Fast Algorithms for Time Series Mining

Lei Li

November 2009
CMU-CS-09-XXX

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.

Keywords: time series, dimensionality reduction, forecasting, summarization

Abstract

Time series data arise in numerous applications, such as motion capture, computer network monitoring, data center monitoring, environmental monitoring and many more. Finding patterns in such collections of sequences is crucial for leveraging them to solve real-world, domain specific problems, for example, to build humanoid robots, to detect pollution in drinking water, and to identify intrusion in computer networks.

The central theme of our work is to answer the question: how to find interesting and unexpected patterns in large time series? In this proposal, we focus on fast algorithms on mining large collections of co-evolving time series, with or with out missing values. We will present three pieces of our current work: natural stitching of human motions, time series mining and summarization with missing values, and a parallel learning algorithm for the underlying model, Linear Dynamical Systems (LDS). Algorithms proposed in these work allow us to obtain meaningful patterns effectively and efficiently, and subsequently to perform various mining tasks including forecasting, compression, and segmentation for co-evolving time series, even with missing values. Furthermore, we apply our algorithms to solve practical problems including recovering occlusions in human motion capture, and generating natural motions by stitching together carefully chosen pairs of candidates. We also proposed a parallel learning algorithm for LDS to fully utilize the power of multicore/multiprocessors, which will serve as a corner stone of many applications and algorithms for time series. All our algorithms scale linearly with respect to the length of sequences, and outperform the competitors often by large factors.

Based on aforementioned work, we propose to attack a number of interesting problems in mining time series data, which can be categorized into two classes: (a) without missing values: including feature extraction, indexing, clustering and data stream monitoring; (b) with missing values: mining under domain constraints, like bone-length constraints in motion capture sequences. Potential applications of these proposed work include occlusion recovery for motion capture, fast retrieval of similar sequences in a large database, and anomaly detection in sensor data and network traffics.

Contents

1 Introduction **1**
- 1.1 Motivation - Scenario . . . 1
- 1.2 Big Picture . . . 4

2 Survey **5**

3 Completed Work **7**
- 3.1 Background - Introduction to Linear Dynamical Systems . . . 7
- 3.2 Natural Motion Stitching . . . 9
  - 3.2.1 Problem Definition . . . 9
  - 3.2.2 Main Idea . . . 10
  - 3.2.3 Results . . . 11
- 3.3 Mining with Missing Values . . . 12
  - 3.3.1 Problem Definition . . . 12
  - 3.3.2 Main Idea . . . 13
  - 3.3.3 Results . . . 16
- 3.4 Parallelizing on Multicore . . . 19
  - 3.4.1 Problem Definition . . . 19
  - 3.4.2 Main Idea . . . 19
  - 3.4.3 Results . . . 20

4 Ongoing and Proposed Work **22**
- 4.1 Mining without Missing Values . . . 22
  - 4.1.1 Feature Extraction for Time Series . . . 22
  - 4.1.2 Stream Monitoring and Control . . . 24
- 4.2 Mining with Missing Values . . . 24
  - 4.2.1 Enhanced General Missing Value Recovery . . . 24
  - 4.2.2 Missing Values under Constraints . . . 25
- 4.3 Timeline . . . 26

5 Conclusion **27**

Chapter 1

Introduction

Given a large collection of co-evolving time sequences, like motion capture sequences, chlorine level measurements in drinking water systems, and temperature monitoring in data centers, we investigate the following questions:

1. How to extract compact and meaningful features from multiple co-evolving sequences that will enable better clustering of time series?
2. How to do forecasting and to recover missing values in time series data?
3. How to identify the patterns in the time sequences that would facilitate further mining tasks such as compression, segmentation and anomaly detection?

Those questions are strongly related to two basic mining tasks for time series: *pattern discovery* and *feature extraction*. The justification is as follows: 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.

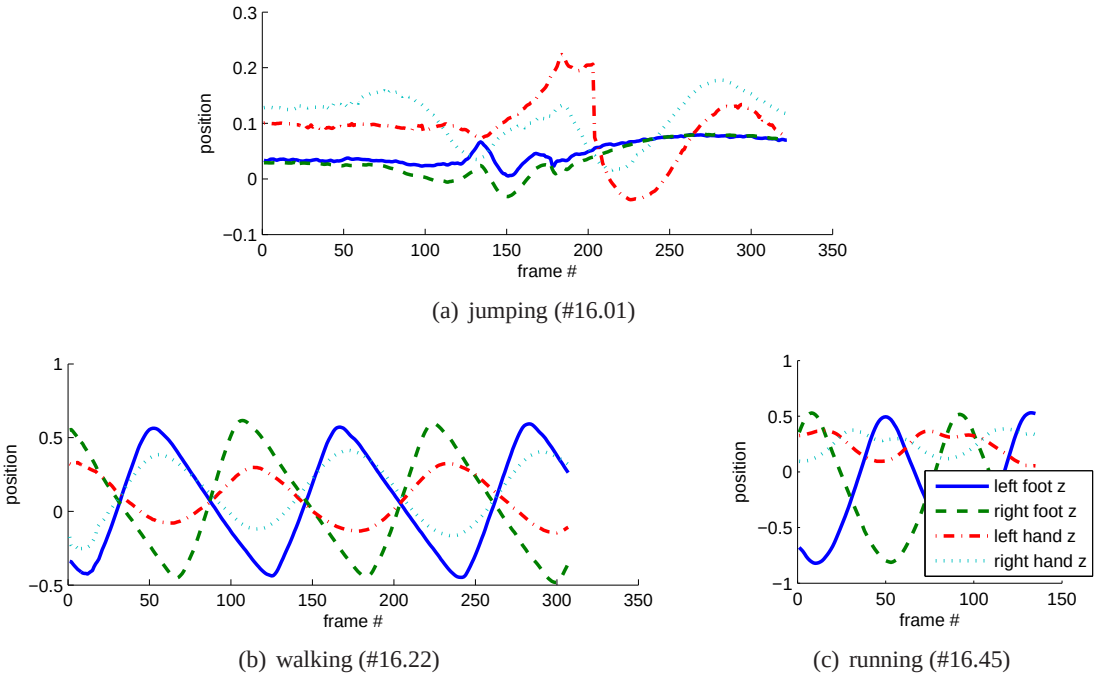
In this report, we will review our approaches to some of these problems, and propose potential attacks to the remaining. We will both look at algorithms that are versatile in diverse applications and mining tasks, and also study domain specific scenarios where domain knowledges should be integrated with general models.

1.1 Motivation - Scenario

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.

Motion capture sequences Motion capture (mocap) is a technique for modelling human motion. CMU researchers have built several large databases of human motions [CMU, b]. 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 video game and interactive entertainment industry is expected to be $57 billion in 2009 [DFC, 2008]. Besides the monetary benefits, research on motion capture databases has increasing applications in improving the quality of life. For example, there is already a motion capture database with various tasks performed in the kitchen [CMU, a], 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: Example 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, b].

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

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

Sensor data Wireless sensors are useful in many situations, such as monitoring chlorine levels in drinking waters systems [Papadimitriou et al., 2005] 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:

- How to summarize the data to reduce the transmission over network? Since in wireless sensors, data transmission consumes much of battery energy.
- How to detect anomalies in sensor data? For example, detecting the a leak or an attack in drinking water by monitoring the chlorine levels.
- 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.

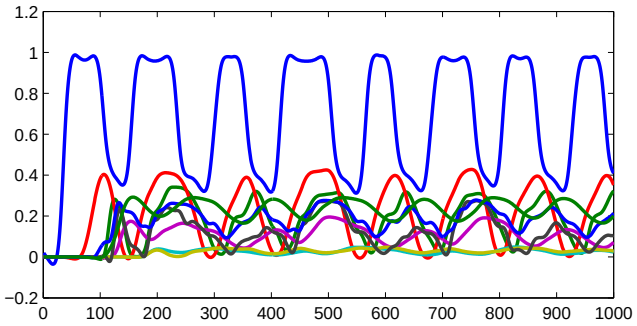


Figure 1.2: Sample snippets of Chlorine concentration versus time, in drinking water for eight households. The data is from [VanBriesen].

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 datacenters 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 datacenters 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.

