DynaMMo: Mining and Summarization of Coevolving Sequences with Missing Values

Lei Li, James McCann, Nancy Pollard, Christos Faloutsos
School of Computer Science, Carnegie Mellon University
{leili, jmccann, nsp, christos}@cs.cmu.edu

ABSTRACT

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 dynamics, making our algorithm able to function even when there are missing values.

We performed experiments on both real and synthetic datasets 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.

Categories and Subject Descriptors: H.2.8 Database applications: Data mining I.2.6 Artificial Intelligence: Learning - parameter learning

General Terms: Algorithms; Experimentation.

Keywords: Time Series; Missing Value; Bayesian Network; Expectation Maximization (EM).

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 [15]). 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 han-
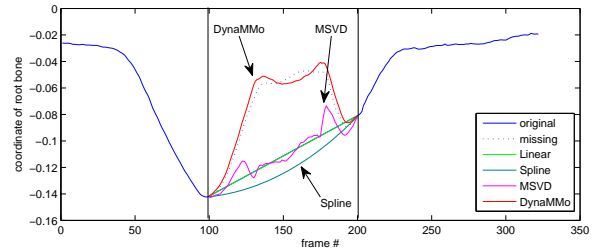
Figure 1: Reconstruction for a jump motion with 322 frames in 93 dimensions of bone coordinates. Blue line: the original signal for *root bone* z-coordinate - the dash portion indicates occlusion from frame 100 to 200. The proposed DynaMMo, in red, gets very close to the original, outperforming all competitors.

dling missing values in diverse settings: sensor data, chlorine levels in drinking water system, and other similar coevolving 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 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).

The focus of our work is to handle occlusions automatically. Straightforward methods like linear interpolation and spline interpolation give poor results (see Section 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 paper, we propose **DynaMMo**, an automatic method
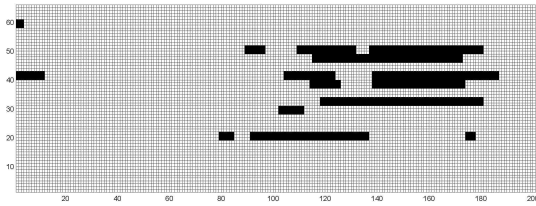
Figure 2: Occlusion in handshake motion. 66 joint angles (rows), for ≈ 200 frames. Dark color indicates a missing value due to occlusion. Notice that occlusions are clustered.

to learn the hidden pattern and handle missing values. Figure 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.

The rest of the paper is organized as follows: In Section 2, we review the related work; the proposed method and its discussion are presented in Section 3; the experimental results are presented in Section 4. Section 5 discusses additional benefits of our method: interpretation, compression and segmentation. Finally, Section 6 concludes the paper.

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) [20] 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 [18] 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 [1] further develop an incremental algorithm to fast compute the singular de-

composition with missing values. Similar to the missing value SVD approach, Liu and McMillan [13] 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 [16] have also used PCA to estimate the missing markers for skin deformation capturing. In another direction, Yi et al [21] have proposed a 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 [5] 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 [6] 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 [2] 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 [9] model the human motion using hierarchical Gaussian processes. [13] 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 [14, 12, 17], indexing [8], classification [3, 19] and outlier detection [10]. Mehta et al [14] 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) [12] is a symbolic representation for time series data, and later generalized for massive time series indexing (iSAX) [17]. Keogh et al use uniform scaling when indexing a large human motion database [8]. Lee et al [10] 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 [3] 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 [7] have adapted Kalman Filters for reducing communication cost in data stream. Tao et al [19] have proposed a recursive filter to predict and index moving objects. Li et al [11] 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. PROPOSED METHOD: DYNAMMO

Given a partially observed multi-dimensional sequence, we

Table 1: Symbols and Definitions

| Symbol	Definition
$\mathcal{X}$	a multi-dimensional sequence of observations with missing values $(\mathbf{x}_1, ... \mathbf{x}_T)$
$\mathcal{X}_g$	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 $(\mathbf{z}_1, \ldots \mathbf{z}_T)$
H	dimension of latent variables $(\mathbf{z}_1 \cdots \mathbf{z}_T)$

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 coevolving sequences.

