, \ldots, \hat{p}_w, \hat{p'}_1, \ldots \hat{p'}_{k_{opt}}, \hat{p}_{w+1}, \ldots, \hat{p}_{2w}$.

9.5 Evaluation

We have already illustrated (Figure 9.1) that the Euclidean distance may lead to counter-intuitive results. Next we present experiments with the *elastic* $L-Score$ ($L_*()$) on (a) synthetic and (b) real motion capture data.

Synthetic Data We generated the `Three-Circles` dataset with a frame rate of $64/cycle$ in 2-dimensional space (m=2), as shown in Figure 9.1. The large, left circle has a radius of 20 units and is centered at $(0, 0)$; the right circles both have radius 10 units and are centered symmetrically at $(30, 10)$ and $(30, -10)$. We perform two experiments: the “forward” and the “backward” transition(Figure 9.1). In these experiments, we identify both the optimal landing point and the optimal number of injected frames. Figure 9.8 shows the results: the *elastic* L-Score favors the forward transition, which agrees with human intuition. It also chooses a larger number of frames and a later landing point to ameliorate the effects of the awkward backward transition.

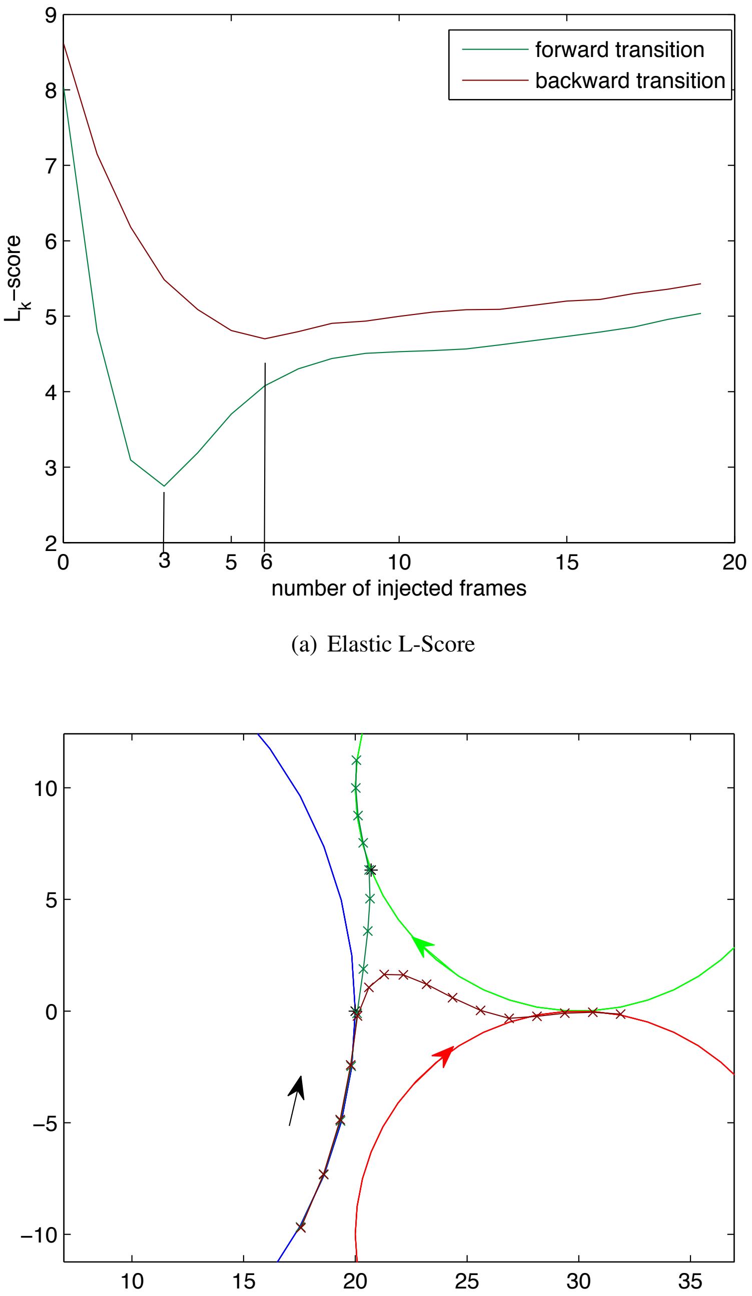
Real Human Motion: We capture a set of waving, walking, running and jumping motions at 30 frames per second. Motions are 300 to 2000 frames in length and have m=93 dimensional joint positions in body local coordinates. We use one Kalman filter for each of the m=93 features as described in Section 9.4, and set the parameters to be $\Delta t = 1$, $\gamma_1 = \gamma_2 = \gamma_3 = \sigma = 0.001$. We use the window of $2w = 10$. We have informally viewed a large variety of transitions within this database and find that our approach consistently performs as well or better than the Euclidean distance metric at generating pleasing transitions.

In order to assess the quality of the stitching found by our *elastic* L-Score, we blank out a short interval (2 frames) and a long interval (11 frames) from the transition made by the human actor during 2 waving circle motions, and we compare the actual trajectory against the transition trajectories estimated by the *elastic* L-Score. The processing time is around two and a half hours on a Pentium class machine. The observations (see Figure 9.9) are as follows:

- Our method computes the correct value of blanked-out frames, or gets very close to it.
- Our generated trajectories match very well the actual trajectories (please see the accompanying video).

9.6 Summary

In this chapter, we design of a new distance function for motion stitching, L-Score, based on first principles. Motivated by the weaknesses of Euclidean distance (Figure 9.1), we wished to more accurately capture the perceived “naturalness” of a trajectory. This led to our Conjecture 9.1, stating that the most natural-looking motion trajectory is the laziest-looking one, that is, the one that requires the least effort. The specific contributions of this chapter are the following:



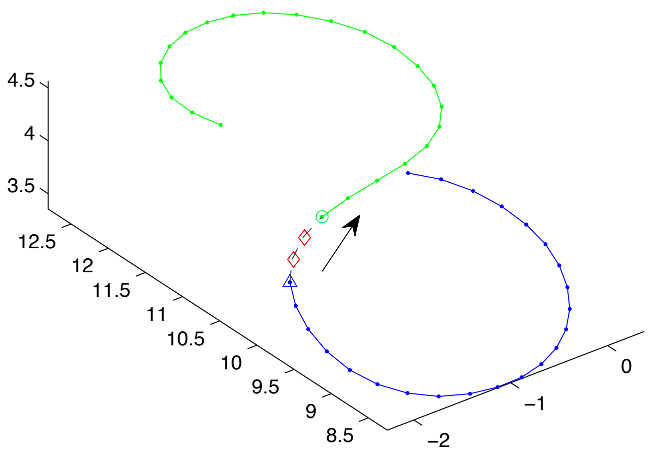
(a) Elastic L-Score

(b) Stitching path

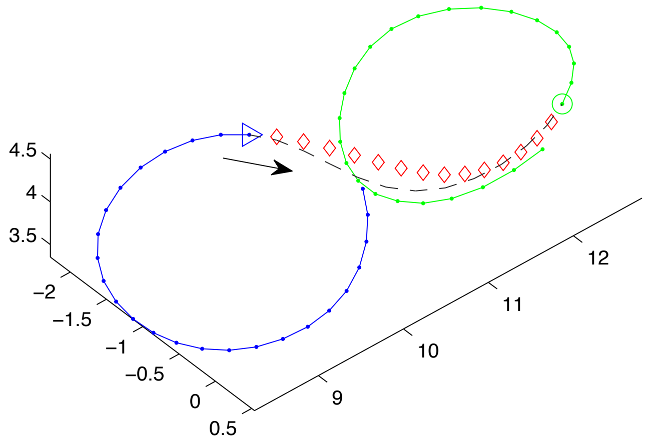
Figure 9.8: Top shows the elastic L-Score versus k (number of injected frames). The starting ("query") motion is the same, 'AB' (as in Figure 9.1), and the data motions correspond to the "forward" and the "backward" cases with both landing on optimal positions. Bottom shows the generated paths for the corresponding k_{opt} optimal number of injected frames. Notice the asymmetric landing positions and that the forward transition has lower elastic L-Score, as well as it needs fewer injected frames ($k = 3$, vs $k = 6$), agreeing with human intuition.

- We show how to compute two dynamics-aware distance functions, the L-Score and the *elastic* L-Score. Among the many possible choices, we recommend the Kalman filter with Newtonian particle dynamics to estimate velocities and accelerations required to compute the L-Score.
- Our technique allows for *elastic* stitching, where we automatically compute the optimal *duration* of a transition. Optimality, again, is judged by the total required effort.
- In experiments on both artificial and real motions, the L-Score chooses good stitching points and produces natural-looking trajectories.

Although our algorithm works well as is, and is simple to implement, it uses a rough (particle-based) approximation of character dynamics for state estimation. Improving this approximation, perhaps by using full-body dynamics and a nonlinear filter, is an interesting direction for future work.



(a)



(b)

Figure 9.9: Real motion stitching: Right-hand coordinates of a human transition motion, with the dashed part blanked out (2 blank-out frames for the left figure, 11 for the right). △/○ marks the take-off/landing frame, respectively. Red ◇ stand for our reconstructed path using *elastic* L-Score; notice how close they are to the ground truth (gray dashed line). The *elastic* L-Score either finds the correct k_{opt} (=2 in left) or gets very close (=14, vs 11, in right)

Chapter 10

Human Motion Occlusion Filling

Chapter 3 has introduced a general method for filling missing values in time series, while this chapter will focus on the occlusions in motion sequences. Moreover, this chapter will take one step forward and demonstrate how to exploit domain knowledge to leverage the capability of statistical models.

Given a motion capture sequence with occlusions, how can we recover the missing values, respecting bone-length constraints? The DynaMMo method introduced in Chapter 3, which works well, except for occasionally violating such constraints, and thus lead to unrealistic results. Our main contribution is a principled approach for preserving such distances. Specifically (a) we show how to formulate the problem as a constrained optimization problem, using two variations: hard constraints, and soft constraints; (b) we show how to efficiently solve both variations; (c) we demonstrate the realism of our approaches against competitors, on real motion capture data, illustrating that our 'soft constraints' version eventually produces more realistic results (Figure 10.1).

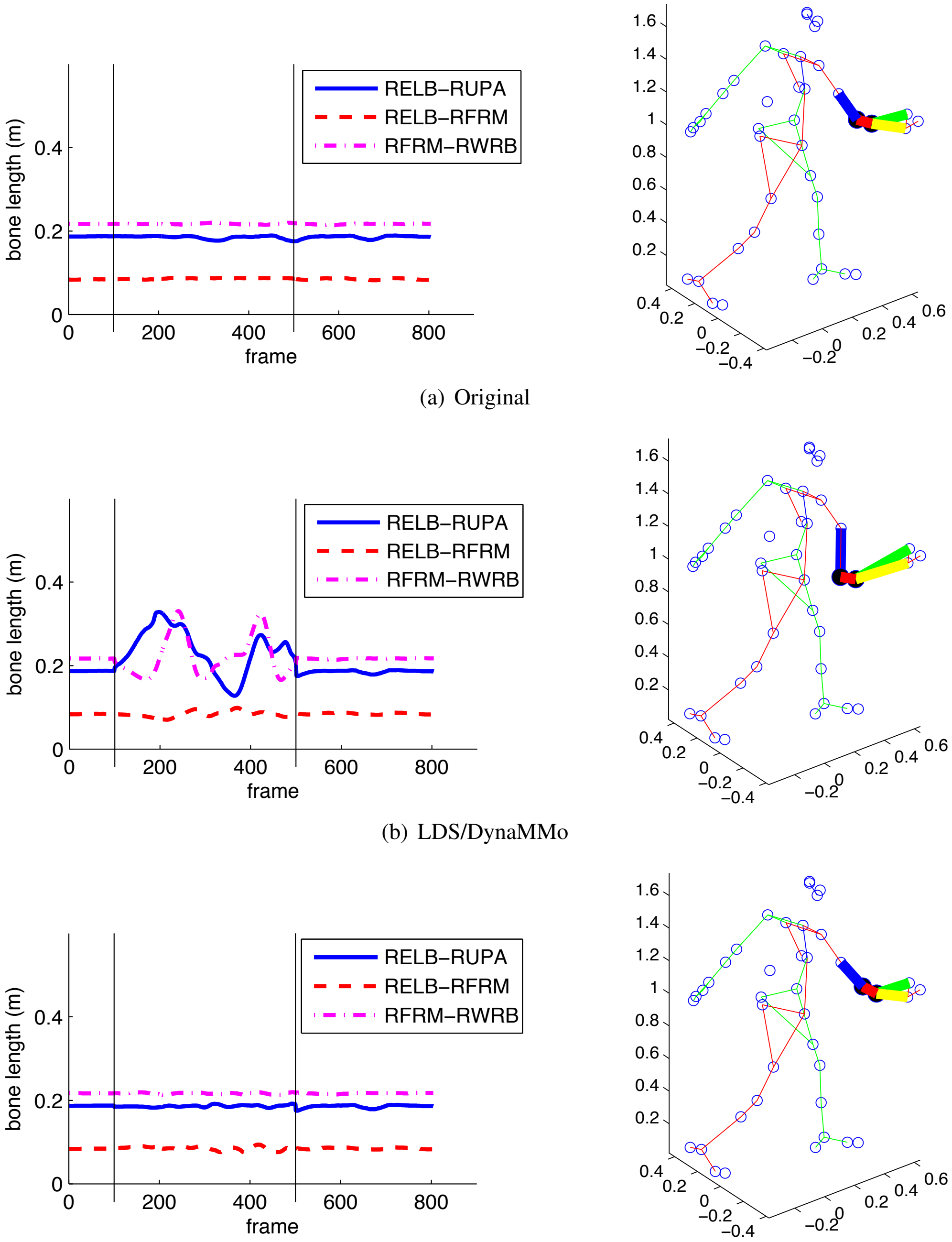
10.1 Introduction

Given motion capture data, with occlusion, how can we recover the missing values, so that we obey bone-length constraints?

Optical motion capture is a useful method for computer animation. In this technique, cameras are used to track reflective markers on an actor's body, and the pose of the actor is reconstructed from these marker positions (Figure 9.2). Figure Of course, such systems are not infallible, and inevitably some markers

Table 10.1: Comparison of Occlusion Filling Methods

Advantages Method	Bone Length constraints	Black-out
Spline	×	✓
MSVD	×	×
LDS/DynaMMo	×	✓
BoLeRO	✓	✓



(a) Original

(b) LDS/DynaMMo

(c) BoLeRO

Figure 10.1: Reconstructing two marker positions on the right arm from frame 100 to 500 of a walking motion (#132.43). Graphs show bone lengths over time; markers are stills at frame 241. LDS/DynaMMo ((b)) fails to preserve inter-marker distances present in the original motion. We propose BoLeRO which does much better ((c)).

cannot be tracked due to occlusions or awkward camera placement. Similarly, one could imagine concatenating two such sequences of marker motion and treating the "transition" region as a large tracking failure.

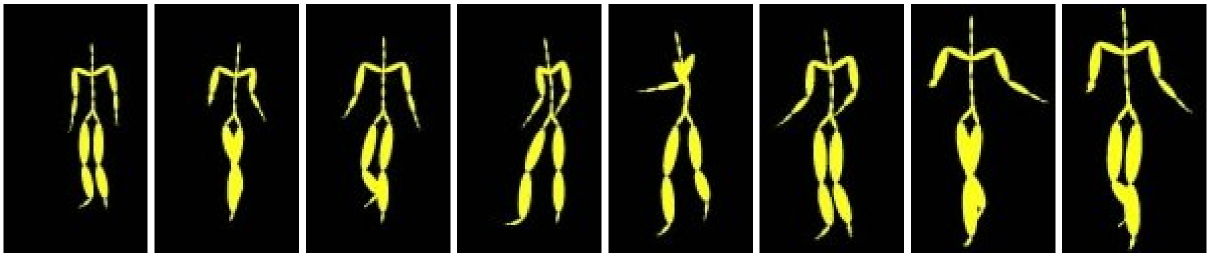


Figure 10.2: Animated film strips of a walking motion for Figure 10.1.

Currently such occlusions are filled manually or through ad-hoc methods. Straightforward methods, like linear interpolation and spline interpolation, do not agree with human intuition, giving poor results. A more principled approach to occlusion filling would be to use a statistical model that accounts for correlations between markers and dynamics across time. One intuitive formulation is a linear dynamical system (LDS), which models observed data as noisy linear projections of low-dimensional state which evolves via noisy linear dynamics. We have already proposed a method based on LDS in Chapter 3, DynaMMo, for general missing values in time series.

One problem with DynaMMo/LDS in this setting is that they do not preserve inter-marker distances. While a joint-angle representation would solve this problem, it would both require that a skeleton be fit to the data (which would prevent LDS from being used for occlusion filling), and it would present a weight-selection challenge (a small angle error in the shoulder is much more noticeable than a small angle error in the wrist).

We show how to solve this problem in an explicit and principled manner, by specifying inter-marker distance constraints and learning an LDS that operates in this constrained space. The focus of our work is to handle occlusions automatically, agreeing with human intuition. Ideally we would like a method with the following properties:

1. Bone Length Constraint: It should be able to keep relative distance for markers on the same bone.
2. Black-out: It should be able to handle "black-outs", when all markers disappear (e.g., a person running behind a wall, for a moment).

Additionally, we also want our method to be scalable and automatic, requiring few (and, ideally, zero) parameters to be set by a human.

In this paper, we propose BoLeRO (**Bo**ne **le**ngth constrained **r**econstruction for **o**cclusion). Fig. 10.3 shows the reconstructed signal for an occluded running motion. Our method gives the best result close to the original value. Our main idea is to simultaneously exploit two properties: (a) body rigidness, through the bone length constraints; and (b) motion smoothness, by using the dynamical system to keep the moving trend. This two-prong approach can help us handle even "*black-outs*", which we define as time intervals where we lose track of all the markers.

The main contributions of this chapter are as follows:

1. We setup the occlusion filling problem and formulate the bone length constraints in a principled way.
2. We propose effective algorithms (Expectation-Maximization-Newton/Gradient) to solve the problem, yielding results agreeing with human intuition.

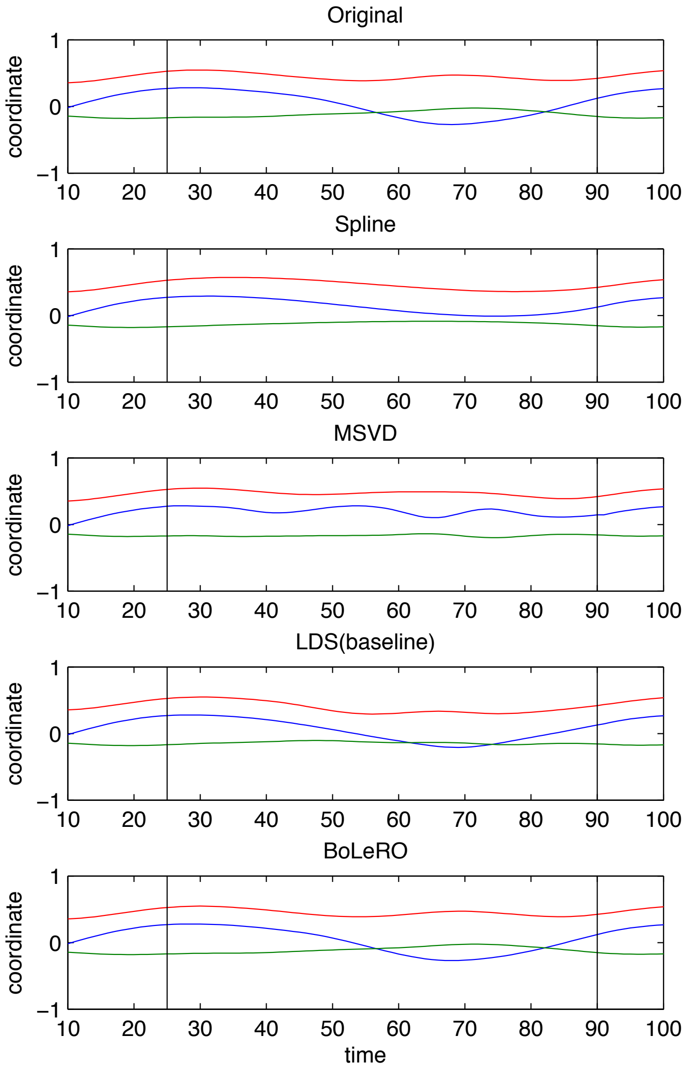


Figure 10.3: Original and reconstructed xyz-coordinates of the marker on right knee for a running motion (subject#127.07) with occlusion from time 25 to 90 (marked with vertical lines). This is 45% of 145 frames in total. x,y,z are in blue, green and red respectively. Top to bottom figures correspond to original motion, reconstruction from spline, MSVD, LDS and our proposed BoLeRO, respectively.

3. We perform experiments on real motion capture sequences to demonstrate the additional benefits from enforcing the bone length constraints.

The rest of the chapter is organized as follows: In Section 10.2, we review the related work; the proposed method and its discussion are presented in Section 10.3; the experimental results are presented in Section 10.4.

