

# Simultaneous Learning of Nonlinear Manifold and Dynamical Models from High-dimensional Time Series: A Variational Bayesian Approach

Rui Li, *Student Member, IEEE*, Tai-Peng Tian, *Student Member, IEEE*,  
and Stan Sclaroff, *Senior Member, IEEE*

## Abstract

The goal of this work is to learn a parsimonious and informative representation for high-dimensional time series. Conceptually, this comprises two distinct yet tightly coupled tasks: learning a low-dimensional manifold and modeling the dynamical process. These two tasks have a complementary relationship as the temporal constraints provide valuable neighborhood information for dimensionality reduction and conversely, the low-dimensional space allows dynamics to be learnt efficiently. Solving these two tasks simultaneously allows important information to be exchanged mutually. If nonlinear models are required to capture the rich complexity of time series, then the learning problem becomes harder as the nonlinearities in both tasks are coupled. The proposed solution approximates the nonlinear manifold and dynamics using piecewise linear models. The interactions among the linear models are captured in a graphical model. The model structure setup and parameter learning are done using a variational Bayesian approach, which enables automatic Bayesian model structure selection, hence solving the problem of over-fitting. By exploiting the model structure, efficient inference and learning algorithms are obtained without oversimplifying the model of the underlying dynamical process. Evaluation of the proposed framework with competing approaches is conducted in three sets of experiments: dimensionality reduction and reconstruction using synthetic time series, video synthesis using a dynamic texture database, and human motion synthesis, classification and tracking on a benchmark data set. In all experiments, the proposed approach provides superior performance.

## Index Terms

Bayesian learning, nonlinear manifold, nonlinear dynamical model, dynamic texture, human motion

## I. INTRODUCTION

Computer vision applications often deal with high-dimensional time series data. One example would be human motion capture sequences, where each frame records the 3D human body pose. As the human body has many degrees of freedom, the body pose parametrization is high-dimensional even with a simple body model. Usually the dimensionality of the pose vector is between 30-60. Another example would be video sequences. If we choose to represent each frame in a sequence by a data vector which is formed by stacking up all the pixels in the frames, then the dimensionality of the data vector used to represent a frame would easily exceed  $10^2$  even for a tiny  $10 \times 10$  pixel image.

However, we should note that the naive representations in these examples tend to have many redundancies given the strong spatial and temporal correlations in the data. These redundancies can be eliminated via dimensionality reduction techniques. Dimensionality reduction also has the benefit of reducing the computational complexity for most vision algorithms. Moreover, the accuracy of these algorithms may also be improved by removing the redundancies in the data representations. Therefore, dimensionality reduction is needed for these high-dimensional data.

Another key property of these high-dimensional time series data is that they exhibit complex dynamical behavior. For example, in human motion capture, the subjects can perform a wide range of complex motions and activities. Oftentimes, analysis of the dynamical behavior plays an important role in the computer vision applications, *e.g.*, in tracking human movement in video and classifying human motions, or in modeling and describing motion patterns in video for coding and retrieval applications, etc. For these applications to be successful, they must deal with the complex dynamics of these time series. And from a generative model point of view, it can be argued that high-dimensional time series can be economically represented by a dynamical process defined on a low-dimensional manifold.

Recovering such representations depends on two distinct yet tightly coupled tasks: reducing the dimensionality and modeling the dynamical process. We advocate for recovering the dynamical model parameters in concert with manifold learning. In isolation, recovering the dynamical model without dimensionality reduction is computationally inefficient. Conversely, dimensionality reduction without temporal information is “blind” since neighborhood information can only be approximated using Euclidean distance rather than using the knowledge of temporal neighbors. Solving these two tasks simultaneously allows important information to be exchanged mutually. However, as nonlinear models are required to capture the rich complexity of these time series, the learning problem becomes harder as the nonlinearities in both tasks are now coupled.

To make learning tractable, we propose to employ a divide and conquer approach: the nonlinear manifold is approximated by piecewise linear regions. Each local region is associated with its own linear dimensionality reducer and a linear dynamical model. Coordination among the local linear dimensionality reducers is needed to ensure consistent coordinates for the time series on the piecewise representation of the manifold, and to assure consistency among local linear dynamical models. Similarly, the linear dynamical models that approximate the nonlinear dynamical process on the manifold must be consistent with the observed high-dimensional time

series. Such coordination and consistency constraints are enforced by estimating the parameters of the piecewise linear models together with the coordination parameters during manifold learning. Learning of the coordinated, piecewise representation is efficient, without oversimplifying the model of the underlying dynamical process.

One major problem in learning multiple linear models to approximate the learning of the nonlinear dynamical process and nonlinear manifold is that it is hard to determine the optimal setup:

- *What is the best number of linear models that we should use for the approximation of the nonlinear dynamics and nonlinear manifold?*
- *What is the optimal dimension of the low-dimensional space?*
- *What is the appropriate order for the dynamical model?*

While the Bayesian approach provides a mechanism for automatic model selection, there is an integration step as noted in [1] that requires us to integrate over all the model parameters  $\theta$  when evaluating the marginal likelihood of the training data sequence  $\mathbf{x}_{1:T}$ . Such integration is often computationally intractable. Therefore we choose to use a variational Bayesian method and derive a learning algorithm to solve the model selection problem in our work.