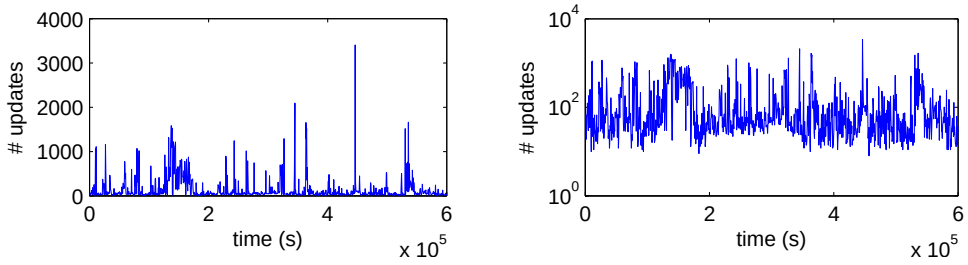


Figure 1.3: Sample snippets from BGP router data at Washington DC: number of updates versus time (in seconds). Notice the original sequence is bursty with no periodicities (shown in part (a)), thus we take the logarithm (shown in part (b)). No obvious patterns, in neither (a) nor (b). Data is from [Feamster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

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 cybersecurity. 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:

- 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.
- How to identify intrusion/anomalies in such computer network traffic data?

1.2 Big Picture

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. Two very promising future directions are: (a) how to learn in a streaming or "never ending" fashion; (b) how to control or intervene given such findings.

In the following chapters, we will review related technical background, summarize our completed work, and describe our proposed work.

Chapter 2

Survey

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

Indexing, signals and streams For indexing, the idea is to extract features [Faloutsos et al., 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].

Dimensionality reduction and matrix methods: There are numerous papers on the topic, with typical methods being PCA [Jolliffe, 1986], SVD/LSI [Dumais, 1994], random projections [Papadimitriou et al., 1998], fractals [Traina et al., 2000]; and a vast literature on feature selection and non-linear dimensionality reduction.

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. 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 database community, Kalman filters have been proposed for sensor data [Jain et al., 2004] as well as for moving objects [Tao et al., 2004].

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[1] 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, 2004] 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://www.openmp.org

Chapter 3

Completed Work

In this chapter, we review some of our work on mining meaningful patterns from multiple coevolving time sequences and using those patterns for solving real problems in motion capture and sensor data monitoring. We present three pieces of work here:

1. Natural motion stitching [Li et al., 2008b];
2. Mining with missing values [Li et al., 2009];
3. Parallelizing on multicore/multiprocessor computers [Li et al., 2008a].

All of these work uses Linear Dynamical Systems (LDS) or its variants as a base model. Here we give a brief introduction to Linear Dynamical Systems (LDS), including its formalization and its learning algorithm. Table 3.1 gives an overview of the symbols used in all tasks and their definitions.

3.1 Background - Introduction to Linear Dynamical Systems

Consider a multi-dimensional sequence $\mathcal{X} = \vec{x}_1, \ldots, \vec{x}_{\mathrm{T}}$ of a length T. For example, $\mathcal{X}$ could be a sequence of marker position vectors captured by video cameras, where each vector $\vec{x}_i$ is of dimensionality m (e.g. $m = 123$ for the motion captured using 41 markers in Figure 3.5, each marker with three coordinates). We could also think of $\mathcal{X}$ as m sequences, each with the duration of T. LDS assumes the evolution of the observation is driven by a hidden Markov process: observations are generated by h hidden variables for that time tick and the hidden variables evolve based on those of previous time ticks. In LDS, both the *transition* among the hidden variables as well as their *projection* to the observations are described as linear Gaussian models (Eq (3.2-3.2)). We denote them as a matrix $\mathbf{F}$ (also denoted as $\mathbf{A}$ in literature) for the *transition* ($h \times h$) with noises $\{\omega_n\}$; and a matrix $\mathbf{G}$ ($m \times h$, also denoted as $\mathbf{C}$ in literature) for the *projection* with the noises $\{\epsilon_n\}$ at each time-tick n. For example, Figure 1.2(b) shows a sample data $\mathcal{X}$ for a walking motion ($m = 4$). Hidden variables in such a case may correspond to position, velocity and acceleration, and $h = 3$ indicates degree of freedom. $\mathbf{F}$ is a 3×3 matrix and can be determined from Newtonian dynamics. $\mathbf{G}$ (4×3) tells how each of observed marker coordinates is generated from the hidden states.

Table 3.1: Symbols and Definitions

Symbol	Definition
$\mathcal{X}$	a multi-dimensional sequence of observations with missing values $(\vec{x}_1, ...\vec{x}_\mathrm{T})$
m	number of sequences
T	duration (length) of sequences
h	number of hidden variables for each time tick
$\mathcal{Z}$	a sequence of latent variables $(\vec{z}_1, \dots \vec{z}_\mathrm{T})$
$\vec{\mu}_0$	initial state for hidden variable, $h \times 1$ vector
$\mathbf{\Gamma}$	initial covariance for hidden variable, $h \times h$
$\mathbf{A}$	transition matrix, $h \times h$
$\mathbf{\Lambda}$	transition noise covariance, $h \times h$
$\mathbf{C}$	projection matrix, $m \times h$
$\mathbf{\Sigma}$	projection noise covariance, $m \times m$
$\mathcal{Q}$	query sequence
$\mathrm{T_Q}$	length of query sequence
$\mathcal{X}_g$	the observed values in the sequence $\mathcal{X}$
$\mathcal{X}_m$	variables for the missing values in the sequence $\mathcal{X}$
$\mathcal{W}$	missing value indication matrix with the same duration and dimension of $\mathcal{X}$

Figure 3.1 provides the graphical representation of following equations defining a LDS:

$$\vec{z}_1 = z_0 + \omega_0 \tag{3.1}$$
$$\vec{z}_{n+1} = \mathbf{F}\vec{z}_n + \omega_n \tag{3.2}$$
$$\vec{x}_n = \mathbf{G}\vec{z}_n + \epsilon_n \tag{3.3}$$

where z_0 is the initial state of the hidden chain, and ω_0, ω_i and $\epsilon_i (i = 1 \dots \mathrm{T})$ are multivariate Gaussian noises:

$$\omega_0 \sim \mathcal{N}(0, \Gamma) \quad \omega_i \sim \mathcal{N}(0, \Lambda) \quad \epsilon_j \sim \mathcal{N}(0, \Sigma)$$

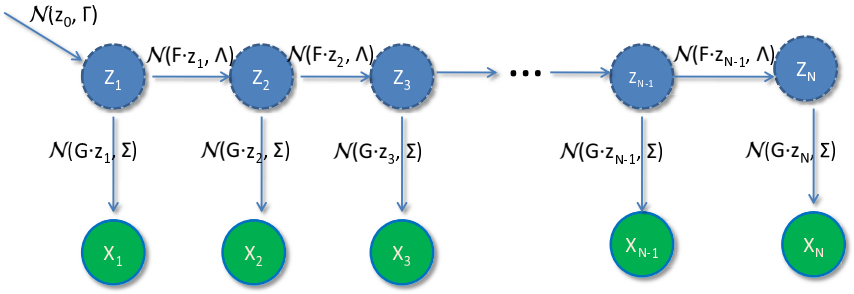


Figure 3.1: A Graphical Representation of the Linear Dynamical System: $\vec{z}_1, \dots, \vec{z}_\mathrm{T}$ indicate hidden variables; $\vec{x}_1, \dots, \vec{x}_\mathrm{T}$ indicate observation. Arrows indicate Linear Gaussian conditional probabilistic distributions.

Given the observation sequence, the goal of the learning algorithm is to compute the optimal parameter set $\theta = (\mu_0, \Gamma, F, \Lambda, G, \Sigma)$. The optimum is obtained by maximizing the log-likelihood $l(\mathcal{X}; \theta)$

over the parameter set θ. A typical learning method for LDS is the expectation-maximization (EM) algorithm [Shumway and Stoffer, 1982, Ghahramani and Hinton, 1996], which iteratively maximizes the expected complete log-likelihood in a coordinate-ascent manner:

$$Q(\theta^{new}, \theta^{old}) = \mathbb{E}_{\theta^{old}}[\log p(\vec{y}_1 \dots \vec{y}_N, \vec{z}_1 \dots \vec{z}_N | \theta^{new})] \tag{3.4}$$

In brief, the algorithm first guesses an initial set of model parameters θ_0. Then, at each iteration, it uses a forward-backward algorithm (known as Kalman filtering [Kalman, 1960] and Kalman smoothing [Rauch, 1963]) to compute expectations of the hidden variables $\hat{\vec{z}}_n = \mathbb{E}[\vec{z}_n \mid \mathcal{X}; \theta_0]$ $(n = 1, \dots, \mathrm{T})$ as well as the second moments and covariance terms, which is the E-step. In the M-step, it maximizes the expected complete log-likelihood of $\mathbb{E}[L(\mathcal{X}, \vec{z}_{1\dots\mathrm{T}})]$ with respect to the model parameters. Since the computation of $\mathbb{E}[\vec{z}_n \mid \mathcal{X}]$ depends on $\mathbb{E}[\vec{z}_{n-1} \mid \mathcal{X}]$ and $\mathbb{E}[\vec{z}_{n+1} \mid \mathcal{X}]$. We refer the reader to an excellent explanation about the EM algorithm for LDS in [Bishop, 2006].