3.1 The Model

We will first define the problem of time series missing value recovery, and then present our proposed **DynaMMo**. Table 1 explains the symbols and annotations.

DEFINITION 1. *Given a time sequence $\mathcal{X}$ with duration T in m dimensions, $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, to recover the missing part of the observations indicated by $\mathcal{W}$. $\mathbf{w}_{t,k} = 0$ whenever $\mathcal{X}$'s k-th dimensional observation is missing at time t, and otherwise $\mathbf{w}_{t,k} = 1$. 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) 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), $\mathbf{z}_n$, 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 (1 - 3) give the mathematical equations of our proposed model, with the
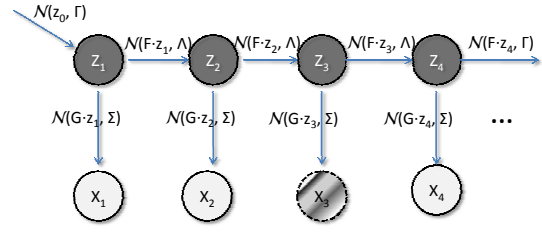
Figure 3: Graphical Illustration of the Model. $\mathbf{z}_{1\cdots4}$: latent variables; $\mathbf{x}_{1,2,4}$: observations; $\mathbf{x}_3$: partial observations. Arrows denote Gaussian distributions.

parameters $\theta = \{\mathbf{F}, \mathbf{G}, \mathbf{z}_0, \Gamma, \Lambda, \Sigma\}$.

$$\mathbf{z}_1 = \mathbf{z}_0 + \omega_0 \tag{1}$$
$$\mathbf{z}_{n+1} = \mathbf{F}\mathbf{z}_n + \omega_n \tag{2}$$
$$\mathbf{x}_n = \mathbf{G}\mathbf{z}_n + \epsilon_n \tag{3}$$

where $\mathbf{z}_0$ is initial state of the latent variables. $\mathbf{F}$ implies the transition and $\mathbf{G}$ is the observation projection. ω_0, ω_i and $\epsilon_i (i = 1 \ldots T)$ are multivariate Gaussian noises with the following distributions:

$$\omega_0 \sim \mathcal{N}(0, \Gamma) \quad \omega_i \sim \mathcal{N}(0, \Lambda) \quad \epsilon_j \sim \mathcal{N}(0, \Sigma) \tag{4}$$

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

$$P(\mathcal{X}_m, \mathcal{X}_g and \mathcal{Z}) = P(\mathbf{z}_1) \cdot \prod_{i=2}^{T} P(\mathbf{z}_i|\mathbf{z}_{i-1}) \cdot \prod_{i=1}^{T} P(\mathbf{x}_i|\mathbf{z}_i) \tag{5}$$

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

$$Q(\theta) = \mathbb{E}_{\mathcal{X}_m, \mathcal{Z}|\mathcal{X}_g, \mathcal{W}}[P(\mathcal{X}_g, \mathcal{Z}_m, \mathcal{Z})] \tag{6}$$

$$= \mathbb{E}_{\mathcal{X}_m, \mathcal{Z}|\mathcal{X}_g, \mathcal{W}}[-D(\mathbf{z}_1, z_0, \Gamma) - \sum_{t=2}^{T} D(\mathbf{z}_t, \mathbf{F}\mathbf{z}_{t-1}, \Gamma) \tag{7}$$

$$- \sum_{t=1}^{T} D(\mathbf{x}_t, \mathbf{G}\mathbf{z}_t, \Sigma) - \frac{\log|\Gamma|}{2} - \frac{(T-1)\log|\Lambda|}{2} - \frac{T\log|\Sigma|}{2}] \tag{8}$$

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

Our proposed **DynaMMo** searches for the optimal solution using Expectation-Maximization [4]. The optimization

algorithm is actually an iterative, coordinate descent procedure: estimating the latent variables, maximizing with respect to parameters, estimating the missing values, and iterating until convergence.