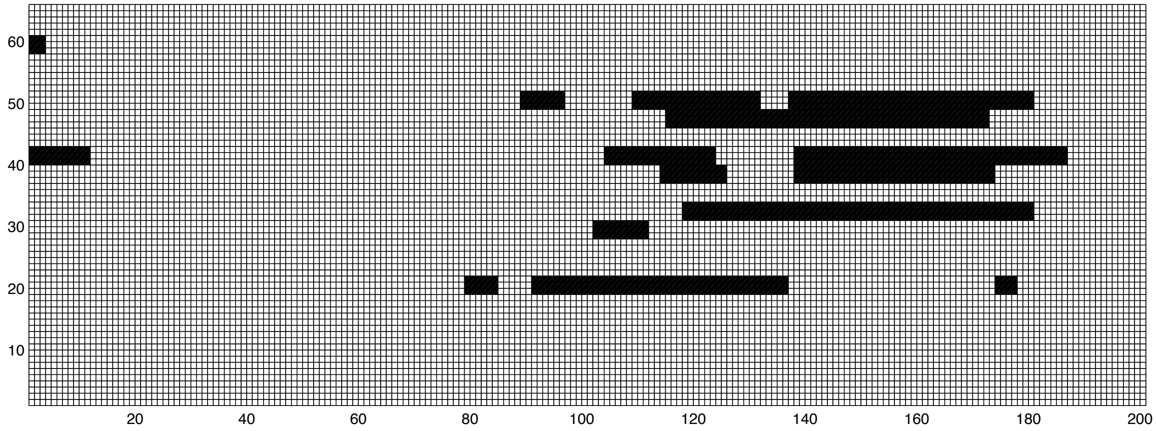


Figure 10.4: Occlusion in a handshake motion. 66 joint angles (rows), for ≈ 200 frames. Dark color indicates a missing value due to occlusion. Notice that occlusions are clustered.

10.2 Other approaches

Past occlusion filling methods and related techniques for occlusion filling can be classified into the following categories: (1) interpolation methods; (2) skeleton based [Herda et al., 2000, Zordan and Van Der Horst, 2003]; (3) dimensionality reduction and latent variables [Liu and McMillan, 2006, Park and Hodgins, 2006, Taylor et al., 2007]; (4) database backed [Hsu et al., 2004, Chai and Hodgins, 2005]; (5) dynamical systems [Wang et al., 2008, Li et al., 2009].

Interpolation methods: Linear interpolation and splines, are commonly used in time series smoothing and also motion capture systems to handle missing markers. These interpolation methods are generally effective for short period occlusions or occasional missing markers.

Skeleton based methods: Herda et al [Herda et al., 2000] used a human body skeleton to track and reconstruct the 3-d marker positions. Their method could predict the position using three previous markers by calculating the kinetics, when a marker is missing. Markers in motion capture system are usually captured in 3D space, while many applications work in joint angle space, thus a mapping from raw 3D data to joint angles is often required. Instead of fixed skeleton, Kirk et al [Kirk et al., 2005] proposed a method to automatically construct structural skeleton from motion capture data. Zordan and Van Der Horst [Zordan and Van Der Horst, 2003] used a fixed limb-length skeleton to map the markers on full body to a representation in joint angles plus reference body center. Our proposed approach works directly in 3D space, there it does not require such mapping. These skeleton methods could work well for short segment of occlusions, however our method could handle much longer occlusions, as well as black-outs.

Dimensionality reduction and latent variable models: Liu and McMillan [Liu and McMillan, 2006] proposed a method that projects the markers into linear principal components and reconstructs the missing parts from the linear models. This approach is similar to the Missing Value SVD (MSVD) [Srebro and Jaakkola, 2003] with single iteration. Furthermore, they proposed an enhanced Local Linear method from a mixture of such linear models. Park and Hodgins [Park and Hodgins, 2006] also used PCA to estimate the missing markers for skin deformation capturing. There were also work on nonlinear models for human motion. Taylor et al [Taylor et al., 2007] used a conditional restricted Boltzmann machine (CRBM) with discrete hidden states to model human motion. Their approach could learn non-linear binary representations for a motion frame, conditioned on several previous frames. Therefore, their method could fill in the missing values from the prediction of several previous frames.

Database backed approach: Hsu et al [Hsu et al., 2004] proposed a method to map from a motion control specification to a target motion by searching over patterns in existing database. Chai and Hodgins [Chai and Hodgins, 2005] used a small set of markers as control signals and reconstruct the full body motion from a pre-recorded database. These methods could also generate motions follow the bone lengths, however they are repetitions or interpolated motions taken directly from database, thus could not generate new motions. The subset of markers should be known in advance, while our method does not assume fixed subsets of the dimension observed or missing.

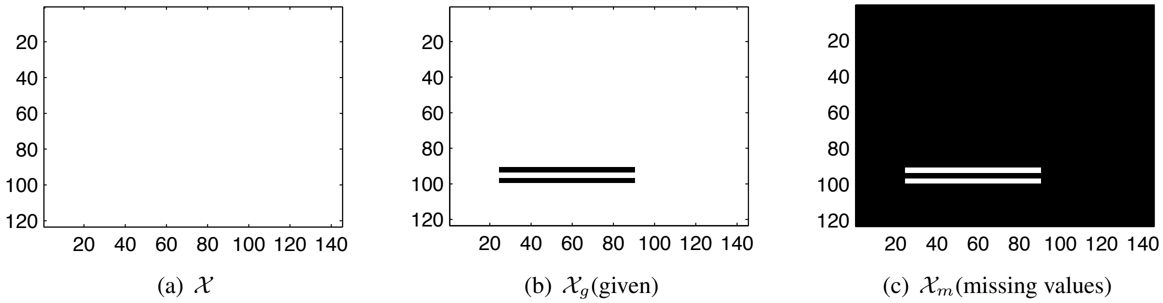
Dynamical systems: Previously, Kalman filters were used for tracking system [Dorfmüller-Ulhaas, 2003] with carefully defined parameters. Shumway and Stoffer [Shumway and Stoffer, 1982] proposed EM algorithm to learn the model parameters. Wang et al [Wang et al., 2008] took a nonparametric approach to model human motion and proposed a Gaussian process dynamical model, which includes a chain of latent variables and nonlinear mapping from the latent space to observed motion. In case of missing observation, it could use the learned model to estimate expectation of missing markers. Aristidou et al [Aristidou et al., 2008] used Kalman filters to predict the missing markers, with parameters conforming to Newton dynamics. Recently, Li et al [Li et al., 2009] used Linear Dynamical Systems to model motion capture data, and proposed an algorithm to recover the missing values. We will use it as the baseline method for comparison and describe it in more details in Section 10.3.1.

Liu and McMillan [Liu and McMillan, 2006] provide a nice summary of related work on occlusion for motion capture data as well as of techniques for related tasks such as motion tracking.

Comparing against all these methods, our proposed BoLeRO can (a) capture the coupling between multiple markers like dimensionality reduction does; (b) it can generate motions that follow the dynamics of natural human motion like LDS/Kalman filters do; (c) it can enforce inter marker distances for those on the same bone (exactly or with a small toleration) as the skeleton method does; (d) it models the motion as a whole, instead of treating each frames individually, and thus BoLeRO is able to use the observed portion as much as possible. That is, each previous method exhibits only one or two of the above properties, but not all of them. This is the intuition why our BoLeRO method achieves better recovery of occlusions.

10.3 Occlusion filling with bone length constraints

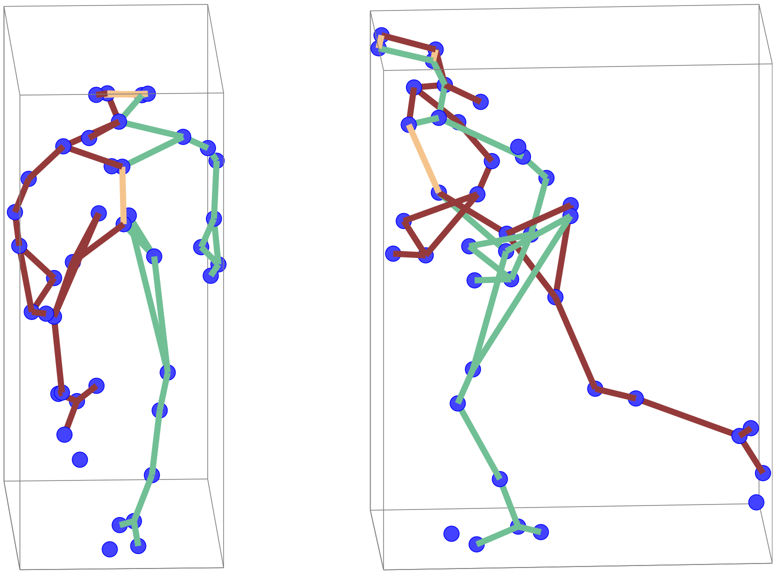
A typical motion capture system usually uses optical cameras to track passive markers in 3-D space. Mathematically, the observations of marker positions form a multi-dimensional time series, denoted as $\mathcal{X}$ (a $\mathrm{T} \times m$ matrix). In case of marker occlusion (denoted as $\mathcal{X}_m$, the set of variables that are missing from the $\mathrm{T} \times m$ matrix $\mathcal{X}$), the goal is to fill in the blanks to reconstruct most natural motion according to the human eye. The relevant symbols are described in Table 10.2.



(a) $\mathcal{X}$ (b) $\mathcal{X}_g$(given) (c) $\mathcal{X}_m$(missing values)

Figure 10.5: Illustration of data matrices $\mathcal{X}$, $\mathcal{X}_g$ and $\mathcal{X}_m$, from left to right, respectively. Rows are marker positions, and columns are time-ticks (frames). Variables contained in the respective matrices are shown in white; the set of all variables $\mathcal{X}$ is divided into the known ones $\mathcal{X}_q$ and the missing ones $\mathcal{X}_m$

Our method is motivated by observing the important role of rigid bones in human body that preserve relative positions of markers: markers on the same bone will maintain a given distance between them. We capture these fixed distances through bone length constraints. Our approach addresses the occlusion filling problem by enforcing the bone length constraints on top of a traditional dynamical system for motion capture data. Figure 10.6 shows a body skeleton in human motion capture and their bones on which the markers are located.



(a) Front view (b) Side view

Figure 10.6: Human body skeleton: solid balls show the markers and lines indicate bones.

How should we incorporate BLC into the dynamic systems? There are two choices with varying implications: namely, hard constraints and soft constraints. The former exerts exact inter-marker distances for same bone markers, while the latter general follows the constraint while allowing occasional violations. We will describe each of them and compare their effectiveness in later sections. Here we first define the meaning of bone length in our context.

Definition 10.1. *A set B lists bone length constraints (BLC), and it contains the following elements:*

$$B = \{\langle i, j, d_{i,j} \rangle | \textit{marker } i, j \textit{ on the same bone}\}$$

where $d_{i,j}$ is the distance between marker i and j.

Definition 10.2. *A matrix W is said to indicate the missing observation if*

$$W(t,i) = \begin{cases} 1 & i\textit{-th marker missing at time } t, \\ 0 & i\textit{-th marker observed at time } t \end{cases}$$

Definition 10.3. *A pair of marker i and j are said to conform to the BLC B if their coordinates of all time tick follows*

$$\langle i, j, d_{i,j} \rangle \in B \Rightarrow \|\vec{x_t}^{(i)} - \vec{x_t}^{(j)}\|^2 = d_{i,j}^2$$

where $\vec{x_t}^{(i)}$ denotes the coordinates for i-th marker at time t.

Table 10.2: Symbols, Acronyms and Definitions

Symbol	Definition
$\mathcal{X}$	a motion sequence with missing values $(\vec{x}_1, ...\vec{x}_\mathrm{T})$ of T time-ticks in m dimensions
$\mathcal{X}_g$	the observed values in a motion sequence $\mathcal{X}$
$\mathcal{X}_m$	variables for the missing values in a motion sequence $\mathcal{X}$
$\vec{x_t^{(i)}}$	position of marker i at time t
$\vec{z_t}$	hidden variables at time t
W	01 matrix indicating missing values (1=missing)
m	number of dimensions (e.g., marker positions) - m=123 here
H	number of hidden dimensions
BLC	bone length constraint
EMN/EMG	expectation, maximization and Newton/gradient descent

10.3.1 Background

Linear Dynamical Systems (LDS, also known as Kalman filters) are commonly used in motion tracking systems. The basic idea is to identify the hidden variables (e.g. velocity, acceleration) in observed data (marker positions) and build a statistical model to characterize the transitions of hidden variables. Such models could then reproduce the motion dynamics, as well as the correlations among markers by choosing a proper number of hidden variables. In LDS, a multi-dimensional motion sequence is modeled with a hidden Markov chain, as follows (for readability, repetition of Eq (2.4-2.6)).

$$\vec{z}_1 = \vec{\mu}_0 + \omega_0 \tag{10.1}$$

$$\vec{z}_{n+1} = \mathbf{F} \cdot \vec{z}_n + \omega_n \tag{10.2}$$

$$\vec{x}_n = \mathbf{G} \cdot \vec{z}_n + \epsilon_n \tag{10.3}$$

where $\Theta = \{\vec{\mu}_0, \Gamma, \mathbf{F}, \Lambda, \mathbf{G}, \Sigma\}$ is the set of parameters. $\vec{\mu}_0$ is initial state of the whole system, $\vec{x}_n$ and $\vec{z}_n$ denote marker coordinates and hidden variables at time n, respectively. $\mathbf{F}$ implies the transition dynamics and $\mathbf{G}$ is the observation projection. ω_0, ω_i and $\epsilon_i (i = 1 \ldots \mathrm{T})$ are multivariate Gaussian noises with the following distributions:

$$\omega_0 \sim \mathcal{N}(0, \Gamma) \;\; \omega_i \sim \mathcal{N}(0, \Lambda) \;\; \epsilon_j \sim \mathcal{N}(0, \Sigma) \tag{10.4}$$

In Chapter 3, we have already discussed using linear dynamical systems for general missing values in time series. In the case of missing observations, Li et al [Li et al., 2009] proposed DynaMMo, an expectation-maximization (EM) like algorithm to recover the marker positions by estimating the expectation of occluded values, given the observed parts, $\mathbb{E}[\mathcal{X}_m|\mathcal{X}_g]$. The algorithm introduced in Chapter 3 finds solutions to maximize the expected log-likelihood with respect to the model parameters, the hidden variables and the missing observations as well. However, the method would have a hard time for multiple markers of long occlusions as we point out the experiments. In the following sections, we will show how to improve the reconstruction of missing values with the help of domain knowledge in human motion case.

10.3.2 BoLeRO-HC (hard constraints)

Intuition: The traditional method to estimate the missing values in the sequence data is to minimize a squared loss function on Eq (10.1)-(10.4), penalizing the model complexity. While a LDS based method such as DynaMMo [Li et al., 2009] could recover short occlusion, it suffers in cases of longer and multiple marker occlusions, because reconstructed markers may break rigid bodies and violate bone length constraints. Our proposed method is based on the basic intuition on human motion: those markers attached to the same bones should not fall apart. Our main contribution in the current chapter is to demonstrate the usefulness of domain knowledge, in this case bone length constraints in occlusion filling. Following this intuition, we propose BoLeRO-HC (hard constraints), enforcing the exact bone length constraints in a LDS-based model. We make a first conjecture, bone length constraints will help recover motion occlusions better, in addition to traditional dynamics information as modeled by LDS. Here we first formulate the domain knowledge and define the bone length constraints.

Problem formulation: We will first present the proposed BoLeRO-hard constraint here. With the bone length constraints (BLC), we link the naturalness of a motion to whether it conforms to desired bone lengths under temporal movement. In our model, we assume the motion is moving according to the linear dynamical systems. Given an occluded motion sequence $\mathcal{X}$, occlusion indication matrix W and an additional BLC B, the problem is to fill in the occlusion so that the resulting motion follows the moving dynamics as captured by LDS and conforms to the bone length constraints. Our proposed BoLeRO will recover the missing markers by estimating the "good" expectation $\mathbb{E}[\mathcal{X}_m|\mathcal{X}_g]$, which conforms to the inter-marker distances on the same bone. The sequence of marker coordinates are modeled based on LDS as above (Eq. (10.1)-(10.3)) with additional constraints. Mathematically, the occlusion filling problem and its cost function are defined as follows.

Problem 10.1 (BoLeRO, hard constraints)**.** *Given (a) $\mathcal{X}_g$ (the observed marker positions), (b) B (bone length constraint), and (c) occlusion indication matrix W, find Θ and X_m to solve the following objective function:*

$$\begin{aligned} \min \quad & Q(X_m, \Theta) \\ \textit{subject to} \quad & \|x_t^{(i)} - x_t^{(j)}\|^2 - d_{i,j}^2 = 0 \quad \forall \langle i, j, d_{i,j} \rangle \in B \end{aligned} \tag{10.5}$$

with the objective function $Q(\cdot)$ to be

$$Q(X_m, \Theta) = \frac{1}{2}\mathbb{E}\Bigg[(\vec{z}_1 - \vec{\mu}_0)^T\Gamma^{-1}(\vec{z}_1 - \vec{\mu}_0) + \sum_{t=2}^{T}(\vec{z}_t - \mathbf{F}\cdot\vec{z}_{t-1})^T\Lambda^{-1}(\vec{z}_t - \mathbf{F}\cdot\vec{z}_{t-1}) + \sum_{t=1}^{T}(\vec{x}_t - \mathbf{G}\cdot\vec{z}_t)^T\Sigma^{-1}(\vec{x}_t - \mathbf{G}\cdot\vec{z}_t)\Bigg] + \frac{\log|\Gamma|}{2} + \frac{T-1}{2}\log|\Lambda| + \frac{T}{2}\log|\Sigma| \quad (10.6)$$

where $\vec{x}_t^{(i)}$ is coordinates of i-th marker at time t, and Θ denotes model parameters $\Theta = \{\mathbf{F}, \mathbf{G}, \vec{z}_0, \Gamma, \Lambda, \Sigma\}$.

Algorithm: The main goal of the learning algorithm is to find optimal values for $\mathcal{X}_m$ such that the objective function $Q(\mathcal{X}_m, \Theta)$ is minimized under the bone length constraints. To solve the optimization problem, we observe that the constraints have nothing to do with the hidden variables and model parameters Θ, which suggests the following coordinate-descent style algorithm. At the high level, our proposed algorithm optimizes the parameters and unknowns piece-wisely and iteratively through the "EMN" procedure (*E*xpectation, Maximization, and Newton optimization), as shown in Algorithm 10.1: We use Expectation-Maximization [Li et al., 2009] to estimate the posterior distribution $P(\mathcal{Z}|\mathcal{X})$, its sufficient statistics ($\mathbb{E}[\vec{z}_t]$, $\mathbb{E}[\vec{z}_t\vec{z}_t^T]$ and $\mathbb{E}[\vec{z}_t\vec{z}_{t-1}^T]$), and Θ respectively (steps 10.1 and 10.1 in Algorithm 10.1), and fill in missing values using Newton's method to solve the Lagrange derived from the BLC (step 10.1 in Algorithm 10.1). Finally, we update the model parameters Θ by maximizing the log likelihood (i.e. minimizing Q), and iterate until convergence.

In more detail, our proposed BoLeRO-HC uses Lagrange multipliers to handle constraints for frames with missing values. The Lagrangian is given by:

$$\mathcal{L}(\mathcal{X}_m, \Theta, \vec{\eta}) = \frac{1}{2}\mathbb{E}\Bigg[(\vec{z}_1 - \vec{z}_0)^T\Gamma^{-1}(\vec{z}_1 - \vec{z}_0) + \sum_{t=2}^{\mathrm{T}}(\vec{z}_t - \mathbf{F}\vec{z}_{t-1})^T\Lambda^{-1}(\vec{z}_t - \mathbf{F}\vec{z}_{t-1}) + \sum_{t=1}^{\mathrm{T}}(\vec{x}_t - \mathbf{G}\vec{z}_t)^T\Sigma^{-1}(\vec{x}_t - \mathbf{G}\vec{z}_t)\Bigg] + \frac{1}{2}\log|\Gamma| + \frac{\mathrm{T}-1}{2}\log|\Lambda| + \frac{\mathrm{T}}{2}\log|\Sigma| + \sum_{t=1}^{\mathrm{T}}\sum_{\langle i,j,d_{i,j}\rangle\in B}\eta_{tij}(\|\vec{x_t^{(i)}} - \vec{x_t^{(j)}}\|^2 - d_{i,j}^2) \quad (10.7)$$

where $\vec{\eta} = \{\eta_{tij}\}$ are Lagrange multipliers. Note here we also include the dummy Lagrange multipliers for observed markers, however, because since those marker positions are known, it will not affect the result.