LDS will serve as a common underlying model for the following work, and we use it or its variant in diverse settings: (a) with domain specific and predefined parameters (e.g. $\mathbf{F}$ and $\mathbf{G}$) in natural motion stitching; (b) extending to missing values for mining tasks; and (c) using the EM algorithm as the baseline competitor of our parallel algorithm.

3.2 Natural Motion Stitching

This section is based on the work in [Li et al., 2008b]. Recently researchers have been creating large databases of human motion capture, for example, the CMU mocap database [CMU, b] and the multi-modal activity database [CMU, a]. One particular goal, among others, is to generate new human motions from such databases. This has a lot of applications in practice, for example, generating new animations in the movie industry, and generating new actions in computer games.

3.2.1 Problem Definition

Given two motion-capture sequences that are to be stitched together, how can we assess the goodness of the stitching?

A good distance function is important for the generation of realistic character motion from motion capture databases. We proposed 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 3.2 shows a problem case for the often-used *windowed Euclidean distance*[Wang and Bodenheimer, 2003]: Suppose we want to connect the segments (AB)-to-(CD) or (AB)-to-(EF), both give equally good results with the euclidean distance. However, (AB)-to-(CD) visually looks more appealing than (AB)-to-(EF). Ideally "goodness" metric should be low if humans consider the stitching to be natural. Our proposed L-score captures that. 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.

Problem 1 (Stitching Naturalness)**.**
Given *a query sequence* $\mathcal{Q}$ *of* T *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* $\vec{x}_b$*,*

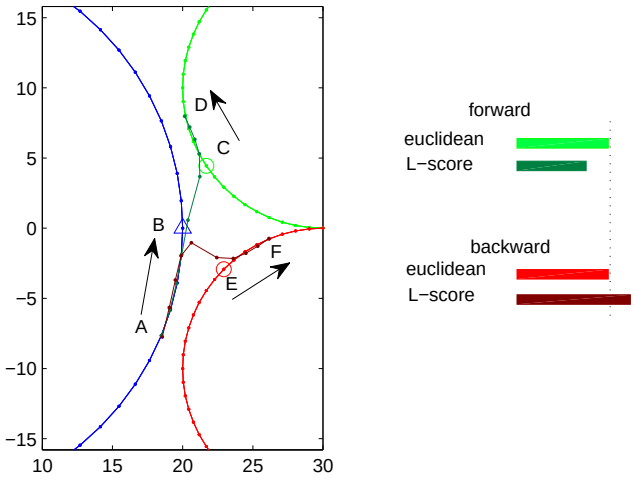


Figure 3.2: Motivating example: The stitching from (AB)-to-(CD) ("forward") seems more natural than the stitching (AB)-to-(EF) ("backward"). The right part shows the corresponding "stitchability" scores. However, the Euclidean distance does not capture the awkwardness of the actual stitching and assigns the same cost (about 47) to both. The "forward" stitching (AB)-to-(CD) has a smoother, more natural-looking trajectory(darker lines).

find *a function to assess the goodness of the resulting stitched sequence* $\mathcal{Y} = \vec{q}_1, \ldots, \vec{q}_a, \vec{x}_b, \ldots, \vec{x}_T$, *so that it agrees with human intuition.*

Figure 3.3 shows an illustration of the problem with the "take-off" and "landing" points.

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].

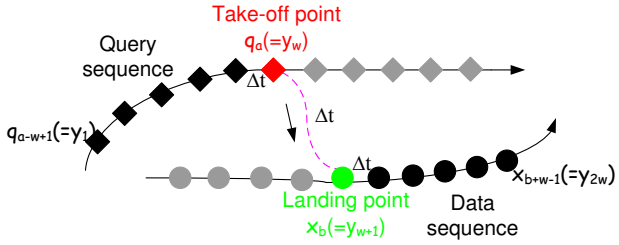


Figure 3.3: Illustration of "take-off" (red square) and "landing" points (green circle). Grayed out points indicate points ignored in our stitching.

3.2.2 Main Idea

How do we capture the "naturalness" of a stitching? Our approach is to go to first principles, informally expressed in our following conjecture:

Conjecture 3.1 (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.

The main contribution of our work is that we proposed an intuitive, first-principles approach, by computing the effort that is needed to do the transition (laziness-effort, or "L-score"). From the above conjecture, the smaller the effort, the more natural the transition will seem to humans. Our L-score relies on a fast, easy to compute estimate of the effort required to make a stitch. We used the Kalman filters (LDS but with fixed parameters) to estimate the motion dynamics (positions, speeds, and accelerations) for the following reasons (a) it has explicit (Newtonian) dynamic equations consistent with first principles (b) it could reduce noise as well.

Mathematical details and a variation: In order to compute L-score, we use Kalman filters to estimate hidden states (real position, velocity, acceleration): $\hat{\vec{z}}_n = (\hat{p}_n, \hat{v}_n, \hat{a}_n)^T (n = 1, \ldots, 2w$, w is window size), we define the following L-score, $L(\mathcal{Q}, a, \mathcal{X}, b, w)$, to approximate the energy spent as the product of force and displacement.

$$L(\mathcal{Q}, a, \mathcal{X}, b, w) = \sum_{n=1}^{2w-1} |(\hat{p}_{n+1} - \hat{p}_n) \cdot \hat{a}_n| \tag{3.5}$$

The details on how to compute and optimize all those objective functions are in [Li et al., 2008b].

3.2.3 Results

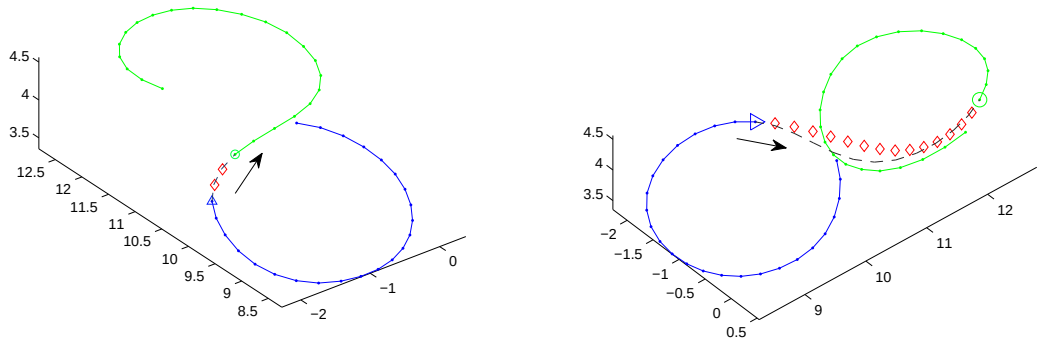
We present experimental results on both artificial (Figure 3.2) and real motions which show that our L-score approach indeed agrees with human intuition, it chooses good stitching points.

We capture a set of waving, walking, running and jumping motions at 30 frames per second. These 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 3.1, and set the parameters according to Newtonian dynamics. We use the window of $2w = 10$, i.e. use five frames right before stitching and five after. To make the stitching more natural, we extend the method to allow elongated stitching with "injected frames" in between "take-off" and "landing" points. 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 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 estimated transition trajectories. The processing time is around two and a half hours on a Pentium class machine. The observations (see Figure 3.4) 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 online video[1]).



(a) (b)

Figure 3.4: 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 out method; notice how close they are to the ground truth (gray dashed line). Our method either finds the correct k_{opt} (=2 in left) or gets very close (=14, vs 11, in right). The data is from [Li et al., 2008b].

3.3 Mining with Missing Values

This section is based on the work in [Li et al., 2009].

Another important problem is mining missing values in time series. Missing values can happen, e.g., due to occlusion in motion capture sequences where some of the markers are temporarily out of sight, or in sensor streams, due to radio frequency interference or due to low battery. For example, periods of occlusion for some markers are common due to the complex nature of human body and relative position of markers (Figure 3.5).