To estimate the parameters, taking the derivatives of Eq 7-8 with respect to the components of θ^{new} and setting them to zero yield the following results:

$$\mu_0^{new} = \mathbb{E}[\mathbf{z}_1] \tag{9}$$

$$\Gamma^{new} = \mathbb{E}[\mathbf{z}_1\mathbf{z}_1^T] - \mathbb{E}[\mathbf{z}_1]\mathbb{E}[\mathbf{z}_1^T] \tag{10}$$

$$\mathbf{F}^{new} = (\sum_{n=2}^{N} \mathbb{E}[\mathbf{z}_n\mathbf{z}_{n-1}^T])(\sum_{n=1}^{N-1} \mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T])^{-1} \tag{11}$$

$$\Lambda^{new} = \frac{1}{N-1}\sum_{n=2}^{N}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] - \mathbf{F}^{new}\mathbb{E}[\mathbf{z}_{n-1}\mathbf{z}_n^T]$$
$$-\mathbb{E}[\mathbf{z}_n\mathbf{z}_{n-1}^T](\mathbf{F}^{new})^T + \mathbf{F}^{new}\mathbb{E}[\mathbf{z}_n\mathbf{z}_{n-1}^T](\mathbf{F}^{new})^T) \tag{12}$$

$$\mathbf{G}^{new} = (\sum_{n=1}^{N}\mathbf{x}_n\mathbb{E}[\mathbf{z}_n^T])(\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T])^{-1} \tag{13}$$

$$\Sigma^{new} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{y}_n\mathbf{y}_n^T - \mathbf{G}^{new}\mathbb{E}[\mathbf{z}_n]\mathbf{y}_n^T$$
$$-\mathbf{y}_n\mathbb{E}[\mathbf{z}_n^T](\mathbf{G}^{new})^T + \mathbf{G}^{new}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T](\mathbf{G}^{new})^T) \tag{14}$$

The calculation of optimal parameters in Eq 9-14 requires estimation of latent variables, which includes our second goal. 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.

Finally, the missing values are easily computed from the estimation of latent variables using Markov property in the graphical model (Figure 3). We have the following equation:

$$\mathbb{E}[\mathcal{X}_m|\mathcal{X}_g, \mathcal{Z};\theta] = \mathbf{G} \cdot \mathbb{E}[\mathcal{Z}]_{\{i,j\}} (\{i,j\} \in \mathcal{W}) \tag{15}$$

The overall algorithm is described in Algorithm 1, omitting details explained in Appendix A.1.

3.3 Discussion

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

- MSVD: If we set $\mathbf{F}$ and z_0 to 0, and $\Gamma = \Lambda$, then the model becomes MSVD, We describe MSVD in Section 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 (1)-(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.

Penalty and Constraints: In the algorithm described above, Eq. (7) is error term for the initial state. Eq. (8) is trying to estimate the dynamics for the hidden states,

---

Algorithm 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 $\hat{\mathbf{z}}_1 \cdots \hat{\mathbf{z}}_{T|}$
- Model parameters θ

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 Γ, Λ, Σ to be identity matrix;
$\theta \leftarrow \{\mathbf{F}, \mathbf{G}, z_0, \Gamma, \Lambda, \Sigma\}$;
repeat

1   Estimate $\hat{\mathbf{z}}_{1\cdots T} = \mathbb{E}[\mathcal{Z}|\hat{X};\theta]$ using belief propagation (see details in Appendix A.1);

2   Maximizing Eq (7)-(8) with $\mathbb{E}[\mathcal{Z}|\hat{X};\theta]$ using Eq. 9-14,
$$\theta^{new} \leftarrow \arg\max_\theta Q(\theta);$$
     forall i,j **do**
       `// update the missing values`
       if $W_{i,j} = 0$ **then** $X_{i,j}$ is missing
         $\hat{X}_{i,j}^{new} \leftarrow (\mathbf{G}^{new} \cdot \mathbb{E}[\mathcal{Z}|\hat{X};\theta])_{i,j}$

until *converge* ;

---

while Eq. (8) is getting the best projection from observed motion sequence to hidden states. Eq. (8) 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.