Algorithm: To derive a solution for the constrained optimization problem, we follow the "EMN" guideline, expectation ($P(\mathcal{Z}|\mathcal{X})$), maximization ($\Theta$), and Newton optimization ($\mathcal{X}_m$): we first take derivative of $\mathcal{L}$ with respect to Θ, yielding the forward-backward belief propagation (also known as Kalman filtering and smoothing) for expectation step and maximization equations. Since Θ is not involved in constraints, the resulting updating equations can be derived in a similar way as DynaMMo in Chapter 3. Here we summarize the key steps in EMN approach (Algorithm 10.1):

1. E-step: fix Θ and missing $\mathcal{X}_m$, using Kalman filtering and Kalman smoothing to estimate posterior $P(\mathcal{Z}|\mathcal{X};\Theta)$,

2. M-step: update the model parameters Θ,

3. N-step: Fix Θ, estimate the missing $\mathcal{X}_m$ under hard constraints using Newton's method, with the previously computed $P(\mathcal{Z}|\mathcal{X};\Theta)$.

The algorithm then iterates over E-, M-, and N-steps until convergence.

To estimate $\mathcal{X}_m$ and η's, we use Newton's method to iteratively search and minimize the objective function with respect to the constraints. The optimal solution is specified by critical point, which requires $\frac{\partial \mathcal{L}}{\partial \mathcal{X}_m} = 0$ and $\frac{\partial \mathcal{L}}{\partial \eta} = 0$. In this step, we iteratively update the $\mathcal{X}_m$ and η in the Newton's descent direction. Let $x_t^{\vec{M}}$ and η_t denote the unobserved marker positions and Lagrangian multipliers at time t, respectively. During each iteration, we update them according to the following update rules:

$$\begin{pmatrix} x_t^{\vec{M}} \\ \eta_t \end{pmatrix}^{new} \longleftarrow \begin{pmatrix} x_t^{\vec{M}} \\ \eta_t \end{pmatrix} - \alpha(\nabla^2_{x_t^{\vec{M}},\eta_t} L)^{-1} \cdot \nabla_{x_t^{\vec{M}},\eta_t} \mathcal{L} \tag{10.8}$$

where $\nabla_{x_t^{\vec{M}},\eta_t} \mathcal{L}$ is the partial gradient and $\nabla^2_{x_t^{\vec{M}},\eta_t} \mathcal{L}$ is the Hessian matrix.

BoLeRO-HC uses Newton's method to iteratively search the optimal solution through the update Eq. 10.8. The partial gradient is given by:

$$\nabla_{x_t^{\vec{M}},\eta_t} = \begin{pmatrix} I_t^M \Sigma^{-1} \cdot (\vec{x_t} - \mathbf{G} \cdot \mathbb{E}[\vec{z_t}]) + 2\sum_{i,j} \delta_B(t,i,j)\eta_{ijt}(x_t^{\vec{(i)}} - x_t^{\vec{(j)}}) \\ (\|x_t^{\vec{(i)}} - x_t^{\vec{(j)}}\|^2 - d_{i,j}^2) \quad \text{for all } \delta_B(t,i,j) = 1 \end{pmatrix} \tag{10.9}$$

where I_t^M is a $(0,1)$-matrix, with each row has exactly one nonzero elements corresponding to the missing marker indices at time t. The auxiliary function $\delta_B(t,i,j)$ is defined as

$$\delta_B(t,i,j) = \begin{cases} 1 & \text{if} \langle i,j,d_{i,j} \rangle \in B \wedge (W_{t,i} = 1 \vee W_{t,j} = 1) \\ 0 & \text{otherwise} \end{cases}$$

By further taking partial derivative, we can get the Hessian,

$$\nabla^2_{x_t^{\vec{M}},\eta_t} = \frac{\partial \nabla_{x_t^{\vec{M}},\eta_t}}{\partial (x_t^{\vec{M}},\eta_t)^T} \tag{10.10}$$

Details are straightforward and omitted for brevity.

Algorithm 10.1: EMN algorithm for BoLeRO-HC

```
Input: $X_g$, $B$ (BLC), $W$ (missing indication matrix), $H$ (hidden dimension)
Output: Recovered motion sequence: $\hat{X}$
// initialization
$X_m \leftarrow 0$;                                          // or other initialization
$\hat{X} \leftarrow X_g \cup X_m$;
$\mathbf{F}, \mathbf{G}, \vec{\mu_0} \leftarrow$ random values;
$\Gamma, \Lambda, \Sigma \leftarrow I$;
$\Theta \leftarrow \{\mathbf{F}, \mathbf{G}, \vec{\mu_0}, \Gamma, \Lambda, \Sigma\}$;
repeat
    E-step: estimate the posterior $P(Z|\hat{X};\Theta)$ and its sufficient statistics $\mathbb{E}[\vec{z}_t|\hat{X};\theta]$, $\mathbb{E}[\vec{z}_t\vec{z}_t^{\,T}|\hat{X};\theta]$,
    and $\mathbb{E}[\vec{z}_t\vec{z}_{t-1}^{\,T}|\hat{X};\theta]$ using belief propagation (Eq.(10.18-10.33));
    M-step: Minimizing Eq (10.7) with respect to $\Theta$ (Eq.(10.36-10.40))
        $$\Theta^{new} \leftarrow \arg\min_{\Theta} \mathcal{L}(X_m, \Theta, \vec{\eta})$$
    for $t \leftarrow 1$ to T do
        // N-step:  estimating missing using Newton's method
        $k \leftarrow$ number of missing markers at time $t$;
        $\eta_t \leftarrow (\underbrace{0, \ldots, 0}_{k})^T$;
        $\alpha \leftarrow 1/2$ ;                               // step size
        repeat
            $D \leftarrow \nabla_{x_t^{\vec{M}}, \eta_t} \mathcal{L}(\cdot)$;          // Page.151 Eq.(10.9)
            $H \leftarrow \nabla^2_{x_t^{\vec{M}}, \eta_t} \mathcal{L}(\cdot)$;         // Page.151 Eq.(10.10)
            $\vec{y} \leftarrow \hat{X}_t^{(i)} \quad \forall i.W_{t,i} = 1$ ;
            $\begin{pmatrix} \vec{y} \\ \eta_t \end{pmatrix}^{new} \longleftarrow \begin{pmatrix} \vec{y} \\ \eta_t \end{pmatrix} - \alpha H \cdot D \; \hat{X}_t^{(i)} \leftarrow \vec{y} \quad \forall i.W_{t,i} = 1$ ;
        until converge ;
    end
until converge ;
```

Discussion: There are several subtle points in the above algorithm.

- BLC optimization order: We randomly pick an order to optimize with respect to the bone length constraints. While it will not affect the result in the ideal case, such a way will improve the stability of solutions in practical motions where even markers on the same bone slightly change a tiny epsilon due to the skin movement or measurement noise.

- Choosing H: There are several ways to choose a proper H, e.g. cross validation. In our setting we choose the number of hidden dimensions so that we keep over 95% of energy in original data ($H = 15$ in experiments).

- Convergence criteria: The convergence of the algorithm is determined by either reaching a maximal number of iterations ($= 10,000$ in experiments) or changes in objective function less than certain threshold ($= 10^{-6}$ in experiments).

10.3.3 BoLeRO-SC (soft constraints)

Intuition - Motivation The hard constraint formulation above would be ideal, except that in reality, markers slightly move, and reality itself *violates* the bone length constraints (BLC)! In such cases, hard constraints may land to a solution with abrupt discontinuity in the recovered marker position, albeit the corresponding marker distances are well preserved.

This implies we do not have to underscore the exact preservation of the bone length, while the ideal system should allow some "reasonable" variation. The intuition comes naturally, the recovered missing values should be a trade off between approaching the maximum likelihood and preserving bone length. To alleviate this problem, we relax the bone constraints and instead solve the following soft constrained problem. Our objective is the likelihood, with additional penalty on the deviation from the desired bone length of those missing markers on the same bones.

Problem Formulation Following the intuition, we get the following objective function: the negative log likelihood penalized by the deviation from the bone length constraints.

$$\begin{aligned}
\min \quad & f(X_m, \Theta) \\
=& \frac{1}{2}\mathbb{E}\left[(\vec{z_1} - \vec{\mu_0})^T \Gamma^{-1} (\vec{z_1} - \vec{\mu_0}) + \sum_{t=2}^{T} (\vec{z_t} - \mathbf{F} \cdot \vec{z_{t-1}})^T \Lambda^{-1} (\vec{z_t} - \mathbf{F} \cdot \vec{z_{t-1}}) \right. \\
& \left. + \sum_{t=1}^{T} (\vec{x_t} - \mathbf{G} \cdot \vec{z_t})^T \Sigma^{-1} (\vec{x_t} - \mathbf{G} \cdot \vec{z_t}) \right] + \frac{\log|\Gamma|}{2} + \frac{T-1}{2}\log|\Lambda| + \frac{T}{2}\log|\Sigma| \\
& + \frac{\lambda}{2}\sum_{t=1}^{\mathrm{T}} \sum_{\langle i,j,d_{i,j}\rangle \in B} (W_{t,i}|W_{t,j})(\|\vec{x_t^{(i)}} - \vec{x_t^{(j)}}\|^2 - d_{i,j}^2)^2
\end{aligned} \tag{10.11}$$

where $W_{t,i}|W_{t,j} = W_{t,i} + W_{t,j} - W_{t,i}W_{t,j}$.

Algorithm: To solve the optimization problem, we propose a coordinate descent approach (Expectation-Maximization-Gradient), which alternately optimizes over a set of unknown variables or parameters (see Algorithm 10.2):

1. E-step: fix Θ and missing $\mathcal{X}_m$, using Kalman filtering and Kalman smoothing to estimate posterior $P(\mathcal{Z}|\mathcal{X};\Theta)$,
2. M-step: update the model parameters Θ,
3. G-step: Fix Θ, estimate the missing $\mathcal{X}_m$ under soft constraints using gradient descent, with the previously computed $P(\mathcal{Z}|\mathcal{X};\Theta)$.

By taking partial derivatives over Θ and setting to zero, we obtain the same equations for E-step and M-step as in above EMN for hard constraints. While N-step is replaced with the gradient descent on soft constraints. The update rule for G-step is:

$$\vec{x_t^{(i)}} \leftarrow \vec{x_t^{(i)}} - \alpha \cdot \frac{\partial f}{\partial \vec{x_t^{(i)}}} \tag{10.12}$$

where $\vec{x_t^{(i)}}$ denotes the i-th marker coordinates at time t.

$$\frac{\partial f}{\partial \vec{x_t^{(i)}}} = I_{(i)} \cdot \Sigma^{-1} \cdot (\vec{x_t} - \mathbf{G} \cdot \mathbb{E}[\vec{z_t}]) + 2\lambda \sum_{\langle i,j,d_{i,j} \rangle \in B} (W_{t,i} | W_{t,j})(\|\vec{x_t^{(i)}} - \vec{x_t^{(j)}}\|^2 - d_{i,j}^2)(\vec{x_t^{(i)}} - \vec{x_t^{(j)}}) \tag{10.13}$$

The learning rate depends on the proper choosing of the learning step size α. We developed an adaptive scheme for adjusting α according to value of the objective function. The basic idea is to enlarge α whenever the objective decreases and to shrink α whenever it increases.

To make the scheme work, we observe that the partial derivative $\frac{\partial f}{\partial \vec{x_t^{(i)}}}$ is independent of all the rest time ticks. So in our algorithm, we isolate the optimization for each time tick, and adaptively choose the learning rate for that time tick. Specifically, we define the following time-decomposed objective function:

$$f_t(x_t) = \frac{1}{2} \mathbb{E}[(\vec{x_t} - \mathbf{G} \cdot \vec{z_t})^T \Sigma^{-1} (\vec{x_t} - \mathbf{G} \cdot \vec{z_t})] + \frac{\lambda}{2} \sum_{\langle i,j,d_{i,j} \rangle \in B} (W_{t,i} | W_{t,j})(\|\vec{x_t^{(i)}} - \vec{x_t^{(j)}}\|^2 - d_{i,j}^2)^2 \tag{10.14}$$

Observing $\frac{\partial f_t}{\partial \vec{x_t^{(i)}}} = \frac{\partial L}{\partial \vec{x_t^{(i)}}}$, the update rule becomes

$$\vec{x_t^{(i)}} \leftarrow \vec{x_t^{(i)}} - \alpha \cdot \frac{\partial f_t}{\partial \vec{x_t^{(i)}}} \tag{10.15}$$

The adaptive gradient descent method works as follows: it only accepts the update when the update will decrease the time-decomposed objective function f_t, doubling α in this case; otherwise halving α.

Algorithm 10.2: EMG algorithm for BoLeRO-SC

```
Input: $X_g$, $B$ (BLC), $W$ (missing indication matrix), $H$ (hidden dimension)
Output: Recovered motion sequence: $\hat{X}$
// initialization
$X_m \leftarrow 0$;                                    // or other initialization
$\hat{X} \leftarrow X_g \cup X_m$;
$\mathbf{F}, \mathbf{G}, \vec{\mu_0} \leftarrow$ random values;
$\Gamma, \Lambda, \Sigma \leftarrow I$;
$\Theta \leftarrow \{\mathbf{F}, \mathbf{G}, \vec{\mu_0}, \Gamma, \Lambda, \Sigma\}$;
repeat
    E-step: estimate the posterior $P(Z|\hat{X};\Theta)$ and its sufficient statistics $\mathbb{E}[\vec{z_t}|\hat{X};\theta]$, $\mathbb{E}[\vec{z_t}\vec{z_t}^T|\hat{X};\theta]$,
    and $\mathbb{E}[\vec{z_t}\vec{z_{t-1}}^T|\hat{X};\theta]$ using belief propagation (Eq.(10.18-10.33));
    M-step: Minimizing Eq (10.11) with respect to $\Theta$ (Eq.(10.36-10.40))
        $$\Theta^{new} \leftarrow \arg\min_{\Theta} f(X_m, \Theta)$$
    for $t \leftarrow 1$ to T do
        // G-step: estimating missing using gradient descent
        $\alpha \leftarrow 1$ ;                                  // step size
        repeat
            foreach $i$ s.t. $W(t,i) = 1$ do
                $D \leftarrow \frac{\partial f}{\partial \vec{x_t^{(i)}}}$ ;               // Eq.(10.13)
                $(\vec{x_t^{(i)}})^{new} \leftarrow \vec{x_t^{(i)}} - \alpha \cdot D$
            end
            if $f_t((\vec{x_t})^{new}) < f_t(\vec{x_t})$ then accept update
                update $\hat{X}$ with $(\vec{x_t})^{new}$;
                $\alpha \leftarrow 2\alpha$;
            elsereject update
                $\alpha \leftarrow \frac{\alpha}{2}$;
            end
        until converge ;
    end
until converge ;
```

10.4 Evaluation

We performed experiments on real human motion capture data to evaluate the effectiveness of our proposed method.

We used a public dataset from CMU mocap database [CMU, a]. Each motion consists of 200 to 1500 frames and 123 features of marker positions (41 markers), converted to body local coordinates by estimating root position and body facing. We rescaled units into meters, which will improve the computation stability since all numbers are within the range of [-2, 2].

For each of the motion in our trial, we create its bone length constraints, B, by estimating the average inter-marker distances (e.g. marker RTHI and RSHN are on the same bone). Alternatively, we can construct the BLC by estimating the variance of the inter marker distances and thresholding, or algorithms by Kirk et al [Kirk et al., 2005] and de Aguiar [de Aguiar et al., 2006], however bone length estimation is beyond the scope of our thesis. For both baseline and BoLeRO, we set the hidden dimension $H = 16$, which is corresponding to over 95% of energy in original data. We set $\lambda = 10^6$ for BoLeRO-SC in our experiments.

To evaluate the effectiveness of our proposed methods BoLeRO-HC and BoLeRO-SC, we select a trial set with 9 motions representing a variety of motion types, including running, walking, jumping, sports, and martial art. We did a statistical test as well as case studies. In statistical test, we randomly occluded a marker for a random consecutive segment, and tested the reconstruction with all candidate methods. Each trial is repeated 10 times with a different random occlusion. Fig. 10.9 shows reconstruction mean square error (Eq. 10.16) against the original motion. Notice BoLeRO-SC consistently has lower mse over the baseline LDS/DynaMMo while BoLeRO-HC occasionally does.

$$\text{mse} = \frac{\sum_{t,i} W_{t,i}(\hat{X}_{t,i} - X_{t,i}^{true})^2}{\sum_{t,i} W_{t,i}} \tag{10.16}$$

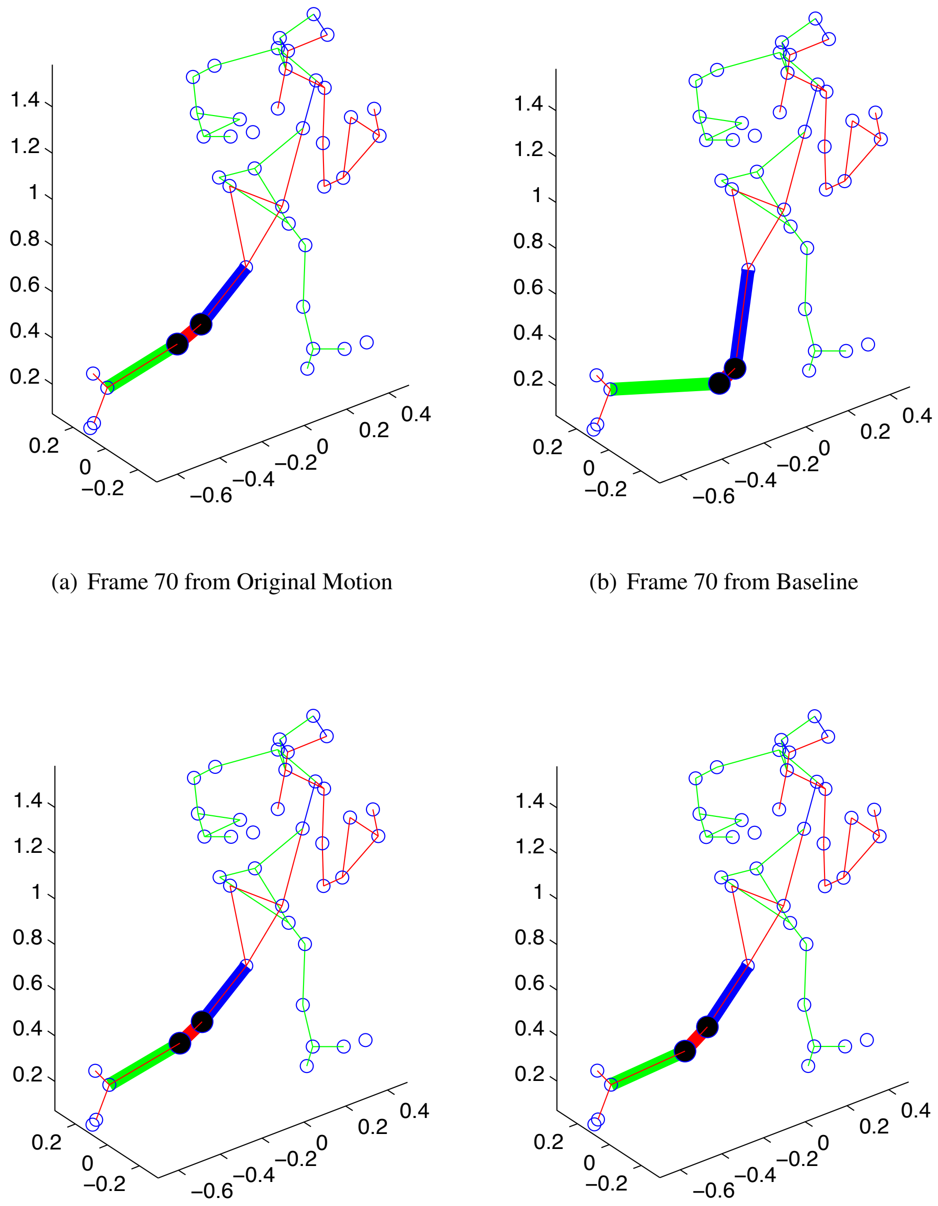
where W indicates missing markers (if $= 1$).

We also test the methods on two or more markers missing as well. Fig. 10.8 shows a case of running motion (subject#127.07) and reconstruction results by two methods. Two markers on the right leg (RKNE for knee and RSHN for shin) are occluded from frame 25 to 90 inclusive. Fig. 10.3 shows the time plot of coordinates for one marker on the right knee, where spline and MSVD clearly deviates much from the original data, hence we did not include them in the following distance plot. Fig. 10.8(a)-10.8(c) show the distances between the two markers and adjacent markers on the body skeleton (thigh-to-knee, knee-to-shin and shin-to-ankle in blue, red and green respectively, see Figure 10.7 for typical frames). All three distances should ideally be constant. The result generated by baseline method violates the bone constraint (particularly around frame 70), while BoLeRO clearly improves the quality of reconstruction by obeying the corresponding BLC. Additional results and animations are shown in the accompanying video.

10.5 Summary

In this chapter, we focus on the problem of occlusion in motion capture data, and specifically on the reconstruction so that we obey bone length constraints.