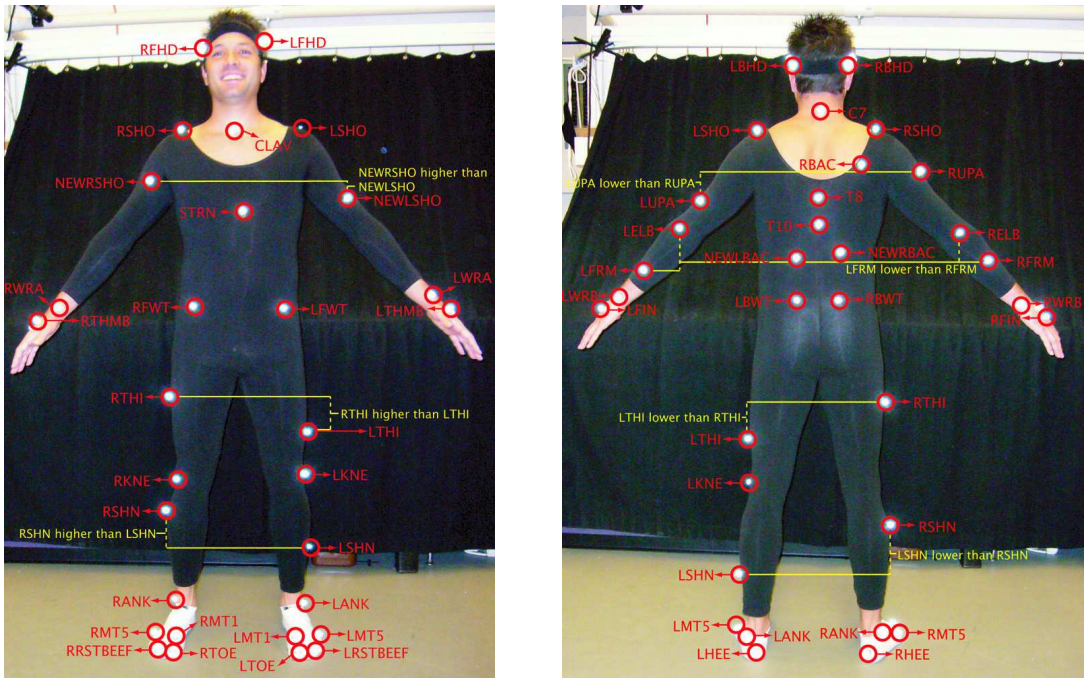
Our second piece of work is exactly on recovering and mining with missing values for general time series.

3.3.1 Problem Definition

Given multiple time sequences with missing values, we propose "DynaMMo" which summarizes, compresses, and finds latent variables. The idea is to discover hidden variables and learn their evolving pattern, making our algorithm able to function even when there are missing values.

The problem is formally defined as follows (see Table 3.1 for a summary of symbols):
Problem 2 (Missing Values)**.**
Given *a time sequence* $\mathcal{X}$ *with the duration* T *in* m *dimensions,* $\mathcal{X} = \{\vec{x}_1, \ldots, \vec{x}_\mathrm{T}\}$*, with the observed*

[1] http://www.cs.cmu.edu/~leili/mocap.stitch



(a) (b)

Figure 3.5: Markers on human actors in CMU motion capture database [CMU, b].

part as $\mathcal{X}_g$, and the missing part as $\mathcal{X}_m$, and missing indication matrix $\mathcal{W}$, $\vec{w}_{t,k} = 0$ whenever $\mathcal{X}$'s k-th dimensional observation is missing at time t, and otherwise $\vec{w}_{t,k} = 1$, the goal is to solve the following issues:

- **recover** *the missing values;*
- **forecast** *the future trends.*

So that both the recovery and forecasting match underlying pattern in the data.

As discussed in the introduction (Chap 1), the ability to forecast is vital in time series mining tasks. Once we have a good model to forecast, we can obtain extra benefits such as summarization by storing less with the forecast and segmentation by detecting changing points that deviate from the forecast.

3.3.2 Main Idea

Our main idea is to simultaneously exploit smoothness and correlation, as shown in Figure 3.7. Smoothness is what splines and linear interpolation exploit: for a single time-sequence (say, the left-hand 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-hand and the right-hand are correlated, lagging each other by half a period. Thus, when we are missing x_n, say, the left hand at time-tick n, we can reconstruct it by examining the corresponding values of the right hand (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.

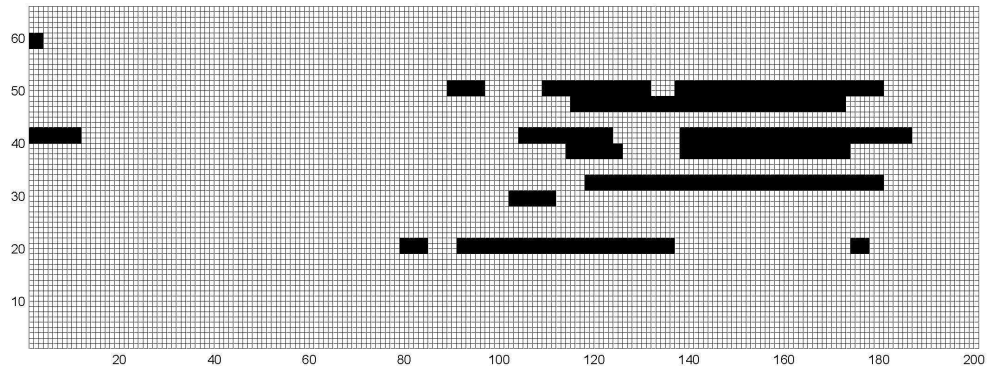


Figure 3.6: Occlusion in handshake motion. 66 joint angles (rows), for ≈ 200 frames. Dark color indicates a missing value due to occlusion. Notice that occlusions are clustered.

The main contribution of our approach is that it shows how to exploit both sources of redundancy (smoothness and correlation) in a principled way. We set up the problem as a Dynamic Bayesian Network (specifically, LDS) 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.

Mathematical details We build a probabilistic model (Figure 3.8) 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. DynaMMo method recovers the missing values by alternately iteratively estimating:

1. the governing dynamics $\mathbf{F}$ and $\mathbf{G}$, as well as other parameters z_0, Γ, Λ and Σ;
2. the latent variables $\hat{\vec{z}}_n = \mathbb{E}[\vec{z}_n]$, $(n = 1 \dots \mathrm{T})$;
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 $\theta = \{\mathbf{F}, \mathbf{G}, z_0, \Gamma, \Lambda, \Sigma\}$:

$$\begin{aligned} Q(\theta) &= \mathbb{E}_{\mathcal{X}_m,\mathcal{Z}|\mathcal{X}_g,\mathcal{W}}[P(\mathcal{X}_g, \mathcal{Z}_m, \mathcal{Z})] \\ &= \mathbb{E}_{\mathcal{X}_m,\mathcal{Z}|\mathcal{X}_g,\mathcal{W}}[-D(\vec{z}_1, z_0, \Gamma) - \sum_{t=2}^{\mathrm{T}} D(\vec{z}_t, \mathbf{F}\vec{z}_{t-1}, \Gamma) \\ &\quad - \sum_{t=1}^{\mathrm{T}} D(\vec{x}_t, \mathbf{G}\vec{z}_t, \Sigma) - \frac{\log|\Gamma|}{2} - \frac{(\mathrm{T}-1)\log|\Lambda|}{2} - \frac{\mathrm{T}\log|\Sigma|}{2}] \end{aligned} \tag{3.6}$$

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

One benefit of DynaMMo is that it helps compress time series more compactly and accurately. The basic compression idea is to store the learned model parameters ($\mathbf{F}, \mathbf{G}$ and etc.) and values of hidden variables

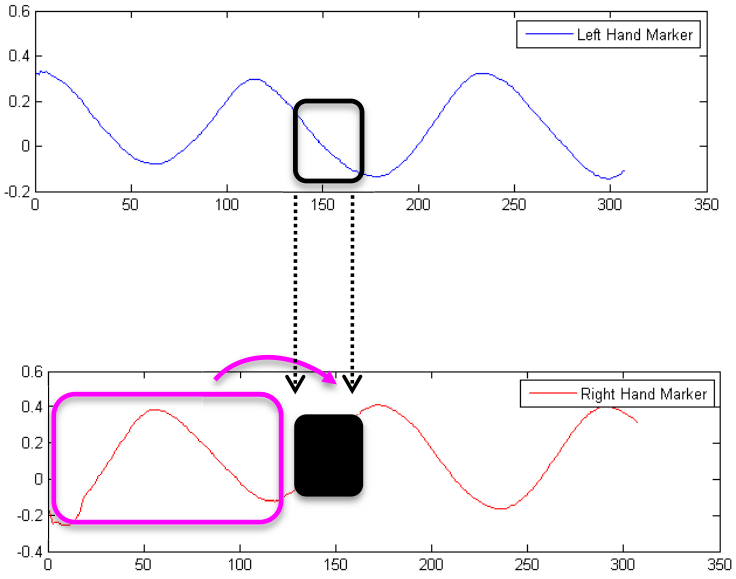


Figure 3.7: Illustration of DynaMMo intuition: use both correlation between sequences (black dashed double arrow) and temporal smoothness (pink arrow) to recover missing values (black box). The data is from #16.22 in [CMU, b].