4. EXPERIMENTAL RESULTS

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.

4.1 Experiment Setup

4.1.1 Baseline Methods

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

Missing Value SVD involves iteratively taking the SVD and fitting the missing values from the result [18]. This method is very easy to implement and already used on motion capture datasets in [13] and [16]. In our implementation (appendix A.2), we initialized the holes by linear interpolation, and use 15 principal dimensions (99% of energy).

4.1.2 Datasets

Chlorine Dataset (Chlorine): The Chlorine dataset (see sample in Figure 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 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.

4.1.3 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 2). 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.

4.2 Experimental Results

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

4.2.1 Qualitative result

Figure 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.

4.2.2 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 MSE is the
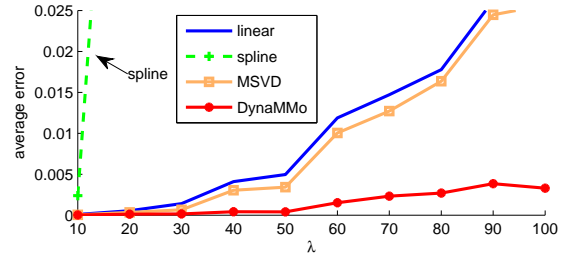
Figure 5: Average error for missing value recovery on a sample mocap data (subject#16.22). Average rmse over 10 runs, versus average missing length λ(from 10 to 100). Randomly 10.44% of the values are treated as "missing". DynaMMo (in red solid line) wins. Splines are off the scale.

average of squared differences between the actual (X) and reconstructed $(\hat{X})$ missing values - formally:

$$MSE(X, W, \hat{X}) = \frac{||(1 - W) \otimes (X - \hat{X})||^2}{||1 - W||^2}$$

$$= \frac{1}{\sum_{t,k}(1 - W_{t,k})} \sum_{t,k} (1 - W_{t,k})(X_{t,k} - \hat{X}_{t,k})^2 \qquad (16)$$

Both MSVD and our method use 15 hidden dimensions ($H = 15$). Figures 4(a)-4(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. It is worth to noting that the reconstruction grows little with increasing occlusion length, compared with other alternative methods (Figure 5). There is a similar result found in experiments on Chlorine data as shown in Figure 6(b). Again, our proposed **DynaMMo** achieves the best performance among the four methods.

4.2.3 Scalability

As we discussed in Section 3, the complexity of **DynaMMo** is $O(\#(iterations) \cdot T \cdot m^3)$. Figure 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.

5. ADDITIONAL BENEFITS

In addition to recovering the missing observations, our

(a) Sample Chlorine data. (b) Reconstruction error.
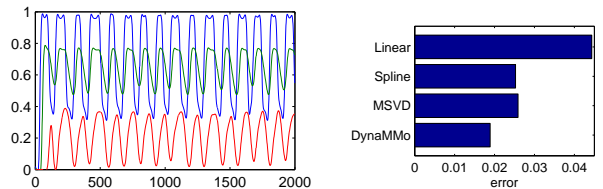
Figure 6: Reconstruction experiment on Chlorine with 10% missing and average occlusion length 40.