Motion capture is a useful technique to obtain realistic motion animation, where optical cameras are used to track marker movement on actors. Unfortunately markers will be out of view sometimes, especially in full body motions like running, football playing and bounce walking, etc., and it takes hours/days for human experts to manually fix the gaps. How can we handle the occluded motion and fill in the gaps automatically and effectively, while respecting bone-length constraints? In this chapter, we propose BoLeRO, a principled approach to reconstruct occluded motion using bone length constraints on body dynamics. The novelty is that it sets up the problem as a linear dynamical system with constraints, thus



(a) Frame 70 from Original Motion

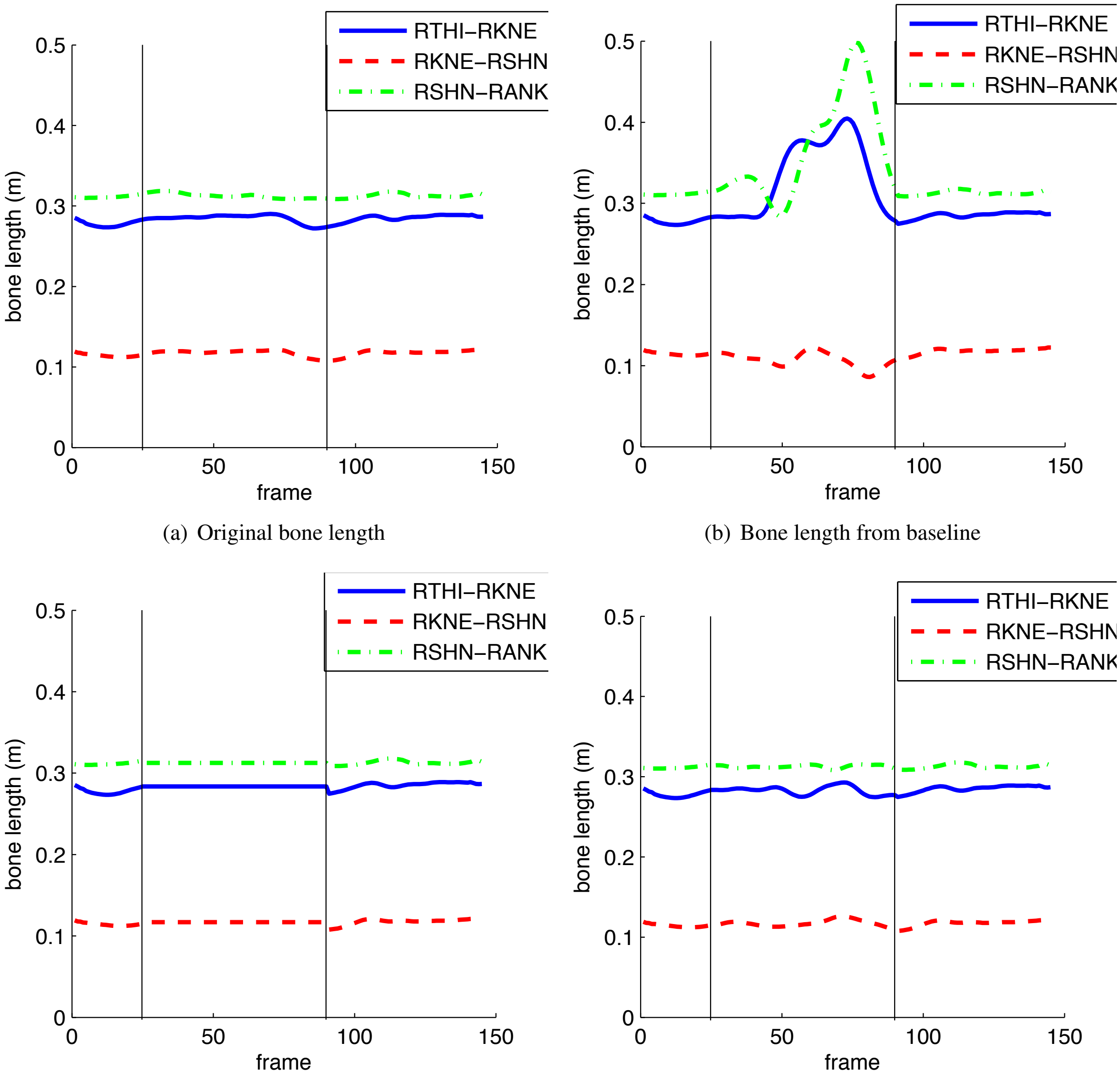
(b) Frame 70 from Baseline

(c) Frame 70 from BoLeRO-HC

(d) Frame 70 from BoLeRO-SC

Figure 10.7: One typical frame in an occluded running motion (subject #127.07) and the recovered ones. Markers articulated with circles. Bold lines illustrate the bones of interest.

explicitly exploiting both the smoothness in motion dynamics, as well as the rigidness in distances between relevant markers. We give two versions of it: "hard constraints" (BoLeRO-HC), and "soft constraints" (BoLeRO-SC), where the reconstructed bone-lengths may slightly differ from the ideal ones.



(a) Original bone length

(b) Bone length from baseline

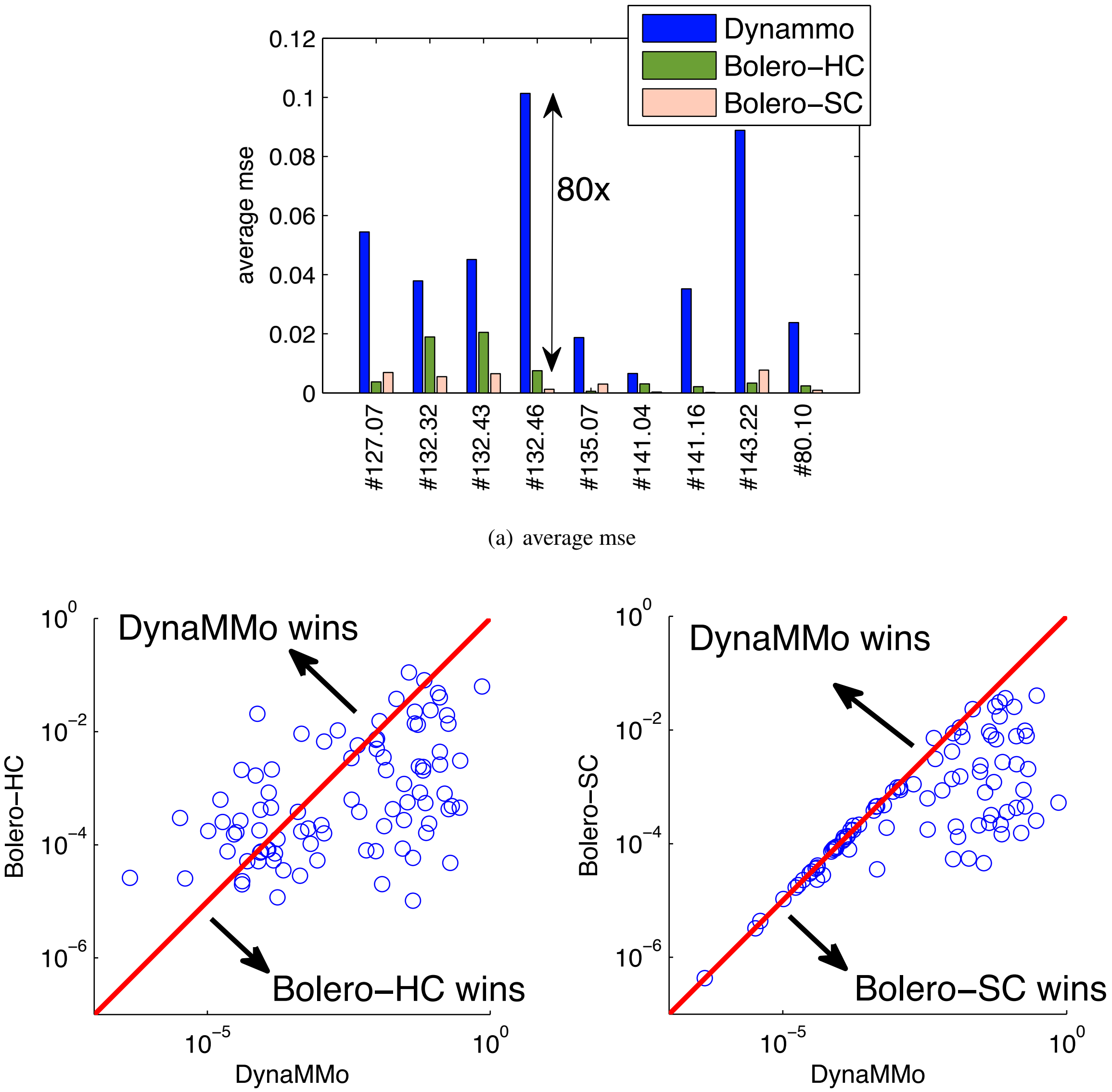
(c) Bone length from BoLeRO-HC

(d) Bone length from BoLeRO-SC

Figure 10.8: Recovery results for an occluded running motion (subject #127.07). Figures show bone lengths in original, baseline, BoLeRO-HC, and BoLeRO-SC, for thigh-to-knee (blue), knee-to-shin (red) and shin-to-ankle(green) respectively. Notice that the original, BoLeRO-HC, BoLeRO-SC lengths are near constant while the baseline has a severe violation of BLC.

The second contribution is that we propose a novel, fast algorithm to solve both versions of the problem, using our "EMN/EMG" formulation (expectation, maximization, Newton/gradient descent): The idea is to alternatingly estimate (a) the hidden variables (b) the parameters of the Linear Dynamical System and (c) the Lagrange multipliers (only in BoLeRO-HC) and missing values; and iterate until convergence.

Experiments on real data show that either version of BoLeRO are significantly better than straightforward alternatives (splines, linear interpolation) and they matches or outperforms very sophisticated alternatives like Kalman filters and recent missing-value algorithms [Srebro and Jaakkola, 2003, Li et al., 2009].



(a) average mse

(b) DynaMMo v.s. BoLeRO-HC

(c) DynaMMo v.s. BoLeRO-SC

Figure 10.9: Comparison between baseline(LDS/DynaMMo), BoLeRO-HC, and BoLeRO-SC. 10.9(a) average mse for LDS/DynaMMo(blue), BoLeRO-HC(green), BoLeRO-SC(orange). 10.9(b),10.9(c) scatter plots of mse's in 90 trials on 9 motions. Our BoLeRO-SC consistently wins over DynaMMo (see (c) - all points are at or below diagonal), with a maximum of 80x improvement (see (a)), while BoLeRO-HC loses occasionally (see (b), points above diagonal).

Among our two version, we recommend the "soft constraints" BoLeRO, which overwhelmingly outperforms the 'hard constraints' version.

10.A Appendix: Details of the learning algorithm

Both versions of BoLeRO require inference about the hidden states and estimation of the new parameters. Those inference can be derived in a similar way as DynaMMo in Chapter 3.

10.A.1 Inference: Forward-backward message passing

Given the parameters $\Theta = (\mathbf{F}, \mathbf{G}, \vec{\mu_0}, \Gamma, \Lambda, \Sigma)$, the estimation problem is finding the marginal distribution for hidden state variables given the observed data, e.g. $\hat{\vec{z}_t} = \mathbb{E}[\vec{z}_n \mid \mathcal{X}](n = 1, \dots, \mathrm{T})$.

Assume the posterior up to current time tick is $p(\vec{z}_n|\vec{x}_1, \dots, \vec{x}_n$, denoted by:

$$\hat{\alpha}(\vec{z}_n) = \mathcal{N}(\mu_n, \mathbf{V}_n) \tag{10.17}$$

We could obtain the following forward passing of the belief. The messages here are $\vec{\mu}_n$, $\mathbf{V}_n$ and $\mathbf{P}_{n-1}$(needed in later backward passing).

$$\begin{align}
\mathbf{P}_{n-1} &= \mathbf{F}\mathbf{V}_{n-1}\mathbf{F}^T + \Lambda \tag{10.18}\\
\mathbf{K}_n &= \mathbf{P}_{n-1}\mathbf{G}^T(\mathbf{G}\mathbf{P}_{n-1}\mathbf{G}^T + \Sigma)^{-1} \tag{10.19}\\
\vec{\mu}_n &= \mathbf{F}\vec{\mu}_{n-1} + \mathbf{K}_n(\vec{x}_n - \mathbf{G}\mathbf{F}\vec{\mu}_{n-1}) \tag{10.20}\\
\mathbf{V}_n &= (\mathbf{I} - \mathbf{K}_n)\mathbf{P}_{n-1} \tag{10.21}\\
& \tag{10.22}
\end{align}$$

The initial messages are given by:

$$\begin{align}
\mathbf{K}_1 &= \Gamma\mathbf{G}^T(\mathbf{G}\Gamma\mathbf{G}^T + \Sigma)^{-1} \tag{10.23}\\
\vec{\mu}_1 &= \vec{\mu_0} + \mathbf{K}_1(\vec{x}_1 - \mathbf{G}\vec{\mu_0}) \tag{10.24}\\
\mathbf{V}_1 &= (\mathbf{I} - \mathbf{K}_1)\Gamma \tag{10.25}\\
& \tag{10.26}
\end{align}$$

For the backward passing, let $\gamma(\vec{z}_n)$ denote the marginal posterior probability $p(\vec{z}_n|\vec{x}_1, \dots, \vec{x}_\mathrm{T})$ with the assumption:

$$\gamma(\vec{z}_n) = \mathcal{N}(\hat{\mu}_n, \hat{\mathbf{V}}_n) \tag{10.27}$$

The backward passing equations are:

$$\begin{align}
\mathbf{J}_n &= \mathbf{V}_n\mathbf{F}^T(\mathbf{P}_n)^{-1} \tag{10.28}\\
\hat{\vec{\mu}}_n &= \vec{\mu}_n + \mathbf{J}_n(\hat{\vec{\mu}}_{n+1} - \mathbf{F}\vec{\mu}_n) \tag{10.29}\\
\hat{\mathbf{V}}_n &= \mathbf{V}_n + \mathbf{J}_n(\hat{\mathbf{V}}_{n+1} - \mathbf{P}_n)\mathbf{J}_n^T \tag{10.30}
\end{align}$$

From the passed belief, we could obtain the following estimation:

$$\begin{align}
\mathbb{E}[\vec{z}_n] &= \hat{\vec{\mu}}_n \tag{10.31}\\
\mathbb{E}[\vec{z}_n\vec{z}_{n-1}^T] &= \mathbf{J}_{n-1}\hat{\mathbf{V}}_n + \hat{\vec{\mu}}_n\hat{\vec{\mu}}_{n-1}^T \tag{10.32}\\
\mathbb{E}[\vec{z}_n\vec{z}_n^T] &= \hat{\mathbf{V}}_n + \hat{\vec{\mu}}_n\hat{\vec{\mu}}_n^T \tag{10.33}
\end{align}$$

where the expectations are taken over the posterior marginal distribution $p(\vec{z}_n|\vec{x}_1, \dots, \vec{x}_\mathrm{T})$.

10.A.2 Parameter estimation

The new parameter Θ^{new} is obtain by maximizing $\mathcal{L}$ in Eq. 10.7 with respect to the components of Θ^{new} given the current estimate of Θ^{old}.

$$Q(\Theta^{new}, \Theta^{old}) = \mathbb{E}_{\Theta^{old}}[\log p(\vec{y}_1 \dots \vec{y}_N, \vec{z}_1 \dots \vec{z}_N | \Theta^{new})] \tag{10.34}$$

Taking the derivatives and let them be zeros gives the following results:

$$\begin{aligned}
\vec{\mu}_0^{new} &= \mathbb{E}[\vec{z}_1] && (10.35)\\
\Gamma^{new} &= \mathbb{E}[\vec{z}_1 \vec{z}_1^T] - \mathbb{E}[\vec{z}_1]\mathbb{E}[\vec{z}_1^T] && (10.36)\\
\mathbf{F}^{new} &= (\sum_{n=2}^{\mathrm{T}} \mathbb{E}[\vec{z}_n \vec{z}_{n-1}^T])(\sum_{n=1}^{\mathrm{T}-1} \mathbb{E}[\vec{z}_n \vec{z}_n^T])^{-1} && (10.37)\\
\Lambda^{new} &= \frac{1}{\mathrm{T}-1} \sum_{n=2}^{\mathrm{T}} (\mathbb{E}[\vec{z}_n \vec{z}_n^T] - \mathbf{F}^{new}\mathbb{E}[\vec{z}_{n-1} \vec{z}_n^T] \\
&\quad - \mathbb{E}[\vec{z}_n \vec{z}_{n-1}^T](\mathbf{F}^{new})^T + \mathbf{F}^{new}\mathbb{E}[\vec{z}_n \vec{z}_{n-1}^T](\mathbf{F}^{new})^T) \\
& && (10.38)\\
\mathbf{G}^{new} &= (\sum_{n=1}^{\mathrm{T}} \vec{x}_n \mathbb{E}[\vec{z}_n^T])(\sum_{n=1}^{\mathrm{T}} \mathbb{E}[\vec{z}_n \vec{z}_n^T])^{-1} && (10.39)\\
\Sigma^{new} &= \frac{1}{\mathrm{T}} \sum_{n=1}^{\mathrm{T}} (\vec{x}_n \vec{x}_n^T - \mathbf{G}^{new}\mathbb{E}[\vec{z}_n]\vec{x}_n^T \\
&\quad - \vec{x}_n \mathbb{E}[\vec{z}_n^T](\mathbf{G}^{new})^T + \mathbf{G}^{new}\mathbb{E}[\vec{z}_n \vec{z}_n^T](\mathbf{G}^{new})^T) && (10.40)
\end{aligned}$$

The resulting equations in E-step and M-step of EMN/EMG algorithms are traditionally known as Kalman filtering and Kalman smoothing. We would refer readers to [Kalman, 1960, Shumway and Stoffer, 1982, Ghahramani and Hinton, 1996] for more details.

Chapter 11

Data Center Monitoring

Efficient thermal management is important in modern data centers as cooling consumes up to 50% of the total energy. Unlike previous work, we consider *proactive* thermal management, whereby servers can predict potential overheating events due to dynamics in data center configuration and workload, giving operators enough time to react. However, such forecasting is very challenging due to data center scales and complexity. Moreover, such a physical system is influenced by cyber effects, including workload scheduling in servers. We propose ThermoCast, a novel thermal forecasting model to predict the temperatures surrounding the servers in a data center, based on continuous streams of temperature and airflow measurements. Our approach is (a) capable of capturing cyber-physical interactions and automatically learning them from data; (b) computationally and physically scalable to data center scales; (c) able to provide online prediction with real-time sensor measurements. The paper's main contributions are: (i) We provide a systematic approach to integrate physical laws and sensor observations in a data center; (ii) We provide an algorithm that uses sensor data to learn the parameters of a data center's cyber-physical system. In turn, this ability enables us to reduce model complexity compared to full-fledged fluid dynamics models, while maintaining forecast accuracy; (iii) Unlike previous simulation-based studies, we perform experiments in a production data center. Using real data traces, we show that ThermoCast forecasts temperature $2\times$ better than a machine learning approach solely driven by data, and can successfully predict thermal alarms 4.2 minutes ahead of time.

11.1 Introduction

A modern data center hosts tens of thousands of servers used to provide reliable and scalable infrastructure for Internet-scale services. The enormous amount (in the order of tens of megawatts) of energy these facilities consume and the resulting operational costs have spurred interest in improving their efficiency.

Traditional data centers are over-provisioned; server rooms (usually called colos) are excessively cooled and the average server utilization is kept quite low (e.g., CPU utilization between 10% to 30%). As a consequence, a "well tuned" data center rarely has thermal alarms and it is sufficient to use *reactive* thermal management, where data center operators take necessary actions only after an over-heated server issues a protective shutdown. However, such a conservative approach leads to waste of computational resources and poor Power Utilization Efficiency (PUE) [1] (close to 2, with $\approx 40\%$ of total data center

[1] PUE is defined as the ratio between total facility energy consumption and the energy used by servers.

energy used for cooling).

With increasing demand for improving data center efficiency, data center operators look into many ways to reduce cooling cost and increase server utilization. For example, a previous study confirms that fans consume most of the energy used by a Computer Room Air-Conditioning(CRAC) system [Liebert, 2008]. A single Liebert Deluxe System/3 CRAC installed in our data center has three 7.57 kW·h fans for a total energy consumption of 22.71kW·h [Liebert, 2007b]. Since the power that fan motors consume increases with the *cube* of fan rotation speed [Liebert, 2008], modern data centers use variable speed fans in order to reduce the CRAC's energy use: a mere 10% reduction in fan speed translates to 27% energy savings for the fan motor. Other energy-saving approaches taken by modern data centers include raising AC temperature set points, using outside air directly for cooling, consolidating workload using virtual machines, and leveraging statistical multiplexing to opportunistically oversubscribe the servers. However, as a result of such aggressive optimizations, the safety margin of data center operation is getting smaller. This trend requires data center monitoring to move from reactive to *proactive*, whereby the servers can predict potential overheating events early enough, giving operators enough time to react.