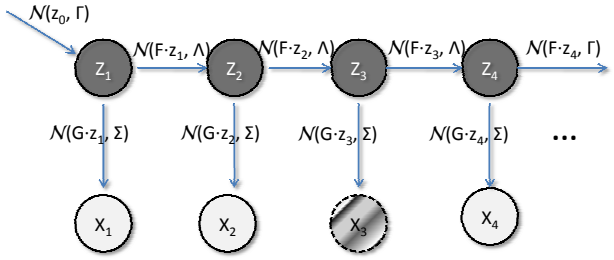


Figure 3.8: Graphical Illustration of the Model. $\vec{z}_{1\ldots4}$: latent variables; $\vec{x}_{1,2,4}$: observations; $\vec{x}_3$: partial observations. Arrows denote Gaussian distributions.

($\vec{z}$'s) for a subset of time ticks. Based on different strategies to choose the subsets, hence different compression ratio and accuracy, we proposed three variants of DynaMMo compression: (a) fixed compression (DynaMMo$_f$), with the time ticks at fixed interval stored; (b) adaptive compression (DynaMMo$_a$), with only the time ticks exceeding error threshold stored; and (c) optimal compression (DynaMMo$_d$), with the best subset of time ticks computed by dynamic programming.

As a further merit, 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 are expected to have different model parameters and latent variables. We use the reconstruction error as an instrument of segmentation, with spikes in error marking boundaries.

We refer the reader to [Li et al., 2009] for more technical details on DynaMMo, its compression, decompression and segmentation algorithms.

3.3.3 Results

We presented experiments on motion capture sequences and chlorine measurements and demonstrated that our proposed DynaMMo method and its extensions (a) can successfully recover missing values, (b) can provide high compression for little loss of reconstruction accuracy, and (c) can identify meaningful segments, (d) scalable on duration of time series.

Recovering missing values Figure 3.9 shows the reconstructed signal for an occluded jumping motion. DynaMMo gives the best result close to the original value. Figures 3.11(a)-3.11(c) show the scatter plots of the average reconstruction error over 58 motions in the Motion dataset, with 10% missing values and 50 average occlusion length. Notice that the reconstruction grows little with increasing occlusion length, compared with other alternative methods (Figure 3.11). There is a similar result found in experiments on additional datasets presented in [Li et al., 2009]. Again, DynaMMo achieves the best performance among the four methods.

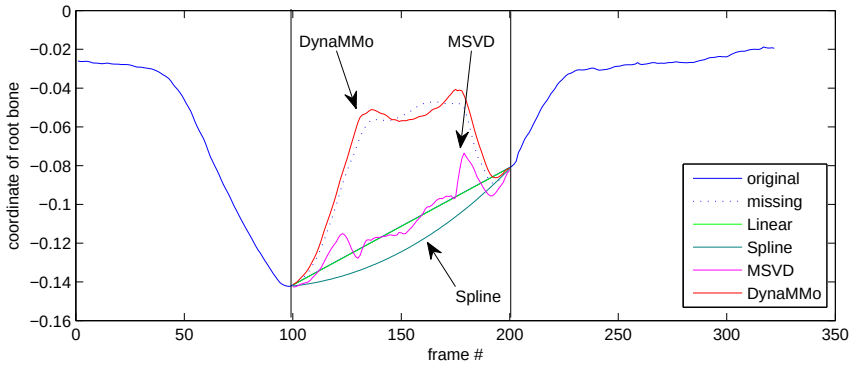
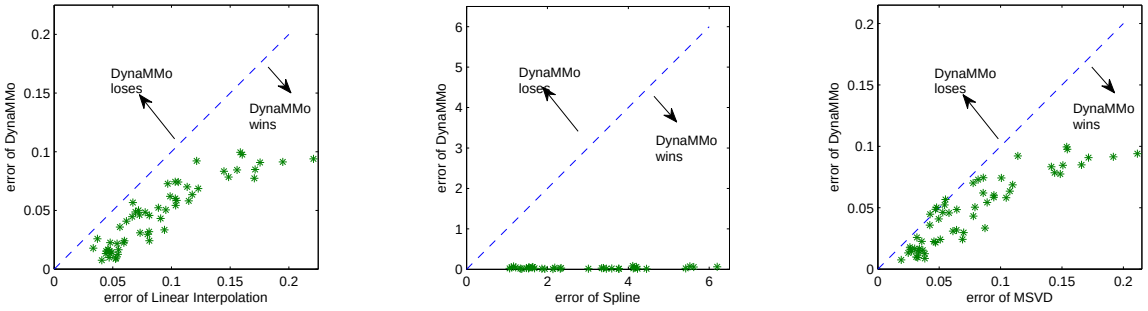


Figure 3.9: Reconstruction for a jump motion with 322 frames in 93 dimensions of bone coordinates. Blue line: the original signal for *root bone* z-coordinate - the dash portion indicates occlusion from frame 100 to 200. The proposed DynaMMo, in red, gets very close to the original, outperforming all competitors. The data is from #16.01 in [CMU, b].



(a) DynaMMo v.s. Linear (b) DynaMMo v.s. Spline (c) DynaMMo v.s. MSVD

Figure 3.10: Scatter plot of missing value reconstruction error for 58 motion sequences(#16 in [CMU, b]). Our DynaMMo bests all competitors: (a) linear interpolation, (b) spline, and (c) missing value SVD (MSVD).

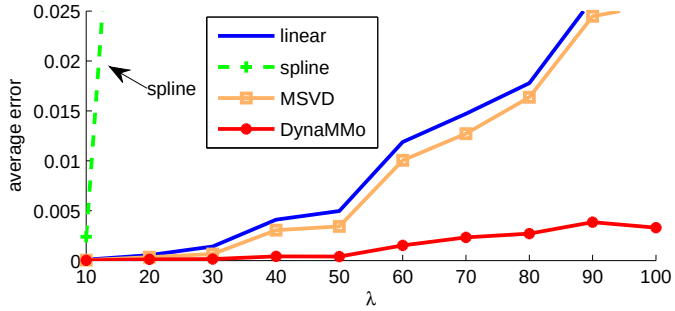


Figure 3.11: Average error for missing value recovery on a sample mocap data (#16.22 in [CMU, b]). Average rmse over 10 runs, versus average missing length λ(from 10 to 100). Randomly 10.44% of the values are treated as "missing". DynaMMo (in red solid line) wins. Splines are off the scale.

Compression Here we present the main results for compression. Figure 3.12 shows the decompression error (in terms of RMSE) versus compression ratio compared with the baseline compression using a combined method SVD and linear interpolation. DynaMMo$_d$ wins especially, in high compression ratios.

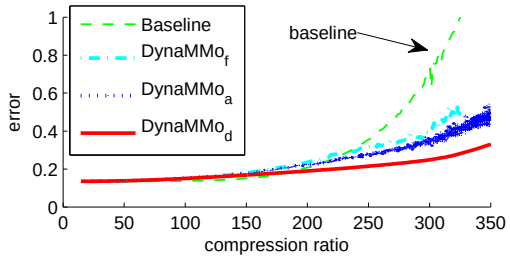


Figure 3.12: Compression for Chlorine dataset: Reconstruction error versus compression ratio. Lower is better. DynaMMo$_d$ (in red solid) is the best.

Segmentation Figure 3.13 shows the segmentation result on a sequence composed of two pieces of sinusoid signals with different frequencies. Our segmentation method could correctly identify the time of frequency change by tracking the spikes in reconstruction error. Figure 3.14 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 m =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.

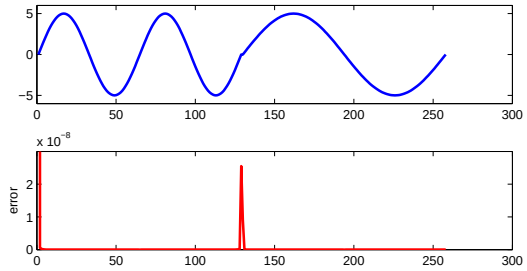


Figure 3.13: Segmentation result on one dimensional synthetic data. Top is a sequence composed of two pieces of sinusoid signals with different frequencies 64 and 128 respectively. Bottom is the reconstruction error per time tick. Note the spike in the middle correctly identify the shifting of frequencies.

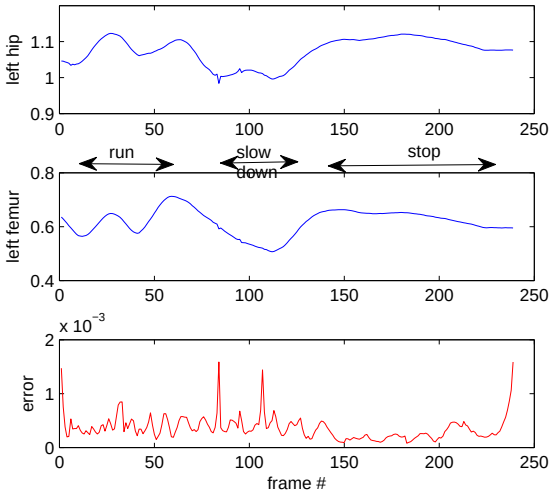


Figure 3.14: Reconstruction error plot for segmentation on a real motion capture sequence in m =93 dimensions with T =250 frames, an actor running to a complete stop, with left hip and femur y-coordinates shown in top plots. The spikes in bottom plot coincide with the slowdown of the pace and transition to stop. The data is from #16.8 in [CMU, b].