Evaluation of our framework vs. competing approaches [2], [3] is conducted in experiments with three common data sets: dimensionality reduction and reconstruction for synthetic time series [2], synthesis of video textures [4], and human motion synthesis, classification and tracking on the benchmark of [5]. Compared to competing approaches, the data reconstruction error of the proposed model is at least one standard deviation smaller in the experiments conducted with both synthetic data and real data. When applied to human motion tracking, based on the error measurement defined in [5], our method yields a mean tracking error which is at least three standard deviations smaller. The performance of our variational Bayesian (VB) formulation is tested in dynamic texture analysis/synthesis and human motion tracking applications. In all the experiments, we obtain comparable performance for the optimal model structure found that is found automatically via our VB formulation, vs. models where the structure is empirically chosen (i.e., tuned by hand to yield the best results). Furthermore, the model learned using VB can be used effectively in labeling human action sequences and can yield results that are comparable to manually labeled data.

## II. RELATED WORK

We can formulate the problem of reducing the dimensionality and modeling the dynamical process in the dimensionality reduced space in a general dynamical process framework. Let  $\mathbf{x}_{1:T} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$  be the high-dimensional time series and  $\mathbf{g}_{1:T}$  be the corresponding low-dimensional time series. We use  $\mathbb{R}^D$  to denote the high-dimensional observation space and  $\mathbb{R}^d$  to represent the low-dimensional latent space, hence  $D \gg d$ . We define  $f_{dyn} : \mathbb{R}^d \rightarrow \mathbb{R}^d$  to be the nonlinear dynamical function that drives the low-dimensional time series; assuming a first-order Markov process, we have:

$$\mathbf{g}_t = f_{dyn}(\mathbf{g}_{t-1}) + \mathbf{n}_{\mathbf{g},t}, \quad (1)$$

where  $\mathbf{n}_{\mathbf{g},t}$  is a zero-mean, white Gaussian noise process. To map  $\mathbf{g}_t$  to observation  $\mathbf{x}_t$ , we define the nonlinear mapping function  $f_{\mathbf{g} \rightarrow \mathbf{x}} : \mathbb{R}^d \rightarrow \mathbb{R}^D$  to be:

$$\mathbf{x}_t = f_{\mathbf{g} \rightarrow \mathbf{x}}(\mathbf{g}_t) + \mathbf{n}_{\mathbf{x},t}, \quad (2)$$

where  $\mathbf{n}_{\mathbf{x},t}$  is also a zero-mean, white Gaussian noise process.

We first review the literature in learning  $f_{\mathbf{g} \rightarrow \mathbf{x}}$  and  $f_{dyn}$  separately, then we move on to discuss the related methods that aim to solve for  $f_{\mathbf{g} \rightarrow \mathbf{x}}$  and  $f_{dyn}$  simultaneously.

### A. Nonlinear dimensionality reduction

$f_{\mathbf{g} \rightarrow \mathbf{x}}$  lifts the low-dimensional representation to the high-dimensional observation space and it can be considered as the reverse function/process of dimensionality reduction  $f_{\mathbf{x} \rightarrow \mathbf{g}}$ .

Dimensionality reduction can be formulated as follows. Given a dataset represented in a  $N \times D$  matrix  $\mathbf{x}_{1:N}$  consisting of  $N$  data vectors  $\mathbf{x}_n$ , where  $n \in \{1, \dots, N\}$  and  $\mathbf{x}_n \in \mathbb{R}^D$ , the task of dimensionality reduction is to transform  $\mathbf{x}_{1:N}$  with dimensionality  $D$  into a new dataset  $\mathbf{g}_{1:N}$  with lower dimensionality  $d$  ( $d \ll D$ ) while retaining the geometry of the data as much as possible. In mathematical terms, intrinsic dimensionality means that the points in dataset  $\mathbf{x}_{1:N}$  are lying on or near a manifold with dimensionality  $d$  that is embedded in the  $D$ -dimensional space.

As a classic dimensionality reduction algorithm, Principal Component Analysis (PCA), is inadequate if the relationship between the low-dimensional representation and high-dimensional data is nonlinear. However, almost all the underlying low-dimensional representations and the corresponding high-dimensional data, especially in computer vision applications [6], [7], [2], [8],

[9], [10], have a nonlinear relationship. Therefore, we only review the nonlinear dimensionality reduction (NLDR) techniques here. We use the term “observation space” to refer to the high-dimensional data space and the term “latent space” to represent the low-dimensional space that has a nonlinear relationship with the observation space.