Central to any proactive thermal management approach is predicting temperature of different servers in a data center. This is extremely challenging due to large scale (a data center usually contains tens of thousands of servers, multiple CRAC units and fans), complex thermal interactions (e.g., due to server fans driving local air flow, by-pass air through gaps between servers and racks), and cyber effects (e.g., workload scheduling algorithms may have visible effects on temperature distribution). Previous works have considered two different approaches for data center thermal management. Thermodynamics-based solutions derive thermal models of different locations inside a data center using fundamental thermodynamics laws and data center layout [Bash and Forman, 2007, Moore et al., 2005, Patel et al., 2003b, Tang et al., 2008]. On the other hand, data-centric solutions use data-mining [Patnaik et al., 2009] or machine learning algorithms [Moore et al., 2006] to model and optimize cooling in a data center. All these existing solutions, however, provide static thermal models and are not adaptive to changes in workloads, CRAC fan speeds, data center layout, etc. Thus, these solutions are not adequate for modern data centers serving dynamic workload [Chen et al., 2008] or using power-efficient variable-speed fans.

In this chapter we propose ThermoCast, a novel thermal forecasting model that addresses the above limitation. ThermoCast uses real-time workload information and measurements from a carefully deployed set of temperature and air-flow sensors to model and predict temperatures around servers. We assume that each server knows temperature of its cold inlet air and hot exhaust air. These data can be obtained from temperature sensors shipped with some servers, or a RACNet-like data center sensor network [Liang et al., 2009]. Two big challenges in building any such real-time adaptive model are *predictability* and *scalability*: the model should be able to predict overheating early enough, without many false positives/negatives, even when system configuration (e.g., workload, fan speed) changes and it should be able to handle millions of data points to monitor within a data center. Reducing false positives/negatives is important to reduce burden on human operators who must take some action following an alarm and predicting early enough is important to give operators enough time to react.

To achieve high predictability, ThermoCast uses a hybrid approach of aforementioned thermodynamics-based and data-centric approaches. ThermoCast is based on thermodynamics laws and cyber-physical interactions, however, it learns and adapts appropriate values of various parameters from real-time sensor data and workload information. Thus, it is able to provide online prediction even when configurations, such as servers' on/off state, workload, set of servers, air-conditioning equipment maintenance, change.

To achieve scalability, we use the insight that temperature around a server is affected mostly by configurations of its neighboring servers and not much by the servers far from it. Therefore, ThermoCast is based on a zonal thermal model that builds a relationship among the cold-aisle vent temperature, the location of the server, the local temperature distribution and the workload from nearby servers, to predict the intake temperature at each server. Because of such local nature, ThermoCast can distribute the modeling task among servers: each server learns and models the temperature around itself by using nearby sensor measurements and workloads of neighboring servers. Thus, ThermoCast is computationally and physically scalable for a large data center.

We have deployed and evaluated ThermoCast in a lab data center with a rack of 40 servers. Through dense data center instrumentation, we show the complex thermal dynamics with variable workload and CRAC activities. Our experiments show that ThermoCast is more effective than pure machine learning approach with better prediction accuracy and mean lookahead time. For example, with real data traces, we show that ThermoCast can predict thermal spikes 4.2 minutes ahead of time, comparing to 2.3 minutes using an auto-regression (AR) model. The extra two minutes can be crucial for thermal management. Previous studies have shown that it takes about a minute to safely suspend a virtual machine in cloud computing environment [Zhao and Figueiredo, 2007, Swalin, 2010]. For connection intensive servers, like Windows Live Messenger, a minute can safely drain 7% of total TCP connections [Chen et al., 2008].

In summary, we make the following contributions in the chapter:

1. We provide a systematic approach to integrate the physical laws and sensor observations in a data center.
2. We provide an algorithm to learn from sensor data for such cyber-physical system, and it enables us to reduce complexity in full fluid models while still achieves good forecasting of future temperatures.
3. Unlike previous simulation-based studies, we perform experiments in a production data center. Using real data traces, we show that ThermoCast can forecast temperatures 2× better than the pure machine learning approach, and can successfully predict thermal alarms on average in 4.2 minutes ahead.

The rest of the chapter is organized as follows. We review the literature in Section 11.2 and summarize the operation and energy cost of data center cooling using air conditioners, and our findings about how temperature inside a data center changes as a function of server load and AC activity in Section 11.3. Section 11.4 describes the proposed ThermoCast framework, while Section 11.6 presents evaluation results.

11.2 Related work

Our work is related to two areas of interest, thermal management in data centers, time series mining and prediction.

Data center thermal management A number of recent papers have investigated methods for efficient thermal management in a data center. The methods can be broadly divided into two categories. The first category of solutions are based on fundamental thermal and air dynamics laws using computational fluid dynamic (CFD) simulators [Bash and Forman, 2007, Moore et al., 2005, Patel et al., 2003b, Ramos and Bianchini, 2008, Tang et al., 2008]. These solutions derive thermal models of different locations inside the data center during an initial profiling phase using data center layout and material thermo properties. The models are subsequently used by various energy-optimizing tasks. Cooling-aware workload placement algorithms [Bash and Forman, 2007, Moore et al., 2005, Patel et al., 2003b, Tang et al., 2008] use such models to place heavy computational workload in cooling-efficient locations. Energy-aware control algorithms [Ramos and Bianchini, 2008] use such models to choose the best dynamics voltage and frequency scaling (DVFS) policy for each server to match its workload. Spatio-temporal scheduling algorithms [Mukherjee et al., 2009] use the models with virtualization to improve cooling efficiency. Our work differs from these existing work in two important ways. First, rather than open-loop CFD models, ThermoCast is based on both thermodynamics laws and real-time measurements, and unlike previous solutions, it can adapt with dynamics in workload, fan speed, etc. Second, our focus is on predicting hot spots early enough, giving data center operators enough time to react. This requires ThermoCast to be scalable and predictable.

The second category of data center thermal management solutions use black-box data-driven approaches. Patnaik et al. [Patnaik et al., 2009] has proposed a temporal data mining solution to model and optimize performance of data center chillers, a key component of the cooling infrastructure. [Moore et al., 2006] proposed a thermal mapping prediction problem that learns the thermal map of a data center for different combinations of workload, cooling configurations, and physical topologies. The paper uses neural networks to learn this mapping from data derived from thermodynamic simulations of a data center. This model is then used for workload placement. This approach avoids the challenges of using thermodynamics to estimate server temperature through a data-driven approach that is amenable to online scheduling of computing workloads. However, the neural network is not dynamic and therefore temperature predictions might be affected by scheduling dynamics and lead to scheduling oscillations.

Mercury software suite [Heath et al., 2006] emulates single server temperatures based on utilization, heat flow, and air flow information. Mercury is then used by Freon, a system for managing thermal emergencies. Unlike Mercury and Freon, ThermoCast models the thermal relationship among nearby servers, which can be used to optimize computation and cooling.

Finally, work that proposes to improve data center energy efficiency through the use of low-power CPUs ([Grunwald et al., 2000]), smart cooling ([Patel et al., 2003a]), and power-efficient networking ([Heller et al., 2010]) is orthogonal to our work that provides methods to improve the efficiency of existing infrastructure through data-driven thermal modeling and thermal-aware dynamic workload placement.

Time series mining and prediction Since our data is collected from distributed sensors (temperature and airflow) in an online fashion. Our work also falls into the category of time series prediction. Autoregressive moving average (ARMA) are a standard family of models for time series analysis and forecasting (Box and Jenkins [Box et al., 1994]), and are discussed in every textbook in time series analysis and forecasting (e.g., [Brockwell and Davis, 1987]). Kalman filters and state-space models are also previously used in mining motion capture sequences and sensor data [Tao et al., 2004]. We use AR model as a baseline in our experiments.

In this chapter, we assume that we can obtain all sensor data. But one of the challenge in sensor data is the missing observations partly due to unreliability of wireless transmission. Li et al [Li et al., 2009] proposed DynaMMo method to learn a linear dynamical system in presence of missing values and fill in them. Their method could then use the learned latent variables to better compress the long time sequences.

Our system can leverage such approaches.

Remotely related is time series indexing, segmentation, classification [Gao et al., 2008, Tao et al., 2004] and outlier detection [Lee et al., 2008]. A common approach for indexing time series is extracting a few features from time sequences and matching them based on the features [Faloutsos and Lin, 1994], such as the Fourier transform coefficients, wavelet coefficients (Jahangiri et al. [Jahangiri et al., 2005]), and local dimensionality reduction (Keogh et al. [Keogh et al., 2001]). On the other hand, through a series of efforts, Keogh et al. developed methods to transform the continuous time series data into discrete representation (Symbolic aggregate approximation (SAX) [Lin et al., 2003]) and generalized these methods to index massive time sequences (iSAX [Shieh and Keogh, 2008]). Li et al. recently proposed PLiF method for time series classification by extracting a few compact features encoding the frequency, cross correlation and lag correlation [Li et al., 2010]. Time series indexing does not offer predictability, which is key in data center management scenarios.

11.3 Background and motivation

To understand the challenges of thermal prediction, we overview the operation of a typical data center including its cooling systems, and basic sensor instrumentation.

11.3.1 Data center cooling system

There are many data center architectures, from ad hoc server cabinets to dedicated containers. However, most enterprise and Internet data centers use a cold-aisle, hot-aisle cooling design. Figure 11.1 illustrates the cross section of a data center server room that follows this design. Server racks are installed on a raised floor in aisles. Cool air is blown by the CRAC (Computer Room Air Conditioning) system to the sub-floor. Perforated floor tiles act as vents, making cool air available to the servers. The aisles with these vents are called *cold aisles*. Typically, servers in the racks draw cool air from the front and blow hot exhaust air to the back in *hot aisles*. To effectively use the cool air, servers are arranged face to face in cold aisles. As Figure 11.1 suggests, cool and hot air eventually mix near the ceiling, and this return air is drawn back into the CRAC.

In its simplest form, a CRAC consists of two parts: a heat exchange unit and one or more fans. To handle the large cooling capacity requirement of a data center, the heat exchange unit typically uses a chilled water-based design. Specifically, the CRAC is connected to water chillers outside of the facility with circulation pipes. These pipes deliver chilled water with which the return air exchanges heat inside the CRAC. The warm water then circulates back to the outside water chillers. Finally, the cooled air is blown by the CRAC's fans to the floor vents. To reduce the energy consumption of the cooling equipment, many CRACs offer adjustable cooling capacity by adjusting the chilled water valve opening and the fan speed according to the return air temperature reported by the temperature sensor at the CRAC's air intake [Liebert, 2007a].

11.3.2 Data Center Sensor Instrumentation

Liu et al. argued about the benefits of using wireless sensor networks (WSNs) for data center monitoring including the ease of deployment in existing facilities with minimal infrastructure requirements [Liu et al.,

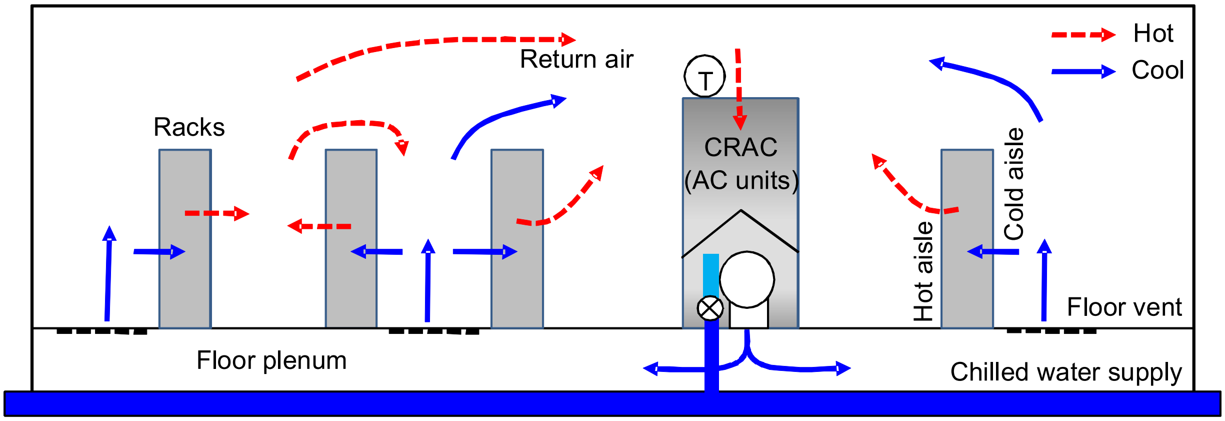


Figure 11.1: An illustration of the cross section of a data center. Cold air is blown from floor vents in cold aisles and hot air rises in hot aisles. Mixed air eventually returns to the CRAC where the chilled water cools the air.

2008]. In our case, using a WSN to measure temperature and airflow speeds across a data center allows us to quickly reconfigure the measurement harness as we vary measurement locations across different experiments.

We deployed a network of 80 sensor nodes at an university data center hosting a high-performance scientific computing cluster. The cluster consists of 171 1U [2] compute nodes with eight CPU cores each connected to two file servers through a low-latency InfiniBand switch. The sensor nodes are equipped with low-power 802.15.4 radios and form a multi-hop routing tree rooted at a gateway. The network comprised 15 air flow velocity sensors [Elektronik] and 65 humidity/temperature sensors [Sensirion, 2010].

We used this network to instrument three server racks according to the following sensor configuration. First, a rack is divided into three sections: *top* (i.e., four top most servers), *middle* (i.e., five middle servers) and *bottom* (i.e., four servers closest to the floor). During the experiments we control the load on the server at the middle of each section (termed as the *controlled server*). Servers in all three sections are instrumented with two humidity/temperature sensors: one at the server's air intake grill, facing the cold aisle, and another at the air ventilation grill in the hot aisle. Second, to measure the velocity of the cold air flowing from the floor vent at different heights, we positioned 12 air flow sensors directly above the floor vent at a vertical interval of 5.25", or every 3U (cf. Fig 11.2). Furthermore, we placed one air flow sensor at the air intake grill of each controlled server. Finally, we used the servers' built-in monitoring facilities to monitor their CPU load, fan speeds, and power consumption.

11.3.3 Observations

This section presents insights derived from the WSN measurements which provide both the motivation and the intuition to our control framework.

Figure 11.3 shows the relation between the cold air velocity from the floor vent and the range of server intake temperatures across a single rack. We make two observations from this figure.

[2] A *rack unit* or U is a unit of measure used to describe the height of equipment intended for mounting in a 19- or 23-inch rack. One rack unit is 1.75 inches.

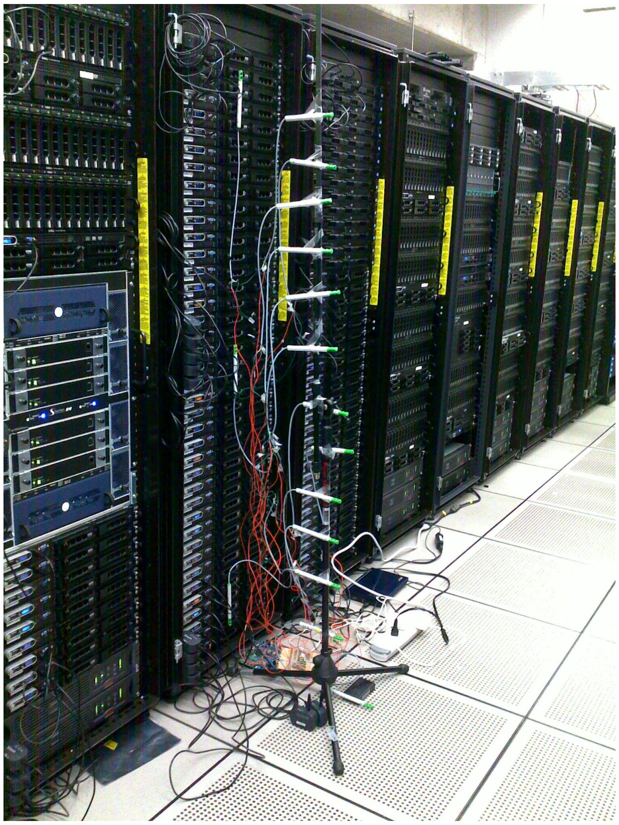


Figure 11.2: A picture of the air flow sensors setup. We positioned 12 air flow sensors directly above the floor vent at a vertical interval of 5.25”, or every 3U.

First, the temperature difference cycle (termed as the *contraction and relaxation* cycle) is in antiphase with the air velocity cycle. In other words, the temperature variation of a rack is smallest when the air velocity is highest. When the air velocity is low, the air is colder closer to the floor vent but less cool air is available at the top of the rack. Hence, the high and low temperature at the top and bottom section respectively are significantly different. At high air velocities, the top section cools down as cold air is forced further up, but the temperature of the bottom section actually increases due to the Bernoulli effect. The implication of this effect, which dictates that fast moving air creates a decrease in pressure, is that hot air from the back of the rack is drawn to the front of the server as the speed of cold air increases [Craig, 2003]. Therefore, simply increasing the CRAC fan speed can lead to unexpected hotspots.

Second, as the floor vent air velocity varies, the coldest section of the rack oscillates between the middle and the bottom section. On the other hand, the top section is almost always the hottest section. In addition to the fact that the CRAC needs to increase the fan speed to deliver cold air to the top section, the top section has a relatively higher initial temperature as it is close to the warm return air flow (cf. Figure 11.1).

Chen et al. suggested shutting down under-utilized servers to reduce the energy consumption of cooling system [Chen et al., 2008]. Intuitively, this approach applies well to servers in the top section which we just showed to frequently be the hottest. However, shutting down one server can impact the intake air temperature of its neighbors. Figure 11.4 illustrates an example of this interaction; shutting down the controlled server causes an increase in the intake air temperature of the server below it. While few servers

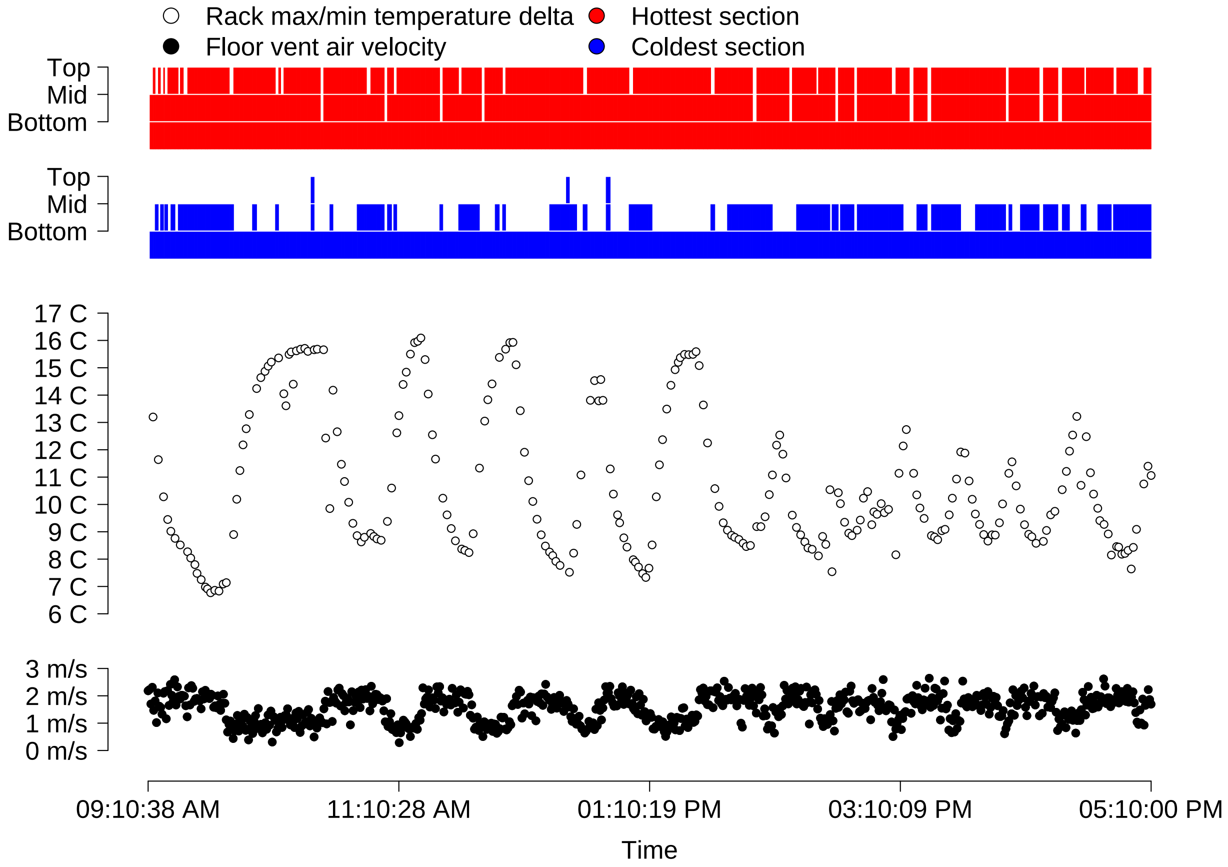


Figure 11.3: The relation between the cold air velocity from the floor vent and the server intake air temperatures of a single rack.

are affected by the actions of one server, a framework that predicts temperatures should consider these interactions.