3.4 Parallelizing on Multicore

This section is based on the work in [Li et al., 2008a]. Symbols used in this section are summarized in Table 3.1.

3.4.1 Problem Definition

In both problems described above, there are estimation or learning steps for Kalman filters or Linear Dynamical Systems involved in the corresponding algorithms. The well known expectation-maximization (EM) algorithm for learning of LDS 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) [Bishop, 2006]. Although EM algorithm generally produces good results, the EM iterations may take long to converge. 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. Given multiple co-evolving sequences, our goal is to develop a parallel algorithm to learn LDS parameters, by taking advantage of the quickly developing parallel processing technologies to achieve dramatic speedup (Problem 3).

Problem 3 (Parallelizating)**.**
Given *a multi-dimensional sequence $\mathcal{X}$ and k shared memory processors,*
design *a parallel learning algorithm for Linear Dynamical Systems, such that it achieve maximum scale up on multi-processors.*

3.4.2 Main Idea

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] give little benefit to the straightforward EM algorithm due to the nontrivial data dependencies in the graphical model of LDS (Figure 3.1).

The basic idea of our Cut-And-Stitch (CAS) 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. On the other hand, it yields *almost linear* speedup. Figure 3.15 and 3.16 illustrate the main idea and the timeline of the whole algorithm on multiple CPUs.

In practice, the Cut-And-Stitch algorithm includes an extra initialization to improve the first guess of parameter: it runs a sequential forward-backward pass on the whole observation, estimate parameters, i.e. it executes 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.

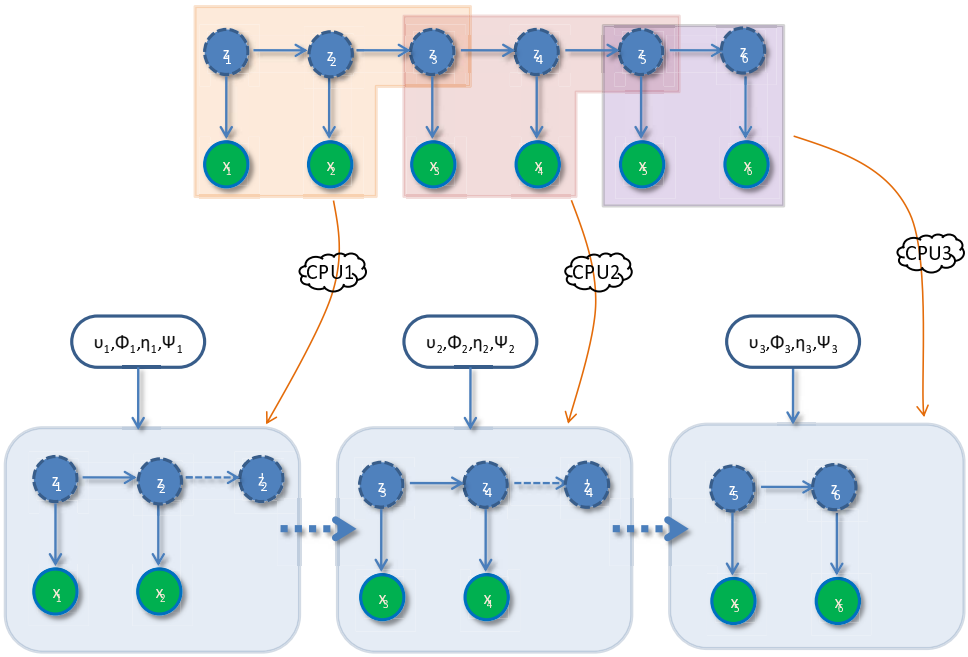


Figure 3.15: Cut-and-stitch main idea: Cut step starts computation without feedback from previous node and later Stitch step reconciles.

Implementation details We implement the CAS algorithm for LDS using OpenMP [2], a flexible and fast multiprogramming interface that supports shared memory on many architectures, including both commercial desktops and supercomputer clusters. In our implementation, we specifically optimize the performance by a set of techniques, including (a) fully exploiting cache by carefully choosing of shared variables, (b) static predefined scheduling of multi-threads in the "`for` loops"; and (c) proper adopting locks and barriers to prevent contention and to minimize synchronization overhead.

The detail equations and implementation issues are presented in [Li et al., 2008a].

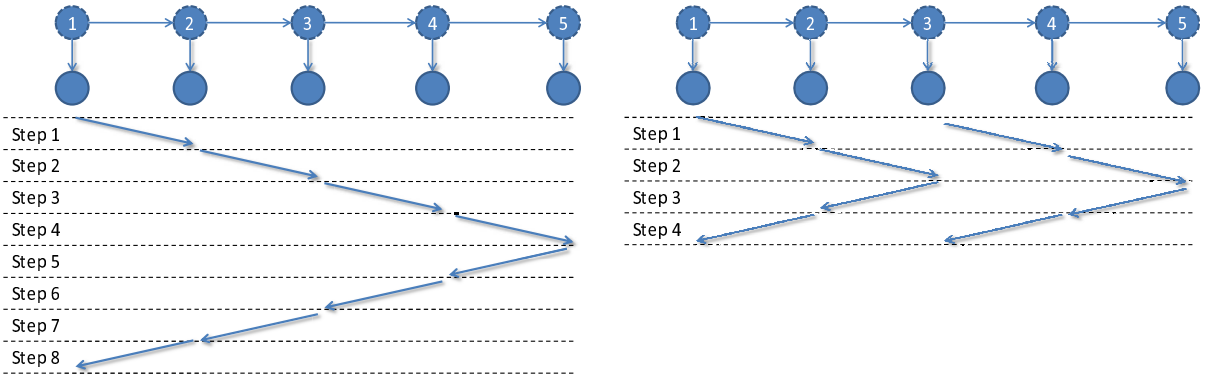
3.4.3 Results

We evaluate our implementation on a standard motion capture dataset from CMU mocap database (#16 in [CMU, b]), comparing with base sequential EM algorithm in both quality and speedup. The conclusion is our proposed Cut-And-Stitch achieves almost linear speed up on both a multi-core desktop and a multi-processor supercomputer.

Table 3.2: Normalized Reconstruction Error

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

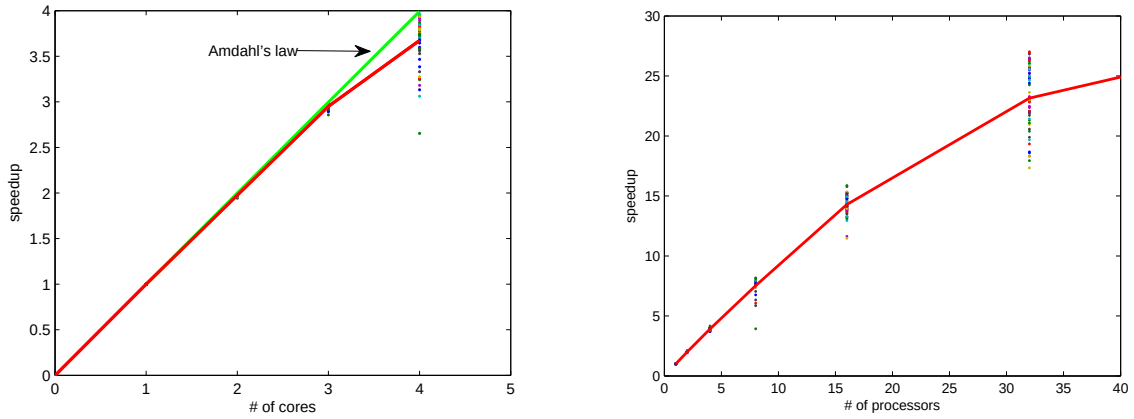
[2] http://www.openmp.org



(a) EM: (sequential) E-step (b) CAS: (parallel) Cut step

Figure 3.16: Graphical illustration of EM and Cut-And-Stitch (CAS) algorithm on multiple CPUs. Arrows indicates the computation on each CPU.

Figure 3.17 shows the speedup (ratio of the wall clock time of sequential EM over CAS) on the multi-core desktop (maximum 4 cores) and a super computer at NCSA[3]. We also include the theoretical limit from Amdahl's law. Table 3.2 shows the reconstruction error: both parallel and serial achieve very small error and are similar to each other. Notice the reconstruction by CAS is very close to that by sequential EM algorithm.



(a) desktop (b) supercomputer

Figure 3.17: Average speedup of CAS on a commercial multi-core desktop and a supercomputer, running on 58 motions (#16 in [CMU, b]). The Sequential version is on one processor, identical to the EM algorithm. The speedup is calculated as the ratio of the wall clock of EM over CAS.