(a) DynaMMo v.s. Linear   (b) DynaMMo v.s. Spline   (c) DynaMMo v.s. MSVD
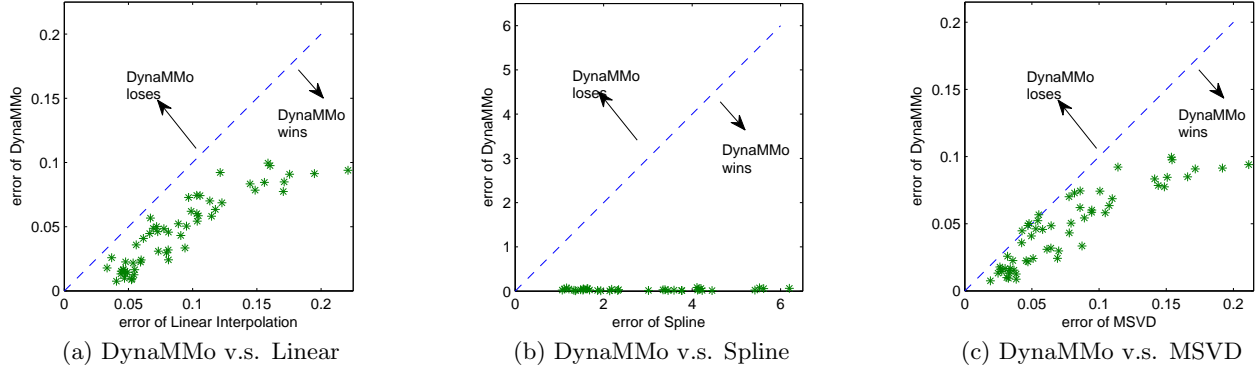
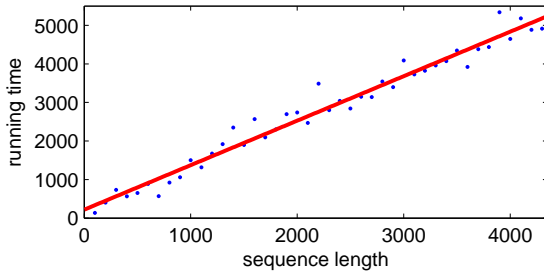Figure 4: Scatter plot of missing value reconstruction error for Mocap dataset.

Figure 7: Running time versus the sequence length on Chlorine dataset. For each run, 10% of the values are treated as "missing".
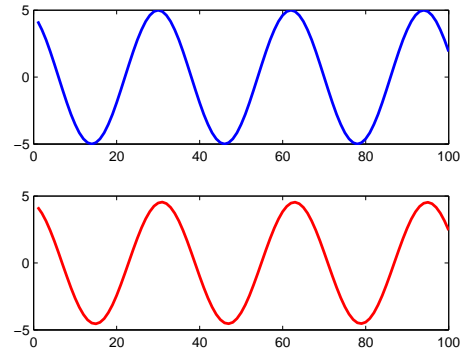

Figure 8: Simulated signal versus original signal. The top is the original signal (blue curve), while the bottom (red curve) is the generated using parameters learnt from original with a hidden dimension of 6. Note the almost identical pattern in both generated and original signal, and will continue if prolonged.

proposed **DynaMMo** method can be easily extended for further data mining tasks. Here we propose several **DynaMMo** extensions for time series compression, segmentation, and forecasting.

5.1 Interpretation and Forecasting

One of the advantages of our proposed **DynaMMo** is that its learnt representation of a data stream contains information about behavior of patterns and trends, as illustrated in the following examples. As shown in top of Figure 8, we create a sinusoid sequence with a cycle of 32, then learn the parameters $\theta = \{\mathbf{F}, \mathbf{G}, z_0, \Gamma, \Lambda, \Sigma\}$ using **DynaMMo** with hidden dimension of 6, and then generate a simulated signal $\hat{\mathbf{x}}$ using only the parameter $z_0, \mathbf{F}, \mathbf{G}$: $\hat{\mathbf{z}}_1 = z_0$, $\hat{\mathbf{z}}_{n+1} = \mathbf{F}\hat{\mathbf{z}}_n$, $\hat{\mathbf{x}}_n = \mathbf{G}\hat{\mathbf{z}}_n$. The simulated signal in Figure 8 presents an identical periodic pattern to the original signal, with negligible difference in amplitude. Note it is easy to forecast the future if we continue ticking the time. As this simple case demonstrates, **DynaMMo** is able to learn dynamics from a data stream and reproduce its behavior even without observations. This suggests that it could be used to compress time sequences, as we will discuss in the next section.