11.4 ThermoCast Framework

As we have seen in Section 11.3, duty cycles of the CRAC system affect the amount of cooling that the different rack sections receive and thus affect the temperature at the servers' intakes. Furthermore, turning a server on or off affects its nearby servers and reaching a new equilibrium can take as long as an hour. ThermoCast faces the unique challenge of modeling the interaction between the computing and cooling systems. Formally, we define the problem as

Problem 11.1. *Given temperatures, airflow, and workload for every server in computer racks, along with their spatial layout, to forecast the future temperatures trend.*

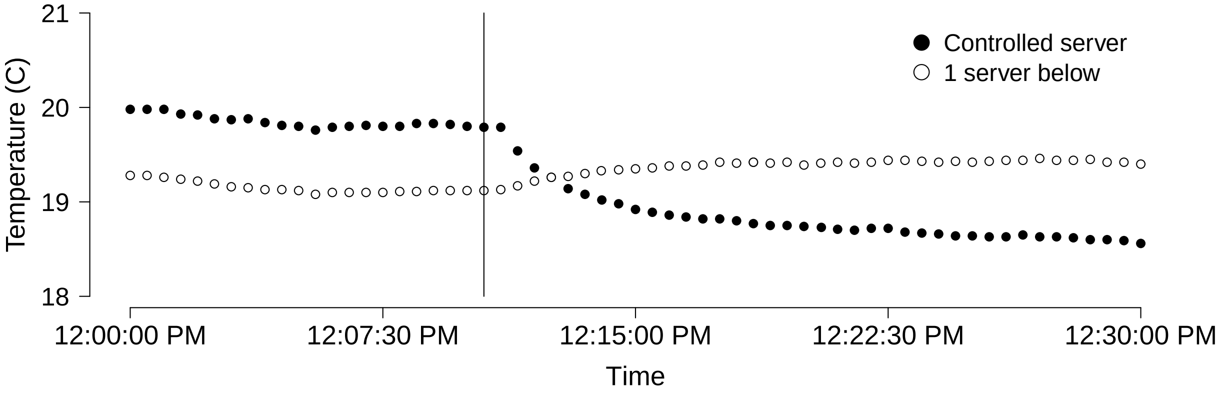


Figure 11.4: The intake air temperature change of the controlled server decreases after the server is shut down, while the temperature of the server below increases. The vertical line indicates when the controlled server was shut down.

11.4.1 Federated modeling architecture

The scale of mega data centers prevents us to use a centralized approach for model building and prediction. It is hard to even visualize tens of thousands of monitoring points on a screen. In ThermoCast, we use a federated modeling architecture that relies on each server to model its own thermal environment and make predictions. Only when local predictions exceed certain thresholds, the system draws the operators' attention, accumulates more sensor points, and possibly performs another tier of prediction and diagnosis.

The federated modeling architecture in ThermoCast takes advantage of the physical properties of heat generation and propagation. That is, heat diffuses locally and gradually following models of thermo- and fluid dynamics. Although the model parameters can be drastically different depending on the local configuration – rack heights, server locations, server types, on/off states etc. – the model structure remains the same. Based on this insight, we use a "gray-box" approach, in which the model is known but the parameters are unknown, as opposed to "white-box" modeling using CFD, and a completely data-driven "black-box" model such as neural networks.

Another advantage of the federated architecture is that model learning and prediction can be done in a distributed fashion. Figure 11.5 shows one section of the graphical model in ThermoCast. First of all, time is discretized into ticks. At every time tick, with step size t_s, server n uses its own intake and exhaust air temperatures, the intake and exhaust air temperatures of its immediate neighbor ($n-1$ and $n+1$), the air speed and temperature at the AC vent, and its own workload to build a model that computes its own intake and exhaust air temperature in the next time tick. The variable dependencies capture the air flow in different directions, as well as local heat generation.

Sensor data can be communicated efficiently in this architecture. If a wireless sensor network is used for monitoring, then each sensor only need to broadcast its value to the local neighbors. If the sensors are on the server chassis, then the data only needs to go through the local top-of-the-rack switch, rather than data center level routers.

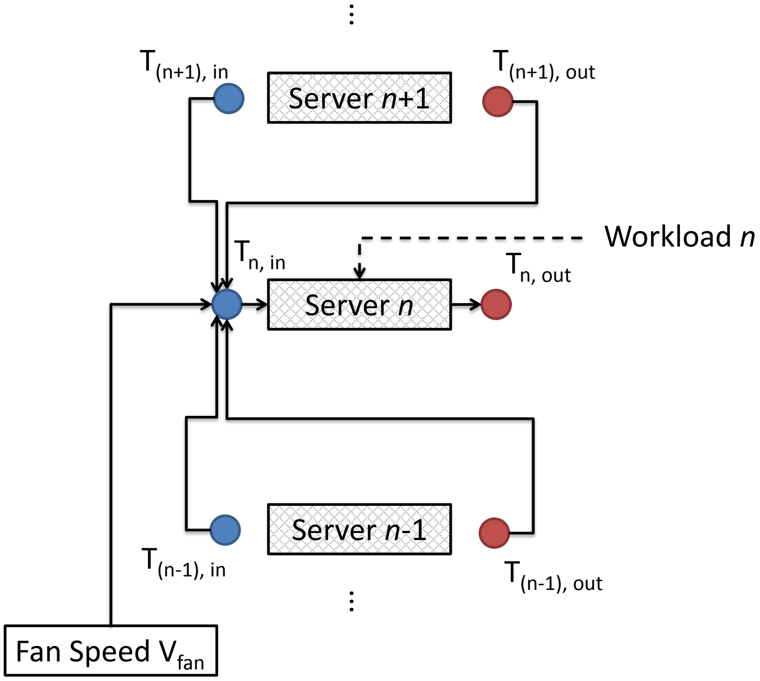


Figure 11.5: The ThermoCast modeling framework. Circles correspond to model variables, while the arrows indicate relationships among these variables.

11.5 Thermal aware prediction model

We build a model based on first principles from thermodynamics and fluid mechanics. While a comprehensive computational fluid dynamics model (CFD) is often complex and computationally expensive, we exploit a zonal model for the thermo/air dynamics near the server rack. The intuition is to divide the data center's indoor environment into a coarse grid of zones such that air and thermal conditions can be considered uniform within each zone. We divide the room into zones, as shown in Figure 11.6, and define the variables shown in Table 11.1.

We make the following assumptions to simplify the model during a prediction cycle:

A0: Incompressible air, which implies the density of air ρ is constant. We ignore dynamic pressure due to height and temperature differences, and care only about the Bernoulli effect caused by high-speed airflow.

A1: T_{RM}, the room temperature is constant in a short period of time.

A2: T_{FL}, the supply air temperature at the floor vent is constant within a short period of time.

A3: Constant server fan speed, thus $U_{F_i} = U_{B_i}$.

A4: The vertical air flow at the back of the server is negligible.

A5: The vertical air flow in the front scales linearly with the floor vent speed, although the scaling factor depends on server height and the on/off status of nearby servers. In other words $V_{F_i} = \delta_i V_{FL}$, where δ_i is constant during a short period of time.

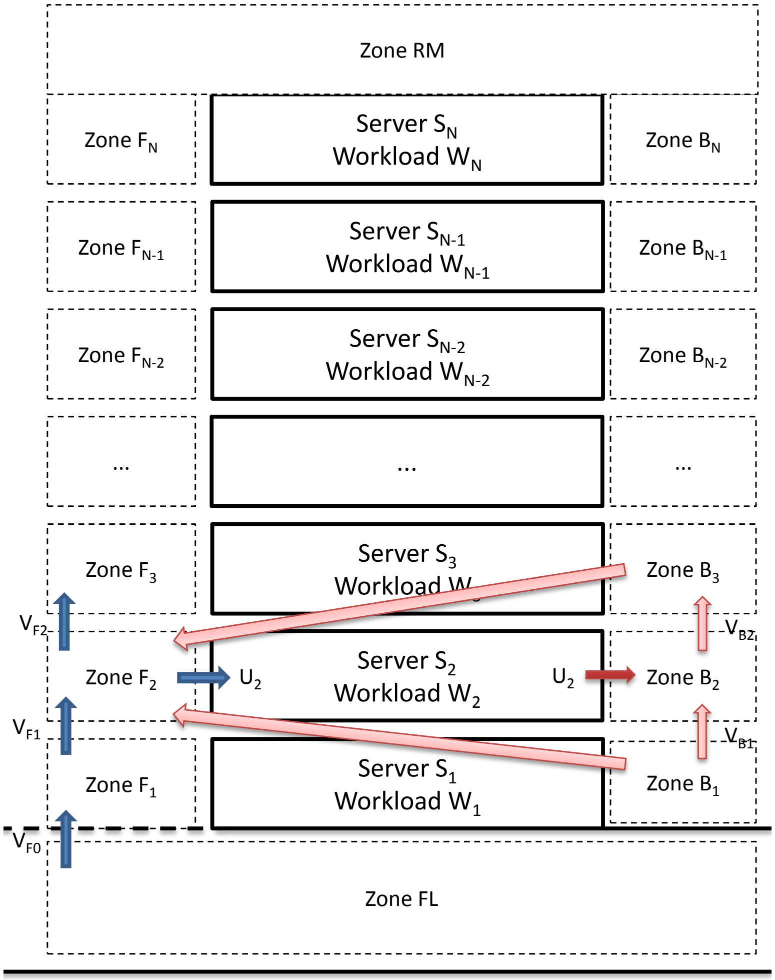


Figure 11.6: The zonal model for thermo dynamics around a rack.

We then model the following relationships between model variables.

Basic fluid dynamics: (Bernoulli's principle):

$$P = -\frac{1}{2}\rho\hat{V}^2 \tag{11.1}$$

where $\hat{V}$ is the total air speed, i.e. $\hat{V_z}^2 = U_z^2 + V_z^2$. Thus, for zone z

$$P_z = -\frac{1}{2}\rho(U_z^2 + V_z^2) \tag{11.2}$$

Now consider server s with front zone F_s and back zone B_s. By (11.2), and assumption [A4], $V_{B_s} = 0$

$$P_{F_s} - P_{B_s} \quad = \quad \frac{\rho}{2}(V_{B_s}^2 - V_{V_s}^2) = -\frac{\rho}{2}V_{F_s}^2 \tag{11.3}$$

This pressure difference drives the hot air to flow from the hot aisle to the cold aisle.

Table 11.1: ThermoCast parameters and their description.

Parameter	Description
$i = 1 \cdots N$	server index in the rack
Zone F_i	area in front of the server i; close enough to get the Bernoulli effect.
Zone S_i	area inside server i
Zone B_i	area immediately behind server i; only impacted by the heat generated by the server
Zone RM	zone for the room ambient air
Zone FL	zone below the vent
T_z	temperature of zone z
V_z	vertical airflow speed out of zone z
U_z	horizontal airflow speed out of zone z
P_z	dynamic air pressure in zone z, i.e. "measurable" pressure minus atmospheric pressure
W_s	Watts generated by server s, representing its workload
ρ	air density

Basic thermodynamics:

Consider zone $z \in \{F_1, ...F_N\}$ of air mass M_z and temperature T_z. During time interval $[t, t + t_s]$, let $\Lambda_{i,z}(t)$ be the air mass flowing into z from zone i with temperature $T_i(t)$, then $\sum_i \Lambda_{i,z}(t)$ is the air flowing out of zone z (due to mass conservation and assumption about incompressible air [A0]) with temperature $T_z(t)$:

$$M_z \cdot T_z(t+1) = M_z \cdot T_z(t) + \sum_i (\Lambda_{i,z}(t) \cdot T_i(t)) - (\sum_i \Lambda_{i,z}(t)) \cdot T_z(t) \tag{11.4}$$

The air mass in exchange per unit time is proportional to air speed. So,

$$\Lambda_{i,z}(t) = \beta_{i,z}\sqrt{P_i(t) - P_z(t)} \tag{11.5}$$

where $\beta_{i,z}$ captures air density (ρ) and all geometric characteristics between i and z, such as gap size, server type, and server on/off etc., i.e. how hard it is to push air from i to z.

So, by (11.4) and (11.5)

$$T_z(t+1) = (1 - \alpha_z) \cdot T_z(t) + \sum_i \beta_{i,z}\sqrt{P_i(t) - P_z(t)} \cdot T_i(t) \tag{11.6}$$

where α_z captures the flowing out of the zone, including those going into the server and those going up/down to the next zone. Clearly, α_z depends on height, fan speed, and server on/off.

Plugging in (11.3) and using assumption [A5], we derive the following structure of the local thermodynamics model:

$$T_z(t+1) = a \cdot T_z(t) + \sum_{j \in \{B_{z-1}, B_z, B_{z+1}\}} (\beta_j \cdot \sqrt{P_z(t) - P_j(t)} \cdot T_j(t)) + \sum_{j \in \{F_{z-1}, F_{z+1}\}} c_j \cdot V_{FL} \cdot T_j(t) \tag{11.7}$$

where parameters $\{a, \beta_j$'s, and c_j's$\}$ are server and location dependent and are learned by each server through parameter estimation.

Including Workload For each server s, the workload W_s converts into heat and effects the temperature at the back of the server. So for zone B_s, we have:

$$T_{B_s}(t+1) = f_1 \cdot T_{B_s}(t) + f_2 \cdot T_{F_s}(t) + f_3 \cdot U_s(t) \cdot W_s(t) + f_4 \cdot T_{B_{s-1}}(t) \tag{11.8}$$

When the server is on, the horizontal mass exchange from front zone F_z to back zone B_z in Eq. (11.5) is dependent on the server's fan speed ($= U_z$). We also assume the interaction between the server's intake and its neighbor's outtake is indirect, thus eliminating the corresponding terms in Eq. (11.7). Therefore, with such reasoning and the result of Eq (11.7), the workload dependent equation for intake temperature becomes,

$$\begin{aligned} T_z(t+1) = & a \cdot T_{F_z}(t) + b_1 \cdot U_z(t) \cdot T_{B_z}(t) + b_2 \cdot (1 - U_z(t)) \cdot T_{B_z}(t) \\ & + b_3 \cdot V_{FL}(t) \cdot T_{F_{z-1}}(t) + b_4 \cdot V_{FL}(t) \cdot T_{F_{z+1}}(t) \end{aligned} \tag{11.9}$$

11.5.1 Parameter Learning

For each server in the rack, there are a total of eleven parameters in the above local model. To make things concrete, we use the notation, $\theta = \{a, b_1, b_2, b_3, b_4, c_1, c_2, f_1, f_2, f_3, f_4\}$. Let $\theta^{(i)}$ be the parameter set for server i, hence $\theta^{(-1)}$, $\theta^{(0)}$ and $\theta^{(1)}$ correspond to the server immediately below, the server itself, and the server directly above. Note that in our framework, the current local server does not know the temperatures and airflow status for neighbors that are two or more slots away on the rack, hence the corresponding parameters $b_3^{(-1)}$, $c_2^{(-1)}$ and $f_4^{(-1)}$ are explicitly made 0.

Base model

To estimate the parameters, we optimize the following objective function:

$$\hat{\theta}^{(i)} \leftarrow \arg\min f(\theta^{(i)}) = \sum_{t=1}^{t_{max}-1} g(\theta^{(i)}, t) \tag{11.10}$$

where

$$\begin{aligned} g(\theta^{(i)}, t) = & \big(T_{F_i}(t+1) - a \cdot T_{F_i}(t) - b_1 \cdot U_i(t) \cdot T_{B_i}(t) - b_2 \cdot (1 - U_i(t)) \cdot T_{B_i}(t) \\ & - b_3 \cdot V_{FL}(t) \cdot T_{F_{i-1}}(t) - b_4 \cdot V_{FL}(t) \cdot T_{F_{i+1}}(t)\big)^2 \\ & + \big(T_{B_i}(t+1) - f_1 \cdot T_{B_i}(t) - f_2 \cdot T_{F_i}(t) - f_3 \cdot U_i(t) \cdot W_i(t) - f_4 \cdot T_{B_{i-1}}(t)\big)^2 \end{aligned} \tag{11.11}$$

Given the available measurements of temperature, server on/off status, workload, and floor vent air velocity, the objective function is convex and there is a global optimal solution. The solution can be obtained by minimizing the least square objective.

Proposed ThermoCast

The base model assign equal weights to the deviation of prediction and observation at all time ticks. However, in reality, temperatures can be more perturbed by temporal nearby events, e.g. shutdown of the server. Intuitively, a good model should forget the events or data in the distant age. In order to adaptively capture changes in dynamics, our proposed ThermoCast assign different weights for different time ticks, according to the temporal locality. We propose to use the following exponentially weighted loss.

$$\hat{\theta}^{(i)} \leftarrow \arg\min f_\lambda(\theta^{(i)}) = \sum_{t=1}^{t_{max}-1} \exp(\lambda t) g(\theta^{(i)}, t) \tag{11.12}$$

where λ is the forgetting factor, which can be tuned either manually or using cross-validation.

Again the solution of this optimization problem is obtained by solving $\frac{\partial f_\lambda(\theta^{(i)})}{\partial \theta^{(i)}} = 0$.

11.5.2 Prediction

In ThermoCast framework, the prediction component works as follows. Based on the learning results, each server predicts its local temperatures for the near future. The predictor uses a a past window of size T_w for training and predicts T_p minutes into the future.

Note that due to the structure of the model from (11.9), the server's intake temperature depends on its own past intake and its neighbors' intake and outtake, as well as the workload on the server. While the outtake temperature depends its intake, workload(fan status) and its neighboring outtakes. On the other hand, the neighbors' future environmental conditions (e.g. servers may shutdown) are unknown during the prediction process. This is a main source of prediction error and the reason that we cannot predict too far into the future.

In order to run the model forward in time, we extrapolate the neighbor's intake and output temperatures. Furthermore, we need the future floor air flow speed and temperatures. To this end, we use a separate autoregressive (second order AR) to predict the future floor vent air flow.

$$V_{FL}(t+1) = \eta_0 \cdot V_{FL}(t) + \eta_1 \cdot V_{FL}(t-1)$$

Where the parameters η_0 and η_1 are estimated using linear least square.

Since AC is the main external stimulus to the system we build a degenerate model for the bottom machine that depends only on the vent airflow. (The vent temperature is assumed to be a constant.) Using the same notation, the model for the bottom machine has the structure:

$$T(t+1) = \sum_{k=\{0,..m-1\}} a_k \cdot T(t-k) + b' \cdot V_{FL}(t) \tag{11.13}$$

We introduce higher orders m in the regression to counterfeit un-modeled factors such as the node's neighbors. In practice, we found $m = 3$ to be adequate. We use the method described in Section 11.5.1 to estimate these parameters as well.

Table 11.2: Execution time (in milliseconds) for different training and prediction time combinations.

		Prediction length (minutes)			
		5	10	15	20
Training length (minutes)	15	5.1	5.0	5.0	5.1
	30	7.2	7.0	14.8	8.2
	45	10.1	10.3	10.7	11.4
	60	13.4	13.5	13.3	16.9
	75	16.0	17.6	17.7	178.0
	90	20.1	19.5	23.7	204.0

With the predicted floor vent air speed and bottom server temperature, it then straightforward to forecast the intake and outtake temperatures using Eq. 11.9 and Eq. 11.8.

11.6 Evaluation

We evaluate ThermoCast using real data traces, controlled experiments, and trace driven simulations. In particular, we are interested in answering the following questions:

- How accurately can a server learn its local thermal dynamics for prediction?
- How much extra computing capacity can ThermoCast achieve compared to other approaches under the same cooling cost?

For environmental data such as temperature distributions and airflow speed, we use the data collected from the university testbed, as described in section 11.3. We use a total of 900 minutes of data traces, during which the AC has both high and low duty cycles. The sampling interval in the trace is 30 seconds. We choose one server at the top of a rack, one in the middle and one at the bottom of the rack to represent different server locations.

11.6.1 Model Quality

We are interested in how much historical training data a server needs to keep in order to obtain a good enough local thermal model. Obviously, less data means faster training speed, less storage, and less communication among servers. We evaluate the model accuracy in terms of its prediction accuracy. In the experiments, we choose a moving window T_w for training and prediction length T_p.

Figure 11.7 shows the prediction results in terms of Mean Square Error (MSE) as a function of training data length (in minutes). We can see that in general, the more data used the training, the more accurate the model is. The shorter prediction length, the better accuracy we can achieve. In fact, if we use 90 minutes of training data and predict 5 minutes into the future, we can obtain very good results. Figure 11.8(b) shows a time domain plot for one of the traces.

Table shows the computational overhead of prediction and learning on each server (Dual core 3.2GHz, 2G RAM, Win XP server). As the data shows, the overhead is small.