[3]http://www.ncsa.illinois.edu/

Chapter 4

Ongoing and Proposed Work

In this chapter, we will describe some interesting problems in mining time series data, and propose a plan to attack those problems. We will also relate those problems with our complete work, based on which we will propose potential solutions.

Here we group the problems into two categories:

- Mining time series without missing values:

  P1 how to extract compact and meaningful features from time series data, for tasks such as time series clustering, indexing, forecasting and summarization?

  P2 how to monitor data streams such as sensor measurements, web click logs, and network traffic, and to identify patterns and anomalies?

- Mining time series with missing values:

  P3 how to design better optimization techniques to recover missing values more effectively for general time series?

  P4 how to use domain knowledge or constraints to better recover missing values for specific time series, like bone length constraints in motion capture data?

4.1 Mining without Missing Values

4.1.1 Feature Extraction for Time Series

Extracting the essence of time sequences is already very useful - it would be even more useful if those features are compact and easy to interpret, and even better if they could help us do forecasting. Linear Dynamical Systems can forecast future values, however, it is not clear what could serve as features for each sequence, nor to interpret. On the other hand, dimensionality reduction methods such as Principal Component Analysis (PCA) or Independent Component Analysis (ICA) can extract good features for time irrelevant or independent data, however they can not 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 *all* of the above goals, any of which alone is already very useful?

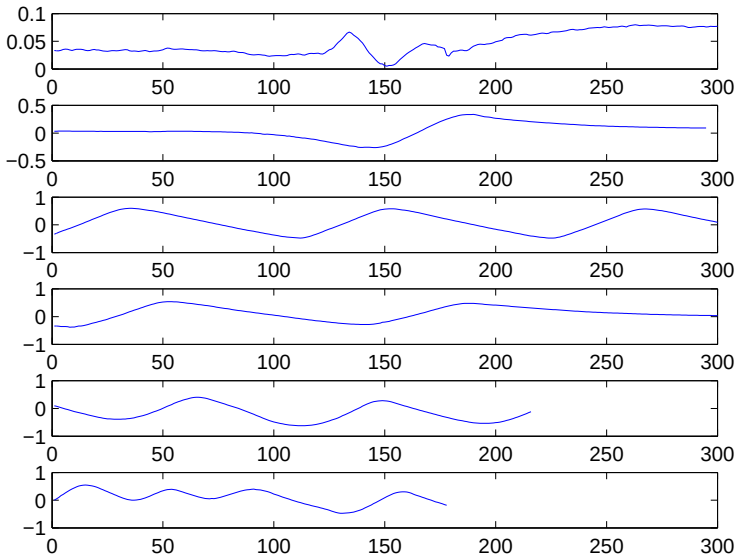


Figure 4.1: Left foot positions jumping (#16.01 and #16.05), walking (#16.21 and #16.34) and running (#16.44 and #16.51) motions.

We would like features for the following motivation scenarios:

- Automatic motion labeling: *Given* a database of motion sequences, each with bone marker coordinates or joint angles, *find* groups of similar motions. Ideally, walking motions should be grouped together even if they start at different footstep or phase. Figure 4.1 includes sample snippets of motion sequences. Good features leads to a good distance function.
- Indexing large motion database: continuing the above scenario, once we have good compact features for each motion sequence, we could design good distance function and build index over the whole database. Eventually we would be able to answer the following query: Give me the most similar motions to the query motion, say, a person's walking sequence. Our goal is to design fast algorithms to answer such query quickly over large motion databases.
- Correlation discovery in network traffic: *Given* network traffic sequences (e.g. BGP updates on routers), we want to find the correlation between traffics, so that we could group similar hosts, routers, and subnetwork together.
- Compressing of time series: *Given* multiple co-evolving sequences, how can we summarize of those time series? The DynaMMo method already achieves high compression ratio, can we do better?

The desired method should achieve the following goals:

1. correlation discovery: the method should be able to find correlations and lag correlations;
2. forecasting: the method should be able to do good forecasting;

3. interpretability: the method should be able to produce meaningful and easy to interpret features.

4.1.2 Stream Monitoring and Control

The mining problem above is mainly for an offline setting, where the data is already present and all the mining is done in batch mode. On the contrary, we have sensor monitoring or click streams coming in an online fashion, and we still want to monitor the time series and find patterns in this scenario. Particularly, we are looking at the following motivation scenarios:

1. Monitoring web click streams and identify patterns and anomalies.
2. Monitoring sensor data to detect anomalies.
3. Monitoring the streams in datacenters, such as CPU utilization, disk utilization, NIC packets, temperature, humidity, and power consumption. A very promising direction is optimal control over data center to save energy consumption.

4.2 Mining with Missing Values

4.2.1 Enhanced General Missing Value Recovery

The DynaMMo algorithm described in Section 3.3 already gives good recovery and mining results. While in our ongoing work, we discover opportunities to enhance the recovery of missing values, by investigating best optimization path/order over parameters and unknowns. We propose a new algorithm to better recover missing values by carefully choosing the optimization order of the objective function. Our proposal is along the same line.

Mathematical details In this work, we propose to solve exactly the same problem 2. Recalling from Section 3.3, DynaMMo is trying to optimize the following objective function:

$$\begin{aligned} Q(\theta) &= \mathbb{E}_{\mathcal{X}_m,\mathcal{Z}|\mathcal{X}_g,\mathcal{W}}[P(\mathcal{X}_g,\mathcal{Z}_m,\mathcal{Z})] \\ &= \mathbb{E}_{\mathcal{X}_m,\mathcal{Z}|\mathcal{X}_g,\mathcal{W}}[-D(\vec{z}_1,z_0,\Gamma) - \sum_{t=2}^{\mathrm{T}} D(\vec{z}_t,\mathbf{F}\vec{z}_{t-1},\Gamma) \\ &\quad - \sum_{t=1}^{\mathrm{T}} D(\vec{x}_t,\mathbf{G}\vec{z}_t,\Sigma) - \frac{\log|\Gamma|}{2} - \frac{(\mathrm{T}-1)\log|\Lambda|}{2} - \frac{\mathrm{T}\log|\Sigma|}{2}] \end{aligned} \tag{4.1}$$

where $D()$ is the square of the Mahalanobis distance $D(\vec{x},\vec{y},\Sigma) = (\vec{x}-\vec{y})^T\Sigma^{-1}(\vec{x}-\vec{y})$ Here the unknowns include the hidden variable $\vec{z}'s$, missing value $\mathcal{X}_m$ (and their distribution), and model parameters. Our proposal is to find a better optimization order for fitting the data. Our preliminary experiments show very promising results.

4.2.2 Missing Values under Constraints

In this work, we want to answer the question: is there better recovering of missing values in situations with domain constraints. As a particular case, we study the problem of occlusion filling in human motion, where the markers are confined to fixed bone lengths on the human body. One problem with DynaMMo in this setting is that they do not always preserve inter-marker distances. While a joint-angle representation would solve this problem, it would both require that a skeleton (Figure 4.2) 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).

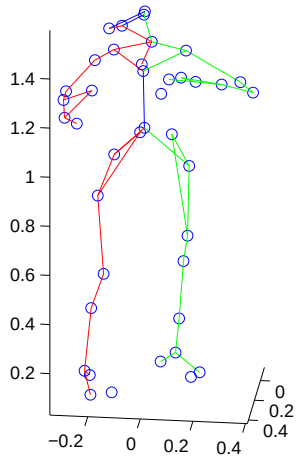


Figure 4.2: Markers on body skeleton.

We want to solve this problem in a 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.

One traditional method to estimate missing values is to optimize a squared loss function, penalizing the model complexity as Eq. 4.1. There are several choices with varying implications. How should we incorporate bone length constraints (BLC)? There are several choices with subtle issues. We choose to use hard constraints through the following formulation, since it will result in an efficient algorithm for solution. The problem is formally defined as follows:

Definition 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 2. *A pair of marker coordinates y_i and y_j ($\in \mathbb{R}^3$) are said to conform to the BLC B if*

$$\forall \langle i, j, d_{i,j}\rangle \in B \Rightarrow \| y_i - y_j \|^2 = d_{i,j}$$

Problem 4 (Bone-length constrained occlusion filling)**.**
Given *(a) $\mathcal{X}_g$ (the observed marker positions) and (b) B (bone length constraint),*
find *values for missing part $\mathcal{X}_m$ respecting the bone length constraints.*

4.3 Timeline

We plan to complete the proposed work according to the following tentative timeline:

- **December 2009:** Thesis proposal.
- **December 2009:** Study enhanced general missing value recovery and mining.
- **January 2010 - February 2010:** Study and analyze missing values under constraints, particularly bone-length constrained occlusion filling.
- **March 2010 - May 2010:** Study compact feature extraction from time series. Based on promising preliminary results, we study how to answer similarity queries efficiently on motion capture database.
- **June 2010 - November 2010:** Study sensor streams and monitoring. We will particularly investigate intrusion and anomaly detection in network traffic, and datacenter monitoring and control.
- **December 2010 - January 2011:** Write the thesis.
- **February 2011:** Thesis defense.