5.2 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, **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 2.

5.2.1 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 $\mathbf{F}$, Λ, $\mathbf{G}$ and Σ. Both covariance Λ and Σ are constrained to $\lambda^2 I$ and $\sigma^2 I$ respectively. It also stores the number k.

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.

5.2.2 Adaptive Compression: DynaMMo$_a$

The adaptive compression will first learn the hidden vari-

Algorithm 2: **DynaMMo**_Decompress

Input: $\hat{\mathbf{z}}_S$, hidden variables, indexed by $S \subseteq [1 \cdots \mathrm{T}]$,
     $\mathbf{F}$, $\mathbf{G}$.
Output: The decompressed data sequence $\hat{\mathbf{x}}_{1 \cdots \mathrm{T}}$

$\mathbf{y} \leftarrow \hat{\mathbf{z}}_1$;
for $n \leftarrow 1$ *to* T **do**
  if n *in* S **then**
    $| \;\; \mathbf{y} \leftarrow \hat{\mathbf{z}}_n$;
  else
    $\lfloor \;\; \mathbf{y} \leftarrow \mathbf{F} \cdot \mathbf{y}$;
  $\hat{\mathbf{x}}_n \leftarrow \mathbf{G} \cdot \mathbf{y}$;

---

ables 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 $\mathbf{F}$, $\boldsymbol{\Lambda}$, $\mathbf{G}$ and $\boldsymbol{\Sigma}$. Both covariance $\boldsymbol{\Lambda}$ and $\boldsymbol{\Sigma}$ 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.

5.2.3 *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 $\mathbf{F}$, $\boldsymbol{\Lambda}$, $\mathbf{G}$ and $\boldsymbol{\Sigma}$. Both covariance $\boldsymbol{\Lambda}$ and $\boldsymbol{\Sigma}$ 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.

5.2.4 *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 9 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.

5.3 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
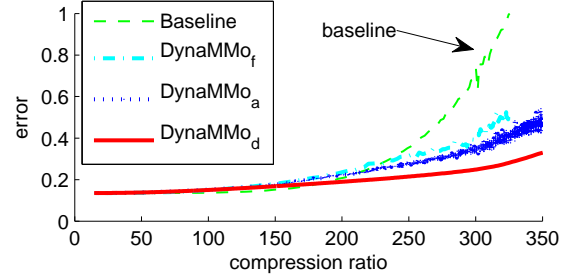
Figure 9: Compression for Chlorine dataset: RMSE versus compression ratio. Lower is better. DynaMMo$_d$ **(in red solid) is the best.**

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.

Algorithm 3: **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

$\{\mathbf{G}, \hat{\mathbf{z}}_{1 \cdots m}\} \leftarrow \mathbf{DynaMMo}(X, W, H)$;
for $n = 1$ *to* m **do**
  Reconstruct the data for time tick n: $\hat{\mathbf{x}}_n \leftarrow \mathbf{G} \cdot \hat{\mathbf{z}}_n$;
  Computer the reconstruction error for time tick n:

  $$\triangle_n \leftarrow \frac{||\mathbf{w}_n^T \otimes (\hat{\mathbf{x}}_n - \mathbf{x}_n)||^2}{\sum \mathbf{w}_n}$$

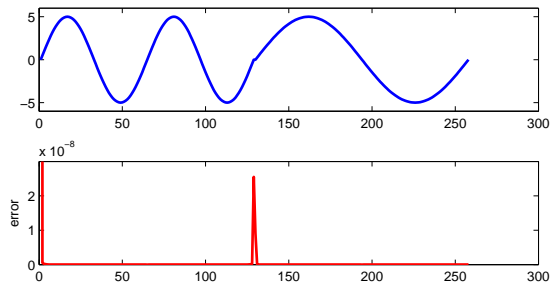find the split: $s \leftarrow \arg \max_k \triangle_k$;

To illustrate, Figure 10 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 11 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.