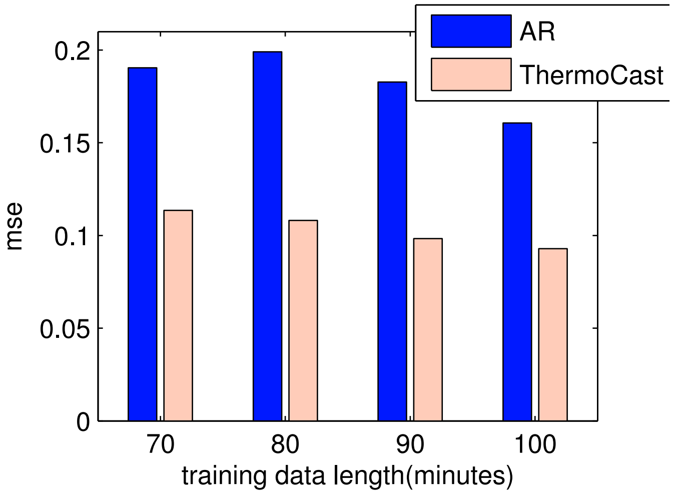
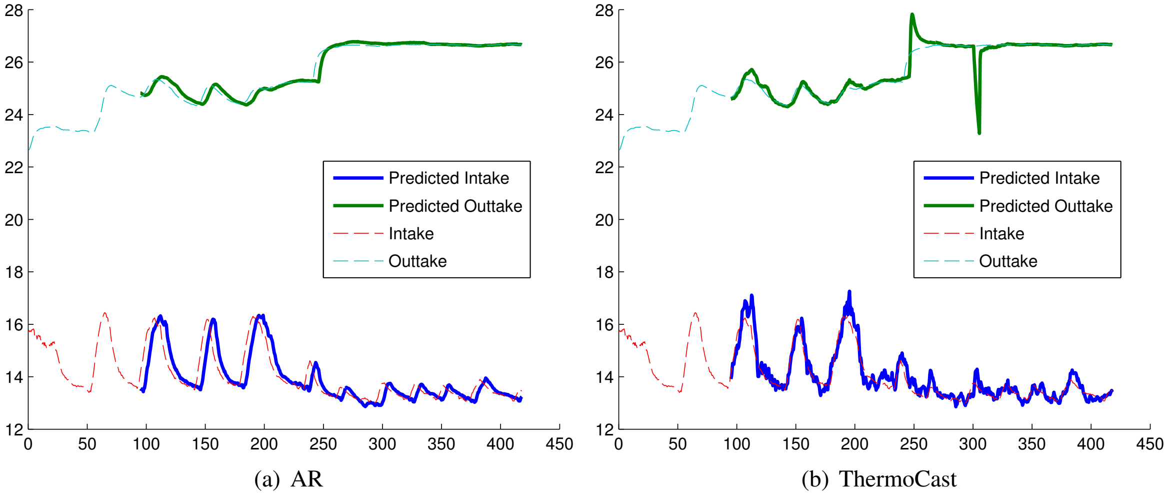


Figure 11.7: Forecasting error (MSE) of the thermal model as a function of training data length. All predictions are made at 5 minutes away from training. ThermoCast produces consistently lower error and is up to 2x better than the baseline AR method.



(a) AR

(b) ThermoCast

Figure 11.8: A time domain trace for prediction quality using ThermoCast. T_w = 90 minutes; all predictions are made at 5 minutes away from training. The baseline AR uses second order auto-regressive model. ThermoCast: $\lambda = 0.006$. ThermoCast intake temperature forecasts closely resemble the actual observations. The spikes seen in the outtake temperature forecasts are due to change in CPU utilization (75% to 100%, and 100% to shutdown). Even though ThermoCast misses a few time ticks in the beginning of these transition, ThermoCast can adapt quickly as new observations become available.

11.6.2 Preventive Monitoring

We did experiments on the real data set to test the capability of our model in case of thermal alarms. The major event of interest in data center is occasional over heating of servers. These event can be caused by a variety of factors such as insufficient cooling, blocking of intake air, fan error, and over placement of workload. Our goal is to exploit ThermoCast to continuously monitor and predict in advance cases of overheating of the intake air. Since we are not allowed to create actual overheating in a real data center, we use real traces of temperature readings and set an artificial threshold (=16). Any temperature higher than such a threshold will trigger an alarm.

The test process works as follows. We first obtain a true labeled trace by identifying "overheating" sections in the temperature sequences. Each section corresponds to a thermal event. In testing, we use ThermoCast or the baseline to forecast the temperatures in the future, and trigger alarms when such predicted temperature is above the thermal threshold. We then calculate two sets of metrics for both our model and the baseline, namely recall(R)/false alarm rate(FAR) and mean look-ahead time(MAT). Recall and false alarm rate are defined on all time ticks with or without alarms, using the following equation:

$$Recall = \frac{\#truealarms}{\#truealarms + \#missedalarms}$$
$$FAR = \frac{\#falsealarms}{\#truealarms + \#falsealarms}$$

Mean look-ahead time (MAT) is to estimate how much time in advance the model can forecast future "overheating" events. It is only measured for the sections when true alarm happens.

$$MAT = \frac{\sum_i^K \max\{\Delta t | f(t_i - \Delta t) > T_{max}\}}{K}$$

where t_i is the starting time of i-th "overheating" section, T_{max} is the temperature threshold. $f(t_i - \Delta t)$ is temperature forecast using all the data before $t_i - \Delta t$ as training and predicting in next few minutes. The longer this time is, the better it predicts since it allows sufficient reaction time.

Table 11.3 show the performance of the alarm prediction based on our proposed ThermoCast and the baseline method. Note our method achieves nearly 10% better recall and forecasts the alarms twice earlier than the baseline approach.

Table 11.3: Thermal alarm prediction performance. Better performance corresponds to higher recall, lower false alarm rate (FAR), and the larger Mean look-ahead time (MAT). $Tmax = 16°$C.

	Baseline	ThermoCast
Recall	62.8%	71.4%
FAR	45%	43.1%
MAT	2.3min	**4.2 min**

11.6.3 Potential Capacity Gains

Better prediction implies better utilization of cooling capacity under the same CRAC load. To evaluate the computing capacity gain, we need to approximate the cooling effect of 1^oC temperature difference in intake temperature.

Our experimental servers are Dell PowerEdge 1950, with 300W peak power consumption. According to its specification, "the airflow through the PE1950III without the added back pressure from the doors is

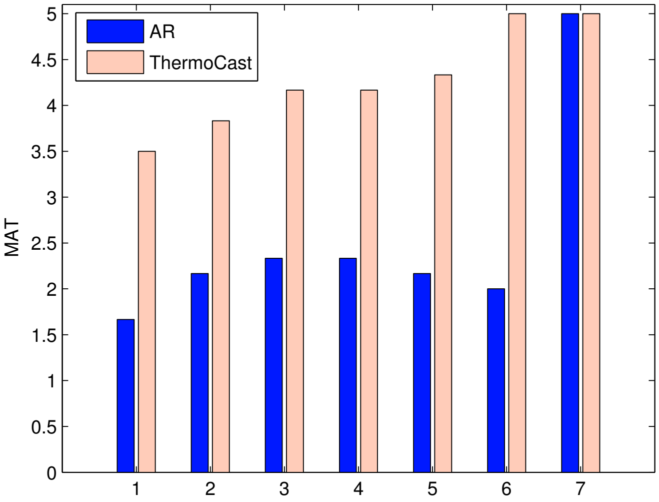


Figure 11.9: Mean look-ahead time(MAT) as a function of the thermal threshold. Higher MAT values provide more time to react. ThermoCast consistently outperforms the baseline AR method.

approximately 35 Cubic Feet Per Minute (CFM)." In [Moss, 2005], Dell Inc. recommended the following rules for estimating cooling capacity.

$$\text{CFM} = 1.78\frac{\text{Power (W)}}{\text{Temperature Differences (°C)}} \tag{11.14}$$

In other words, under 35 CFM, 1^oC air flow can cool 20W of workload.

We compare predictive load placement with static profiling-based workload placement decisions. We use 5-minute forecast length, since it is long enough to change load balancer (or load skewer) policies or migrating virtual machines.

Let's assume that the profiling is tight. That is, we use the maximum measured temperature as the basis for profiling results and compute the difference between the static profiling (16.5^oC at the intake) and prediction results. In both cases, we add a 10% safety margin. With ThermoCast, we can operate the server at 13.75^oC on average, which leads to 53W computing power. That is, on average, the same server now can potentially take up to 53W more workload without adding any additional cooling requirement.

Compare to the 300W peak power consumption, we gain extra 17% compute power with the same cooling. Note that we assume that the 53W is moved from other places in the data center to this server. So, the overall CRAC duty cycle is unchanged. In this way, we can achieve better workload consolidation and shut down more servers from the whole data center perspective.

11.7 Summary

Data center temperature distribution and variations are complicated, which make most workload placement methods shy away from fine-graining thermal aware load scheduling. In this chapter, through dense instrumentation and a gray-box thermo-dynamics model, we show that it is possible to predict servers' thermal condition in real time. The gray-box model is derived from a zonal partition of space near each server. In comparison to CFD models, zonal models are simple to evaluate and robust to unmodeled disturbances through parameter estimation.

To solve the scalability and coordination challenges, ThermoCast uses a federated architecture to delegate model building and parameter estimates to individual servers. Using predictions at each server, workload can be consolidated to servers with access to extra cooling capacity without changing CRAC setting.

This work is a building block towards a holistic data center load management solution that takes into account both the dynamic variation of workload and the responses of cooling system. We also plan to investigate in the future the dynamic server provisioning algorithm based on the local thermal effects due to turning servers on/off.

Chapter 12

Conclusion and Future Directions

Many real world applications generate time series data, i.e. sequences of time stamped numerical or categorical values. Yet there are not a set of readily tools for analyzing the data and exploiting the patterns (e.g. dynamics, correlation, trend and anomalies) in the sequences. Finding patterns in such collections of sequences is crucial for leveraging them to solve real-world, domain specific problems, for motivating examples: (1) motion capture, where analyzing databases of motion sequences helps create animation of natural human actions in movie and game industry ($57 billion business) and design assisting robots; (2) environmental monitoring (e.g. chlorine level measurements in drinking water systems), where the goal is to alert households to safety issues in environments; (3) data center monitoring, with the goal of saving energy consumption for better sustainability and lower cost ($4.5 billion cost in 2006); and (4) computer network traffics, where the goal is to identify intrusion or spams in computer networks.

The thesis focuses on learning and mining large collections of co-evolving sequences, with the goal of developing fast algorithms for finding patterns, summarization, and anomalies. In particular, this thesis builds a series of effort in answering the following research challenges:

1. *Forecasting and imputation*: How to do forecasting and to recover missing values in time series data?
2. *Pattern discovery and summarization*: How to identify the patterns in the time sequences that would facilitate further mining tasks such as compression, segmentation and anomaly detection?
3. *Similarity and feature extraction*: How to extract compact and meaningful features from multiple co-evolving sequences that will enable better clustering and similarity queries of time series?
4. *Scale up*: How to handle large data sets on modern computing hardware?

Throughout the whole thesis, those questions are strongly related to two basic learning tasks for time series: pattern discovery and feature extraction. We have demonstrate in chapters that both mining tasks are closely related. Once we discover patterns (like cross-correlations, auto-correlations) in time series, we can do (a) forecasting (by continuing pattern trends), (b) summarization (by a compact representation of the pattern, like a covariance matrix, or auto-regression coefficients), (c) segmentation (by detecting a change in the observed pattern), and (d) anomaly detection (by identifying data points that deviating too much from what the pattern predicts). Similarly, once we have good features, we can do (a) clustering of similar time sequences, (b) indexing large time series database, and (c) visualizing long time series, plotting them as points in a lower-dimensional feature space.

The thesis answers those questions in three parts as listed in Table 12.1, (i) general models and algorithms; (ii) parallel algorithms and (iii) case studies and domain specific solutions.

Part (i) includes DynaMMo algorithm for mining with missing values, CLDS model and PLiF algorithm for feature extraction. The DynaMMo algorithm enable us obtaining meaningful patterns effectively and efficiently, and subsequently performing various mining tasks including forecasting, compression, and segmentation for co-evolving time series, even with missing values. The PLiF algorithm can find from the multiple correlated time series interpretable features, which enable effective clustering, indexing and similarity search. The CLDS provides a uniform probabilistic graphical model of time series clustering through complex-valued dynamical systems.

Part (ii) describes algorithms for learning and mining for large scale time series data. Linear Dynamical Systems (LDS) and Hidden Markov Models (HMM) are among the most frequently used models for sequential data, however their traditional learning algorithms do not parallelize on multi-processor. To fully utilize the power of multi-core/multiprocessors, we develop a new paradigm (Cut-And-Stitch) for parallelizing their learning algorithms on shared-memory processors (SMP, e.g. multi-core). Both Cut-And-Stitch for LDS (CAS-LDS) and Cut-And-Stitch for HMM (CAS-HMM) scale linearly with respect to the length of sequences, and outperform the competitors often by large factors in term of speedup, without losing any accuracy. In this part, we will also describe our WindMine, a distributed algorithm for finding patterns in large web-click streams. We will discover interesting behavior of user browsing and abnormal pattern as well.

Part (iii) includes special models and algorithms that incorporate domain knowledge. For motion capture, we will describe the natural motion stitching and occlusion filling for human motion. In particular, we provide a metric (L-Score) for evaluating the naturalness of motion stitching, based which we choose the best stitching. Thanks to domain knowledge (body structure and bone lengths), our BoLeRO algorithm is capable of recovering occlusions in mocap sequences, better in accuracy and longer in missing period. For data center, we develop ThermoCast algorithm for forecasting thermal conditions in a warehouse-sized data center. The forecast will help us control and manage the data center in a energy-efficient way, which can save a significant percentage of electric power consumption in data centers.

Summary of Contributions

- We developed algorithms that outperform the best competitors on recovering missing values in time series. They can also achieve the highest compression ratio with lowest error;
- We developed an effective algorithm and a unified model for feature extraction. It can achieve the best clustering accuracy.
- We developed the first parallel algorithm for learning Linear Dynamical Systems. It achieves linear speed up on both super-computers and multi-core desktop machines.

We applied our algorithms in real world applications,

Impact

- Our algorithms have been successfully implemented in motion capture practice, to generate realistic human motions and recover occluded motion sequences;

Table 12.1: Time series mining challenges, and proposed solutions (in italics) covered in the thesis. Repeated for reader's convenience.

	mining	parallel learning
general purpose models	• similarity and feature extraction (*PLiF* Chap.4 and *CLDS* Chap.5) • forecasting and imputation (*DynaMMo* Chap.3) • pattern discovery and summarization (*DynaMMo*, *PLiF*, and *CLDS*)	• parallel LDS on SMPs (*CAS-LDS* Chap.6) • parallel HMM on SMPs (*CAS-HMM* Chap.7)
domain specific	• natural motion stitching (*L-Score* Chap.9) • motion occlusion filling (*BoLeRO* Chap.10) • thermal prediction in data center (*ThermoCast* Chap.11)	• web-click stream monitoring (*WindMine* Chap.8)

- Our algorithms have been applied in data centers at a large company and a university, and help improve the energy efficiency and reduce the power consumption in data centers.
- Our algorithms have been applied to identify patterns and anomalies in web-clicks.

12.1 Future directions

Our long term research goal is to harness large scale multi-dimensional co-evolving time sequences to discover and predict patterns. We would like to expand the research in time series learning and forecasting and apply our research in ubiquitous real world applications. Based on our recent results and research experience, we believe that learning and interacting with co-evolving time series data is a promising direction. in mid-term, we would like to carry out two specific topics along the line of research: never ending learning and gray-box learning for time series.

Gray-box learning One of the ever lasting goal in machine learning is to exploit domain knowledge in the learning models. In our thesis, we have already presented two such cases, BoLeRO and ThermoCast, both of which exploit the domain knowledge to leverage the capability of the models. Next step is to build general gray-box models.

As a specific example, servers and cooling air in data centers interact in complex thermal dynamical manner. Traditional approaches include white-box and black-box methods: the former is using computational fluid dynamics to simulate thermal conditions in the whole data center; while the latter is to train time series models like auto-regression from the sensor observations. Our gray-box approach will incorporate the thermal dynamics in the model but learn appropriate model parameters from the data. Our presented ThermoCast represents one example of such approaches, and it has already obtained impressive results in predicting thermal events. We believe that such approaches can be generalized in many applications, such as modeling the blood pressure in health care domains, modeling taxi traffics and so on.

Never ending learning Time series data usually come from continuously sensor monitoring settings. Our another goal is to build algorithms that can incrementally learn models from data in a *never ending* fashion. One potential approach would be updating the model parameters on the fly. While another more powerful approach is to keep updating the models themselves. For example, the algorithm learns linear regression models initially, and later on trains a logistic model with more observations.

Non-linear and time varying models The basic models proposed in this thesis adopt linear structure on observation and hidden variables. In real applications, it is often more common to observe nonlinear behavior. One interesting future research topic would be extending the models proposed in the thesis such as CLDS to nonlinear case. Promising approaches include:

- Switching models: extending CLDS to switching models to accommodate non-homogeneous behavior (e.g. a walking connecting to a dancing);
- Non-parametric Bayesian models: extending to Gaussian process or hierarchical Dirichlet process to accommodate time varying dynamics.

Bibliography

H. H. Andersen, M. Hojbjerre, D. Sorensen, and P. S. Eriksen. *Linear and graphical models for the multivariate complex normal distribution*. Lecture notes in statistics. Springer-Verlag, 1995. ISBN 9780387945217. 57

O. Arikan and D. A. Forsyth. Interactive motion generation from examples. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-521-1. doi: http://doi.acm.org/10.1145/566570.566606. 129, 131

A. Aristidou, J. Cameron, and J. Lasenby. Predicting missing markers to drive real-time centre of rotation estimation. In *AMDO '08: Proceedings of the 5th international conference on Articulated Motion and Deformable Objects*, pages 238–247, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-70516-1. doi: http://dx.doi.org/10.1007/978-3-540-70517-8_23. 146

L. A. Barroso and U. Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009. ISBN 159829556X, 9781598295566. 3

C. Bash and G. Forman. Cool job allocation: measuring the power savings of placing jobs at cooling-efficient locations in the data center. In *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, pages 29:1–29:6, Berkeley, CA, USA, 2007. USENIX Association. ISBN 999-8888-77-6. 164, 166

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1): 164–171, 1970. ISSN 00034851. 96

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1st ed. 2006. corr. 2nd printing edition, Oct. 2006. ISBN 978-0-387-31073-2. 79, 83

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003. ISSN 1532-4435. doi: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. 106

G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control*. Forecasting and Control Series. Prentice Hall, Englewood Cliffs, NJ, 3rd edition, 1994. ISBN 9780130607744. 10, 33, 166

M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proceedings of the 7th European Conference on Computer Vision*, pages 707–720, London, UK, 2002. Springer-Verlag. ISBN 3-540-43745-2. 17

D. Brandwood. A complex gradient operator and its application in adaptive array theory. *Communications, Radar and Signal Processing, IEE Proceedings F*, 130(1):11 –16, 1983. 61

P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer-Verlag New York, Inc., New York, NY, USA, 1987. ISBN 0-387-96406-1. 10, 166

G. Buehrer, S. Parthasarathy, S. Tatikonda, T. Kurc, and J. Saltz. Toward terabyte pattern mining: an architecture-conscious solution. In *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '07, pages 2–12, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-602-8. doi: http://doi.acm.org/10.1145/1229428.1229432. 11, 81

J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 686–696, New York, NY, USA, 2005. ACM. doi: http://doi.acm.org/10.1145/1186822.1073248. 18, 145, 146

E. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui, E. Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, and Z. Qiu. Psvm: Parallelizing support vector machines on distributed computers. In *Advances in Neural Information Processing Systems*, volume 20. 2007.