Chapter 5

Conclusion

In this proposal, we present fast algorithms on mining co-evolving time series, with or with out missing values. Our algorithms could mine meaningful patterns effectively and efficiently. With those patterns, our algorithms can do forecasting, compression, and segmentation. Furthermore, we apply our algorithm to solve practical problems including occlusions in motion capture, and generating natural human motions by stitching low-effort motions. We also propose a parallel learning algorithm for Linear Dynamical Systems, which will serve as corner stone of many applications and algorithms for time series.

Next, we will work on the proposed problems and find better and faster algorithms for them, such as missing values with bone length constraints, indexing and mining sensor streams.

Bibliography

L. A. Barroso and U. Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009. ISBN 159829556X, 9781598295566. 3

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1 edition, October 2006. ISBN 0387310738. 9, 19

G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, Englewood Cliffs, NJ, 3rd edition, 1994. 5

P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer Verlag, New York, 1987. 5

G. Buehrer, S. Parthasarathy, S. Tatikonda, T. Kurc, and J. Saltz. Toward terabyte pattern mining: an architecture-conscious solution. In *SIGPLAN*, pages 2–12. ACM, 2007. doi: http://doi.acm.org/10.1145/1229428.1229432. 6

E. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. Parallelizing support vector machines on distributed computers. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS*, pages 257–264. MIT Press, 2007. 6

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*, pages 281–288. MIT Press, 2006. 6

CMU. CMU multi-modal activity database, a. URL `http://kitchen.cs.cmu.edu`. 2, 9

CMU. CMU motion capture database, b. URL `http://mocap.cs.cmu.edu`. 2, 9, 13, 15, 16, 17, 18, 20, 21

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. 6

C. B. Colohan, A. Ailamaki, J. Gregory Steffan, and T. C. Mowry. Tolerating Dependences Between Large Speculative Threads Via Sub-Threads. In *33rd International Symposium on Computer Architecture (ISCA 2006), June 17-21, 2006, Boston, MA, USA*, pages 216–226, 2006. doi: 10.1109/ISCA.2006.43. 19

S. Cong, J. Han, and D. Padua. Parallel mining of closed sequential patterns. In *Proc. of the 11th ACM SIGKDD*, pages 562–567. ACM, 2005. doi: http://doi.acm.org/10.1145/1081870.1081937. 6

J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04*, 2004. 6

A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599, 2004. 5

DFC. Dfc intelligence forecasts video game market to reach $ 57 billion in 2009, June 2008. URL `http://www.dfcint.com/wp/?p=222`. 2

S. T. Dumais. Latent semantic indexing (LSI) and TREC-2. In *TREC-2*, pages 105–115, Gaithersburg, MD, Mar. 1994. NIST. Special publication 500-215. 5

EPA. Epa report to congress on server and data center energy efficiency. Technical report, U.S. Environmental Protection Agency, 2007. 3

C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, Minneapolis, MN, May 25-27 1994. 5

X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *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/. 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. 11

A. W.-C. Fu, E. J. Keogh, L. Y. H. Lau, and C. A. Ratanamahatana. Scaling and time warping in time series querying. In *VLDB*, pages 649–660, 2005. 5

M. Garofalakis, J. Gehrke, and R. Rastogi. *Data Stream Management: Processing High-Speed Data Streams*. Springer, 2009. 5

Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, February 1996. 9

A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *VLDB*, pages 79–88, 2001. 5

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, *NIPS*, pages 521–528. MIT Press, Cambridge, MA, 2005. 6

D. Gunopulos and G. Das. Time series similarity measures and time series indexing. In *SIGMOD Conference*, Santa Barbara, CA, 2001. Tutorial. 5

A. C. Harvey. *Forecasting, structural time series models, and the Kalman filter*. Cambridge University Press, Cambridge ; New York :, 1990. ISBN 0521405734 0521321964 0521405734. 6

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

M. Jahangiri, D. Sacharidis, and C. Shahabi. Shift-split: I/o efficient maintenance of wavelet-transformed multidimensional data. In *SIGMOD*, pages 275–286, 2005. 5

A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using kalman filters. In *SIGMOD*, pages 11–22, 2004. 1, 6

C. S. Jensen and S. Pakalnis. Trax - real-world tracking of moving objects. In *VLDB*, pages 1362–1365, 2007. 5

I. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986. 5

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. 9

K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of arima time-series.

In *ICDM*, pages 273–280, 2001. 5

E. J. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002. 5

E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*, Santa Barbara, CA, 2001. 5

E. J. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle. Indexing large human-motion databases. In *VLDB*, pages 780–791, 2004. 1, 5

G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. *PODS*, pages 261–272, 1999. 1

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

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 *SIGGRAPH '02*, pages 491–500, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-521-1. doi: http://doi.acm.org/10.1145/566570.566607. 10

J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *SIGKDD*, pages 420–429, San Jose, CA, USA, 2007. ACM. 1, 5

L. Li, W. Fu, F. Guo, T. C. Mowry, and C. Faloutsos. Cut-and-stitch: efficient parallel learning of linear dynamical systems on smps. In *KDD '08*, pages 471–479, New York, NY, USA, 2008a. ACM. ISBN 978-1-60558-193-4. doi: http://doi.acm.org/10.1145/1401890.1401949. 7, 19, 20

L. Li, J. McCann, C. Faloutsos, and N. Pollard. Laziness is a virtue: Motion stitching using effort minimization. In *Short Papers Proceedings of EUROGRAPHICS*, 2008b. 7, 9, 11, 12

L. Li, J. McCann, N. Pollard, and C. Faloutsos. Dynammo: Mining and summarization of coevolving sequences with missing values. In *KDD '09*, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-193-4. 7, 12, 15, 16

C. Liu, F. Guo, and C. Faloutsos. Bbm: bayesian browsing model from petabyte-scale data. In *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. 4

K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis. Continuous nearest neighbor monitoring in road networks. In *VLDB*, pages 43–54, 2006. 5

Ü. Y. Ogras and H. Ferhatosmanoglu. Online summarization of dynamic time series data. *VLDB J.*, 15 (1):84–98, 2006. 5

C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *PODS*, pages 159–168, 1998. 5

S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, hands-off stream mining. *VLDB*, Sept. 2003. 1, 3

S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. *VLDB*, 2005. 1, 3

D. Patnaik, M. Marwah, R. Sharma, and N. Ramakrishnan. Sustainable operation and management of data center chillers using temporal data mining. In *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

D. Rafiei and A. O. Mendelzon. Similarity-based queries for time series data. In *SIGMOD Conference*, pages 13–25, Tucson, AZ, 1997. 5

C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *HPCA '07*, pages 13–24. IEEE Computer Society, 2007. ISBN 1-4244-0804-0. doi: http://dx.doi.org/10.1109/HPCA.2007.346181. 6

H. Rauch. Solutions to the linear smoothing problem. *Automatic Control, IEEE Transactions on*, 8(4): 371–372, Oct 1963. ISSN 0018-9286. 9

G. Reeves, J. Liu, S. Nath, and F. Zhao. Managing massive time series streams with multiscale compressed trickles. *PVLDB*, 2(1):97–108, 2009. 1, 5

S. Reinhardt and G. Karypis. A multi-level parallel implementation of a program for finding frequent patterns in a large sparse graph. In *IPDPS*, pages 1–8, 2007. 6

A. Safonova and J. K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph.*, 26(3):106, 2007. 5

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. 9

J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. *SDM*, Apr. 2007. 1, 4

Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *SIGMOD*, pages 611–622, New York, NY, USA, 2004. ACM Press. ISBN 1581138598. doi: http://dx.doi.org/10.1145/1007568.1007637. 6

H. Tong. *Non-linear Time Series: A Dynamical System Approach*. Clarendon Press, Oxford, 1990. 5

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. 5

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. 11

J. M. VanBriesen. Chlorine levels data. URL `http://www.cs.cmu.edu/afs/cs/project/spirit-1/www/`. 3

J. Wang and B. Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. In *SCA*, pages 232–238, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. ISBN 1-58113-659-5. 5, 9

J. Wang and B. Bodenheimer. Computing the duration of motion transitions: an empirical approach. In *SCA*, pages 335–344, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. ISBN 3-905673-14-2. doi: http://doi.acm.org/10.1145/1028523.1028568. 9

B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. *ICDE*, pages 201–208, 1998. 5