6. CONCLUSIONS

Given multiple time sequences, we propose **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.

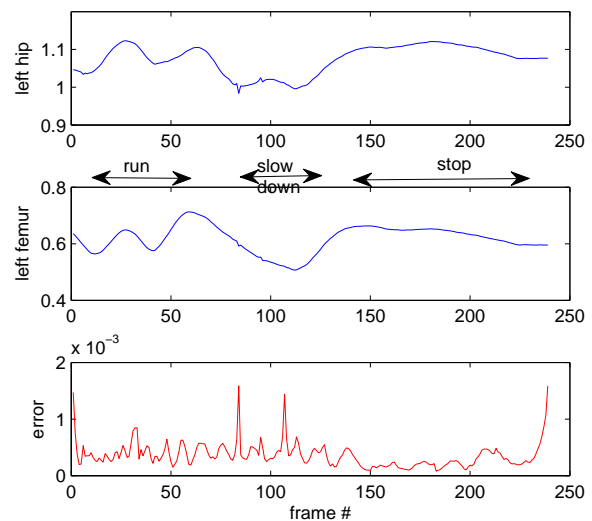We presented experiments on motion capture sequences and chlorine measurements and demonstrated that our pro-

Figure 10: 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 11: 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.

posed **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.

7. REFERENCES

[1] 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.

[2] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 686–696, New York, NY, USA, 2005. ACM.

[3] J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12(6):37–49, 2008.

[4] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120–127. Morgan Kaufmann Publishers, Inc., 1994.

[5] L. Herda, P. Fua, R. Plänkers, R. Boulic, and D. Thalmann. Skeleton-based motion capture for robust reconstruction of human motion. In *CA '00: Proceedings of the Computer Animation*, page 77, Washington, DC, USA, 2000. IEEE Computer Society.

[6] E. Hsu, S. Gentry, and J. Popović. Example-based control of human motion. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 69–77, Aire-la-Ville, Switzerland, 2004. Eurographics Association.

[7] A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using kalman filters. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 11–22, New York, NY, USA, 2004. ACM.

[8] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle. Indexing large human-motion databases. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 780–791. VLDB Endowment, 2004.

[9] 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.

[10] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. *IEEE 24th International Conference on Data Engineering*, pages 140–149, April 2008.

[11] 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.

[12] 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 Press.

[13] G. Liu and L. McMillan. Estimation of missing markers in human motion capture. *Vis. Comput.*, 22(9):721–728, 2006.

[14] S. Mehta, S. Parthasarathy, and R. Machiraju. On trajectory representation for scientific features. *IEEE International Conference on Data Mining*, 2006.

[15] S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, hands-off stream mining. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 560–571. VLDB Endowment, 2003.

[16] S. I. Park and J. K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Trans. Graph.*, 25(3):881–889, 2006.

[17] J. Shieh and E. Keogh. isax: indexing and mining terabyte sized time series. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, New York, NY, USA, 2008. ACM.

[18] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

[19] 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.

[20] 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.

[21] B.-K. Yi, N. D. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *ICDE '00: Proceedings of the 16th International Conference on Data Engineering*, page 13, Washington, DC, USA, 2000. IEEE Computer Society.