G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 337–350, Berkeley, CA, USA, 2008. USENIX Association. ISBN 111-999-5555-22-1. 164, 165, 169

C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *NIPS 19*, pages 281–288. MIT Press, 2006. 11, 80, 81

CMU. Motion capture database, a. URL `http://mocap.cs.cmu.edu`. 2, 155

CMU. Multi-modal activity database, b. URL `http://kitchen.cs.cmu.edu`. 2

R. Collobert, S. Bengio, and Y. Bengio. A Parallel Mixture of SVMs for Very Large Scale Problems. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *NIPS*. MIT Press, 2002. 11, 81

C. B. Colohan, A. Ailamaki, J. G. Steffan, and T. C. Mowry. Tolerating dependences between large speculative threads via sub-threads. In *Proceedings of the 33rd annual international symposium on Computer Architecture*, ISCA '06, pages 216–226, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2608-X. doi: http://dx.doi.org/10.1109/ISCA.2006.43. 80

S. Cong, J. Han, and D. Padua. Parallel mining of closed sequential patterns. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 562–567, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: http://doi.acm.org/10.1145/1081870.1081937. 11, 81

G. Craig. *Introduction to Aerodynamics*, volume 1. Regenerative Press, Anderson, IN, 1st edition, 2003. 169

E. de Aguiar, C. Theobalt, and H.-P. Seidel. Automatic learning of articulated skeletons from 3d marker trajectories. In *ISVC (1)*, pages 485–494, 2006. 156

J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. volume 51, pages 107–113, New York, NY, USA, January 2008. ACM. doi: http://doi.acm.org/10.1145/1327452.1327492. 11, 80, 81

A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 588–599. VLDB Endowment, 2004. ISBN 0-12-088469-0. 8

C. Ding and X. He. K-means clustering via principal component analysis. In *Proceedings of the twenty-*

first international conference on Machine learning, ICML '04, pages 29–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: http://doi.acm.org/10.1145/1015330.1015408. 55, 65

K. Dorfmüller-Ulhaas. Robust optical user motion tracking using a kalman filter. Technical Report 2003-6, Institut fuer Informatik, Universitätsstr. 2, 86159 Augsburg, May 2003. 146

S. T. Dumais. Latent semantic indexing (LSI) and TREC-2. In D. K. Harman, editor, *The Second Text Retrieval Conference (TREC-2)*, pages 105–115, Gaithersburg, MD, Mar. 1994. NIST. Special publication 500-215. 8

E. Elektronik. Ee575 series - hvac miniature air velocity transmitter. Available at `http://www.epluse.com/uploads/tx_EplusEprDownloads/datasheet_EE575_e_02.pdf`. 168

EPA. Epa report to congress on server and data center energy efficiency. Technical report, U.S. Environmental Protection Agency, 2007. 3

C. Faloutsos and K.-I. D. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. CS-TR-3383 UMIACS-TR-94-132 ISR TR 94-80, Dept. of Computer Science, Univ. of Maryland, College Park, 1994. 8, 167

X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-706-3. doi: http://doi.acm.org/10.1145/1250662.1250665. 3

N. Feamster, D. Andersen, H. Balakrishnan, and F. Kaashoek. Bgp monitor - the datapository project, http://www.datapository.net/bgpmon/. 3, 4

T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *J Neurosci*, 5(7):1688–1703, July 1985. ISSN 0270-6474. 126

A. W.-c. Fu, E. Keogh, L. Y. H. Lau, and C. A. Ratanamahatana. Scaling and time warping in time series querying. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 649–660. VLDB Endowment, 2005. ISBN 1-59593-154-6. 8

Y. Fujiwara, Y. Sakurai, and M. Yamamuro. Spiral: efficient and exact model identification for hidden markov models. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 247–255, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: http://doi.acm.org/10.1145/1401890.1401924. 105, 106

K. Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990. ISBN 0-12-269851-7. 39, 46

J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu. Classifying data streams with skewed class distributions and concept drifts. *Internet Computing*, 12(6):37–49, 2008. ISSN 1089-7801. doi: 10.1109/MIC.2008.119. 18, 106, 167

M. Garofalakis, J. Gehrke, and R. Rastogi. *Data Stream Management: Processing High-Speed Data Streams*. Springer, 2009. ISBN 9783540286073. 8

Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, February 1996. 11, 20, 22, 161

A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 79–88, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-804-4. 8, 9, 105

G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. ISBN 0801854148. 42, 47

N. R. Goodman. Statistical analysis based on a certain multivariate complex gaussian distribution (an introduction). *The Annals of Mathematical Statistics*, 34(1):152–177, 1963. 57

H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik. Parallel support vector machines: The cascade svm. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 521–528. MIT Press, Cambridge, MA, 2005. 11, 80, 81

D. Grunwald, C. B. Morrey, III, P. Levis, M. Neufeld, and K. I. Farkas. Policies for dynamic clock scheduling. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation - Volume 4*, OSDI'00, pages 6–6, Berkeley, CA, USA, 2000. USENIX Association. 166

D. Gunopulos and G. Das. Time series similarity measures and time series indexing. In *SIGMOD Conference*, Santa Barbara, CA, 2001. Tutorial. 8, 55

A. C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, Mar. 1990. ISBN 0521321964. 10

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, July 2003. ISBN 0387952845. 37, 49

T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini. Mercury and freon: temperature emulation and management for server systems. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, ASPLOS-XII, pages 106–116, New York, NY, USA, 2006. ACM. ISBN 1-59593-451-0. doi: http://doi.acm.org/10.1145/1168857.1168872. 166

B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: saving energy in data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 17–17, Berkeley, CA, USA, 2010. USENIX Association. 166

L. Herda, P. Fua, R. Plänkers, R. Boulic, and D. Thalmann. Skeleton-based motion capture for robust reconstruction of human motion. In *Proceedings of the Computer Animation*, CA '00, pages 77–86, Washington, DC, USA, 2000. IEEE Computer Society. 17, 145

A. Hjorungnes and D. Gesbert. Complex-valued matrix differentiation: Techniques and key results. *IEEE Transactions on Signal Processing*, 55(6):2740 –2746, 2007. 61

T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1. doi: http://doi.acm.org/10.1145/312624.312649. 106

E. Hoke, J. Sun, J. D. Strunk, G. R. Ganger, and C. Faloutsos. Intemon: continuous mining of sensor data in large-scale self-infrastructures. *SIGOPS Oper. Syst. Rev.*, 40(3):38–44, 2006. ISSN 0163-5980. doi: http://doi.acm.org/10.1145/1151374.1151384. 3

E. Hsu, S. Gentry, and J. Popović. Example-based control of human motion. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 69–77, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. ISBN 3-905673-14-2. doi: http://doi.acm.org/10.1145/1028523.1028534. 17, 145, 146

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Net-*

works, 13(4-5):411–430, 2000. 9, 107

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley-Interscience, 1 edition, 2001. ISBN 047140540X. 8

Intel. Intel research advances 'era of tera': www.intel.com/pressroom/archive/releases/20070204comp.htm, 2007. URL http://www.intel.com/pressroom/archive/releases/20070204comp.htm. 80

M. Jahangiri, D. Sacharidis, and C. Shahabi. Shift-split: I/o efficient maintenance of wavelet-transformed multidimensional data. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 275–286, New York, NY, USA, 2005. ACM. ISBN 1-59593-060-4. doi: http://doi.acm.org/10.1145/1066157.1066189. 8, 9, 167

A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 11–22, New York, NY, USA, 2004. ACM. ISBN 1-58113-859-8. doi: http://doi.acm.org/10.1145/1007568.1007573. 1, 10, 18, 33

C. S. Jensen and S. Pakalnis. Trax: real-world tracking of moving objects. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 1362–1365. VLDB Endowment, 2007. ISBN 978-1-59593-649-3. 8

I. Jolliffe. *Principal Component Analysis*. Springer Verlag, 2nd edition, 2002. ISBN 0-387-95442-2. 8, 105, 107

S. J. Julier and K. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management*, 1997. 131

S. Kagami, M. Mochimaru, Y. Ehara, N. Miyata, K. Nishiwaki, T. Kanade, and H. Inoue. Measurement and comparison of human and humanoid walking. In *Proceedings of 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, volume 2, pages 918 – 922 vol.2, 2003. 55

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME C Journal of Basic Engineering*, (82 (Series D)):35–45, 1960. 11, 47, 161

K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of arima time-series. In *ICDM 2001: Proceeding of 2001 IEEE International Conference on Data Mining*, pages 273–280, 2001. 10

E. Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 406–417. VLDB Endowment, 2002. 8, 105, 106

E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, SIGMOD '01, pages 151–162, New York, NY, USA, 2001. ACM. ISBN 1-58113-332-4. doi: http://doi.acm.org/10.1145/375663.375680. 8, 33, 167

E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle. Indexing large human-motion databases. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 780–791. VLDB Endowment, 2004. ISBN 0-12-088469-0. 1, 8, 18, 33, 105, 106

A. G. Kirk, J. F. O'Brien, and D. A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2005*, pages 782–788, June 2005. 145, 156

T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM '08: Proceeding of Eighth IEEE International Conference on Data Mining*, pages 363–372, 2008. doi: 10.1109/ICDM.2008.89. 106

G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '99, pages 261–272, New York, NY, USA, 1999. ACM. ISBN 1-58113-062-7. doi: http://doi.acm.org/10.1145/303976.304002. 1, 33

F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, SIGMOD '97, pages 289–300, New York, NY, USA, 1997. ACM. ISBN 0-89791-911-4. doi: http://doi.acm.org/10.1145/253260.253332. 107

L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-521-1. doi: http://doi.acm.org/10.1145/566570.566605. 129, 131

F. D. la Torre Frade, J. K. Hodgins, A. W. Bargteil, X. M. Artal, J. C. Macey, A. C. I. Castells, and J. Beltran. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. Technical Report CMU-RI-TR-08-22, Robotics Institute, Pittsburgh, PA, April 2008. 2

N. D. Lawrence and A. J. Moore. Hierarchical gaussian process latent variable models. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 481–488, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: http://doi.acm.org/10.1145/1273496.1273557. 18

J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 39–48, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. doi: http://dx.doi.org/10.1145/311535.311539. 55

J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 491–500, New York, NY, USA, 2002. ACM. ISBN 1-58113-521-1. doi: http://doi.acm.org/10.1145/566570.566607. 127, 129, 131

J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *ICDE 2008: IEEE 24th International Conference on Data Engineering*, pages 140 –149, april 2008. doi: 10.1109/ICDE.2008.4497422. 18, 106, 167

J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, San Jose, California, USA, 2007. ACM. http://doi.acm.org/10.1145/1281192.1281239. 1, 8

L. Li, J. McCann, C. Faloutsos, and N. Pollard. Laziness is a virtue: Motion stitching using effort minimization. In *Short Papers Proceedings of EUROGRAPHICS*, 2008. 18, 79

L. Li, J. McCann, N. Pollard, and C. Faloutsos. Dynammo: Mining and summarization of coevolving sequences with missing values. In *KDD '09: Proceeding of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-193-4. 33, 51, 52, 106, 145, 146, 149, 150, 158, 166

L. Li, B. A. Prakash, and C. Faloutsos. Parsimonious linear fingerprinting for time series. *Proc. VLDB Endow.*, 3:385–396, September 2010. ISSN 2150-8097. 167

Y. Li, T. Wang, and H.-Y. Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 465–472, New York, NY, USA, 2002. ACM. ISBN 1-58113-521-1. doi: http://doi.acm.org/10.1145/566570.566604. 129

C.-J. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao. Racnet: a high-fidelity data center sensing network. In *Sensys*, pages 15–28, 2009. ISBN 978-1-60558-519-2. doi: http://doi.acm.org/10.1145/1644038.1644041. 164

Liebert. Liebert deluxe system/3 - chilled water - system design manual. Available at http://shared.liebert.com/SharedDocuments/Manuals/sl_18110826.pdf, 2007a. 167

Liebert. Liebert deluxe system/3 precision cooling system. Available at http://www.liebert.com/product_pages/ProductDocumentation.aspx?id=13&hz=60, 2007b. 164

Liebert. Technical note: Using ec plug fans to improve energy efficiency of chilled water cooling systems in large data centers. Available at http://shared.liebert.com/SharedDocuments/White%20Papers/PlugFan_Low060608.pdf, 2008. 164

J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, New York, NY, USA, 2003. ACM. doi: http://doi.acm.org/10.1145/882082.882086. 18, 105, 106, 167

C. Liu, F. Guo, and C. Faloutsos. Bbm: bayesian browsing model from petabyte-scale data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 537–546, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: http://doi.acm.org/10.1145/1557019.1557081. 3

G. Liu and L. McMillan. Estimation of missing markers in human motion capture. *Vis. Comput.*, 22(9): 721–728, 2006. ISSN 0178-2789. doi: http://dx.doi.org/10.1007/s00371-006-0080-9. 17, 18, 24, 145, 146

J. Liu, B. Priyantha, F. Zhao, C.-J. M. Liang, Q. Wang, and S. James. Towards discovering data center genome using sensor net. In *Proceedings of the 5th Workshop on Embedded Networked Sensors (HotEmNets)*, 2008. 167

Z. Liu and M. F. Cohen. Keyframe motion optimization by relaxing speed and timing. In D. Terzopoulos and D. Thalmann, editors, *Computer Animation and Simulation '95*, pages 144–153. Springer-Verlag, 1995. 131

J. H. Mathews and R. W. Howell. *Complex Analysis for Mathematics and Engineering*. Jones & Bartlett Pub, 5 edition, January 2006. ISBN 9780763737481. 61

S. Mehta, S. Parthasarathy, and R. Machiraju. On trajectory representation for scientific features. In *ICDM '06. IEEE Sixth International Conference on Data Mining*, pages 997 –1001, dec. 2006. doi: 10.1109/ICDM.2006.120. 18, 105, 106

J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making scheduling "cool": temperature-aware workload placement in data centers. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pages 5–5, Berkeley, CA, USA, 2005. USENIX Association. 164, 166

J. Moore, J. Chase, and P. Ranganathan. Weatherman: Automated, online and predictive thermal mapping and management for data centers. In *ICAC '06. IEEE International Conference on Autonomic Computing*, pages 155 – 164, june 2006. doi: 10.1109/ICAC.2006.1662394. 164, 166

D. Moss. Guidelines for assessing power and cooling requirements in the data center. Available at http://www.dell.com/downloads/global/power/ps3q05-20050115-Moss.pdf, 2005. 180

K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis. Continuous nearest neighbor monitoring in road networks. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 43–54. VLDB Endowment, 2006. 8

T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. K. S. Gupta, and S. Rungta. Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers. *Computer Networks*, 53(17):2888–2904, 2009. 166

D. Newman, C. Chemudugunta, and P. Smyth. Statistical entity-topic models. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 680–686, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. doi: http://doi.acm.org/10.1145/1150402.1150487. 106

Y. Ogras and H. Ferhatosmanoglu. Online summarization of dynamic time series data. *The VLDB Journal*, 15:84–98, January 2006. ISSN 1066-8888. doi: http://dx.doi.org/10.1007/s00778-004-0149-x. 8

C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: a probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '98, pages 159–168, New York, NY, USA, 1998. ACM. ISBN 0-89791-996-3. doi: http://doi.acm.org/10.1145/275487.275505. 8

S. Papadimitriou and P. Yu. Optimal multi-scale patterns in time series streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 647–658, New York, NY, USA, 2006. ACM. ISBN 1-59593-434-0. doi: http://doi.acm.org/10.1145/1142473.1142545. 106

S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, hands-off stream mining. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '2003, pages 560–571. VLDB Endowment, 2003. ISBN 0-12-722442-4. 1, 2, 15

S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. pages 697–708, 2005. 1, 2, 33, 47, 105

S. I. Park and J. K. Hodgins. Capturing and animating skin deformation in human motion. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 881–889, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1179352.1141970. 17, 24, 145

C. Patel, C. Bash, R. Sharma, and R. Friedrich. Smart cooling of data centers. In *ASME Interpack*, 2003a. 166

C. Patel, R. Sharma, C. Bash, and S. Graupner. Energy aware grid: Global workload placement based on energy efficiency. In *ASME International Mechanical Engineering Congress and R&D Expo*, 2003b. 164, 166

D. Patnaik, M. Marwah, R. Sharma, and N. Ramakrishnan. Sustainable operation and management of data center chillers using temporal data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1305–1314, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: http://doi.acm.org/10.1145/1557019.1557159. 3, 164, 166

D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, SIGMOD '97, pages 13–25, New York, NY, USA, 1997. ACM. ISBN 0-89791-911-4. doi: http://doi.acm.org/10.1145/253260.253264. 8

L. Ramos and R. Bianchini. C-oracle: Predictive thermal management for data centers. In *HPCA 2008. IEEE 14th International Symposium on High Performance Computer Architecture*, pages 111–122, feb. 2008. doi: 10.1109/HPCA.2008.4658632. 166

C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*, pages 13–24, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 1-4244-0804-0. doi: 10.1109/HPCA.2007.346181. 11, 80, 81

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, Aug. 1965. ISSN 0001-1452. doi: 10.2514/3.3166. 11

G. Reeves, J. Liu, S. Nath, and F. Zhao. Managing massive time series streams with multi-scale compressed trickles. *Proc. VLDB Endow.*, 2:97–108, August 2009. ISSN 2150-8097. 1, 8, 33

S. Reinhardt and G. Karypis. A multi-level parallel implementation of a program for finding frequent patterns in a large sparse graph. In *IPDPS 2007. IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, 2007. 11, 81

Reuters. Factbox: A look at the $65 billion video games industry, June 2011. URL http://uk.reuters.com/article/2011/06/06/us-videogames-factbox-idUKTRE75552I20110606. 2

R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory predcition and occlusion modeling. Technical Report 1998-007, 2, 1998. 131

C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 147–154, New York, NY, USA, 1996. ACM Press. ISBN 0-89791-746-4. doi: http://doi.acm.org/10.1145/237170.237229. 131

A. Safonova and J. K. Hodgins. Construction and optimal search of interpolated motion graphs. 2007. doi: http://doi.acm.org/10.1145/1275808.1276510. 8

A. Safonova, N. Pollard, and J. K. Hodgins. Optimizing human motion for the control of a humanoid robot. In *AMAM2003*, March 2003. 55

Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: stream mining through group lag correlations. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 599–610, New York, NY, USA, 2005a. ACM. ISBN 1-59593-060-4. doi: http://doi.acm.org/10.1145/1066157.1066226. 106, 119

Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: fast similarity search under the time warping distance. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '05, pages 326–337, New York, NY, USA, 2005b. ACM. ISBN 1-59593-062-0. doi: http://doi.acm.org/10.1145/1065167.1065210. 105, 106

Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *ICDE 2007. IEEE 23rd International Conference on Data Engineering*, pages 1046–1055, Istanbul, Turkey, April 2007. doi: 10.1109/ICDE.2007.368963. 106

Sensirion. Datasheet sht1x (sht10, sht11, sht15) - humidity and temperature sensor. Available at http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf, 2010. 168

J. Shieh and E. Keogh. isax: indexing and mining terabyte sized time series. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 623–631, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: http://doi.acm.org/10.1145/1401890.1401966. 18, 105, 106, 167

H. J. Shin, J. Lee, S. Y. Shin, and M. Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. Graph.*, 20(2):67–94, 2001. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/502122.502123. 132

R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of Time Series Analysis*, 3:253–264, 1982. 11, 146, 161

N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003. 17, 24, 145, 158

J. Sun, S. Papadimitriou, and C. Faloutsos. Distributed pattern discovery in multiple streams. pages 713–718, Singapore, 2006a. 105

J. Sun, S. Papadimitriou, and P. S. Yu. Window-based tensor analysis on high-dimensional and multi-aspect streams. In *ICDM '06. Sixth International Conference on Data Mining*, pages 1076–1080, 2006b. doi: 10.1109/ICDM.2006.169. 106

J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 374–383, New York, NY, USA, 2006c. ACM. ISBN 1-59593-339-5. doi: http://doi.acm.org/10.1145/1150402.1150445. 106

J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. In *In Proceeding SIAM International Conference on Data Mining*, 2007. 1, 3

J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Trans. Knowl. Discov. Data*, 2(3):1–37, 2008. ISSN 1556-4681. doi: http://doi.acm.org/10.1145/1409620.1409621. 106

K. R. Swalin. Evaluating microsoft hyper-v live migration performance using ibm system x3650 m3 and ibm system storage ds3400. Available at ftp://public.dhe.ibm.com/common/ssi/ecm/en/xsw03091usen/XSW03091USEN.PD, 2010. 165

S. Tak and H.-S. Ko. A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24:98–117, January 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1037957.1037963. 131, 132

Q. Tang, S. K. S. Gupta, and G. Varsamopoulos. Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. *IEEE Transactions on Parallel and Distributed Systems*, 19(11):1458–1472, 2008. 164, 166

Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 611–622, New York, NY, USA, 2004. ACM Press. ISBN 1581138598. doi: http://dx.doi.org/10.1145/1007568.1007637. 10, 18, 106, 166, 167

G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In

B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1345–1352. MIT Press, Cambridge, MA, 2007. 145, 146

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999. 39, 69

H. Tong. *Non-linear Time Series: A Dynamical System Approach*. Clarendon Press, Oxford, 1990. ISBN 9780198523000. 10

C. Traina, A. Traina, L. Wu, and C. Faloutsos. Fast feature selection using the fractal dimension,. In *XV Brazilian Symposium on Databases (SBBD)*, Paraiba, Brazil, Oct. 2000. 8

Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 61(2):89–101, June 1989. doi: 10.1007/BF00204593. 126

J. M. VanBriesen. Chlorine levels data. URL `http://www.cs.cmu.edu/afs/cs/project/spirit-1/www/`. 3

M. E. Wall, A. Rechtsteiner, and L. M. Rocha. Singular value decomposition and principal component analysis. In D. P. Berrar, W. Dubitzky, and M. Granzow, editors, *A Practical Approach to Microarray Data Analysis*, pages 91–109, Norwell, MA, Mar 2003. Kluwel. 17, 105

J. Wang and B. Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 232–238, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. ISBN 1-58113-659-5. 8, 125, 131

J. Wang and B. Bodenheimer. Computing the duration of motion transitions: an empirical approach. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 335–344, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. ISBN 3-905673-14-2. doi: http://dx.doi.org/10.1145/1028523.1028568. 125, 131

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008. doi: 10.1109/TPAMI.2007.1167. 145, 146

X. Wei, J. Sun, and X. Wang. Dynamic mixture models for multiple time series. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2909–2914, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. 106

G. Welch and G. Bishop. An introduction to the kalman filter, siggraph 2001 courses, 2001. 133

B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceeding of 14th International Conference on Data Engineering*, pages 201–208, 1998. doi: 10.1109/ICDE.1998.655778. 8

B.-K. Yi, N. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *Proceedings of the 16th International Conference on Data Engineering*, pages 13–22, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0506-6. 17, 105

Z. N. Zhang. The jordan canonical form of a real random matrix. In *Numer. Math. J. Chinese Univ.*, 2001. 42

M. Zhao and R. Figueiredo. Experimental study of virtual machine migration in support of reservation of cluster resources. In *2nd International Workshop on Virtualization Technology in Distributed Computing*, 2007. 165

V. B. Zordan and N. C. Van Der Horst. Mapping optical motion capture data to skeletal motion using a physical model. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 245–250, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. ISBN 1-58113-659-5. 145