APPENDIX

A. APPENDIX

A.1 Algorithm 1 details

In Algorithm 1, the estimation (line 1) is to find the marginal distribution for hidden state variables given the data, e.g. $\hat{\mathbf{z}}_n = \mathbb{E}[\mathbf{z}_n \mid \mathcal{X}_g, \mathcal{X}_m](n = 1, \ldots, T)$. Since both prior and conditional distributions in the model are Gaussian, the posterior up to current time tick $p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_T)$ should also be Gaussian, denoted by $\hat{\alpha}(\mathbf{z}_n) = \mathcal{N}(\mu_n, \mathbf{V}_n)$. Let $p(\mathbf{x}_n|\mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$ denoted as c_n, We have the following propagation equation:

$$c_n\hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \int \hat{\alpha}(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})d\mathbf{z}_{n-1} \quad (17)$$

From Eq 17 we could obtain the following forward passing of the belief. The messages here are μ_n, $\mathbf{V}_n$ and $\mathbf{P}_{n-1}$(needed in later backward passing).

$$\mathbf{P}_{n-1} = \mathbf{F}\mathbf{V}_{n-1}\mathbf{F}^T + \Lambda \quad (18)$$
$$\mathbf{K}_n = \mathbf{P}_{n-1}\mathbf{G}^T(\mathbf{G}\mathbf{P}_{n-1}\mathbf{G}^T + \Sigma)^{-1} \quad (19)$$
$$\mu_n = \mathbf{F}\mu_{n-1} + \mathbf{K}_n(\mathbf{x}_n - \mathbf{G}\mathbf{F}\mu_{n-1}) \quad (20)$$
$$\mathbf{V}_n = (\mathbf{I} - \mathbf{K}_n)\mathbf{P}_{n-1} \quad (21)$$
$$c_n = \mathcal{N}(\mathbf{G}\mathbf{F}\mu_{n-1}, \mathbf{G}\mathbf{P}_{n-1}\mathbf{G}^T + \Sigma) \quad (22)$$

The initial messages are given by:

$$\mathbf{K}_1 = \mathbf{\Gamma}\mathbf{G}^T(\mathbf{G}\mathbf{\Gamma}\mathbf{G}^T + \Sigma)^{-1} \quad (23)$$
$$\mu_1 = \mu_0 + \mathbf{K}_1(\mathbf{x}_1 - \mathbf{G}\mathbf{F}\mu_0) \quad (24)$$
$$\mathbf{V}_1 = (\mathbf{I} - \mathbf{K}_1)\mathbf{\Gamma} \quad (25)$$
$$c_1 = \mathcal{N}(\mathbf{G}\mu_0, \mathbf{G}\mathbf{\Gamma}\mathbf{G}^T + \Sigma) \quad (26)$$

For the backward passing, let $\gamma(\mathbf{z}_n)$ denote the marginal posterior probability $p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_N)$ with the assumption:

$$\gamma(\mathbf{z}_n) = \mathcal{N}(\hat{\mu}_n, \hat{\mathbf{V}}_n) \quad (27)$$

The backward passing equations are:

$$\mathbf{J}_n = \mathbf{V}_n\mathbf{F}^T(\mathbf{P}_n)^{-1} \quad (28)$$
$$\hat{\mu}_n = \mu_n + \mathbf{J}_n(\hat{\mu}_{n+1} - \mathbf{F}\mu_n) \quad (29)$$
$$\hat{\mathbf{V}}_n = \mathbf{V}_n + \mathbf{J}_n(\hat{\mathbf{V}}_{n+1} - \mathbf{P}_n)\mathbf{J}_n^T \quad (30)$$

Hence, the expectation for Algorithm 1 line 1 are computed using the following equations:

$$\mathbb{E}[\mathbf{z}_n] = \hat{\mu}_n \quad (31)$$
$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_{n-1}^T] = \mathbf{J}_{n-1}\hat{\mathbf{V}}_n + \hat{\mu}_n\hat{\mu}_{n-1}^T \quad (32)$$
$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] = \hat{\mathbf{V}}_n + \hat{\mu}_n\hat{\mu}_n^T \quad (33)$$

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

From these estimations, the new parameter θ^{new} is obtain by maximizing the Equations 7-8 with respect to the components of θ^{new} given the current estimate of θ^{old}, yield the Algorithm 1 line 2.

A.2 Missing Value SVD

Algorithm 4: Missing Value SVD

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
        $\hat{X}_{i,j}^{new} \leftarrow Y_{i,j}$
until *converge* ;