NLDR techniques can be classified into embedding-based versus mapping-based techniques. Embedding-based techniques [11], [12], [13], [14], [15], [16] model the structure of the data that generates the manifold without providing mapping functions between the observation space and the latent space. Given the nonlinear embedding result of data set  $\mathbf{x}_{1:N}$ , none of these methods can infer the embedding of samples that are not included in  $\mathbf{x}_{1:N}$ . This out-of-sample problem has greatly limited the applications of these methods. A straightforward solution would be expanding the dataset  $\mathbf{x}_{1:N}$  by adding in the new samples and computing the embedding on the new dataset [17], [18]. Since the nonlinear embedding is recomputed every time new samples are available, a lot of redundant computation is involved. This approach is generally not practical for time critical real world applications. A more practical solution would be using regression methods [6], [8] to learn the mapping functions after the embedding.

Mapping-based techniques learn the nonlinear mapping functions either by modeling the nonlinear functions directly [19], [20], [21] or by using a combination of local linear models [22], [23], [24] during dimensionality reduction. Mapping-based techniques allow interpolation between samples on the low-dimensional manifold by using the learned mapping functions from training data. However, these mapping-based NLDR algorithms assume that the data are independently and identically distributed (i.i.d.) even in applications where they are temporally correlated [7], [13], [9], [10]. Ignoring the temporal correlations causes inconsistencies in the learned manifold [2], [25], [26].

### *B. Nonlinear dynamical models*

Similar to the limitation of linear dimensionality reduction algorithms like PCA, linear dynamical systems (LDS) are not suitable for a large classes of problems. If there is nonlinearity in the time series, the nonlinear variations will be treated as noise in a LDS model, resulting in overly smoothed motion during simulation. To capture a richer set of dynamics, nonlinear dynamical models have been actively studied. Two main themes in nonlinear dynamical model learning are the use of a combination of linear models (*e.g.*, [3]), and the use of nonlinear functions directly

[27], [28]. There are two main problems with estimating dynamical model parameters in the observation space.

- 1) There might be redundant information and noise in the observation data and fitting a model to such data certainly affects the accuracy of estimated model parameters.
- 2) The high-dimensionality of the observation data increases the computational complexity of model parameter estimation.

### *C. Modeling high-dimensional time series*

In the manifold learning literature, some NLDR algorithms have been extended to incorporate temporal correlations during learning. Temporal Kohonen Maps [29] are suited to uncover structure in high-dimensional time series, but require *a priori* specification of expected structure topology. The algorithm proposed by Jenkins and Matarić [12] takes advantage of the temporal coherence between adjacent samples of the input time series. Their algorithm extends Isomap [16] by grouping temporally adjacent samples and favors temporally adjacent groups to have similar latent space coordinates. Though this algorithm does take advantage of the temporal ordering of points, it does not model the dynamics.

Recent work [25], [26] combines a dynamical model with the standard Gaussian Process Latent Variable Model (GPLVM) [20] by augmenting the GPLVM cost function with terms from a kernel matrix that models the dynamics in the latent space. In [25], [26], the dynamical model parameters are considered as incidental and are integrated out. Hence, this extension cannot be directly used for discriminative tasks such as classification. Another concern with the approaches of [25], [26] is the kernel sparsification problem, as there is no principled way to choose an active set from a time series. Without sparsification, the full kernel matrix has to be inverted at each iteration of learning. Thus, it is difficult to apply the dynamics extensions of GPLVM to large data sets. To avoid discontinuity problems caused by the use of an active set, Snelson and Ghaharmani propose sparsification techniques that make use of pseudo-inputs [30]. There are still two open problems with [30]: how to choose the number of pseudo-inputs, and how to avoid overfitting. Furthermore, the success of applying such techniques to computer vision problems has yet to be demonstrated.

Lin et al. [2] propose to use a piecewise linear model together with a global linear dynamical model in the latent space to learn a low-dimensional representation from high-dimensional time

series. Their work extends [23] by using a global linear dynamical model in the latent space together with learning of the mapping functions between the latent space and observation space. Their extension works well with large training data sets. However, the global linear dynamical model assumed in the latent space limits the types of dynamics that can be modeled.

In this paper, we take a step further and propose to approximate both  $f_{dyn}$  and  $f_{\mathbf{g} \rightarrow \mathbf{x}}$  using piecewise linear functions. The interaction among these linear models is formulated in a dynamic Bayesian network (DBN). Simultaneous learning of  $f_{dyn}$  and  $f_{\mathbf{g} \rightarrow \mathbf{x}}$  is formulated as the model parameter estimation problem in this DBN. In an earlier version of this work [31], we choose the number of linear models and the dimensionality of the latent space empirically. In the second half of this paper, we tackle the problem of automatically choosing the optimal model setup via a variational Bayesian (VB) learning formulation.

#### D. Variational Bayesian learning techniques

In MacKay's work [1], it is shown that we need to compute the posterior distribution over a set of models given some prior knowledge of the model structure  $p(m)$  and its associated model parameters  $p(\boldsymbol{\theta}|m)$  to do Bayesian model selection. Based on Bayes' rule, the posterior over model  $m$  given observation  $\mathbf{x}$  is:

$$p(m|\mathbf{x}) = \frac{p(m)p(\mathbf{x}|m)}{p(\mathbf{x})}, \quad (3)$$

where  $p(\mathbf{x}|m)$ , the marginal likelihood, which is a key quantity to compute in Bayesian learning. It is obtained via integrating out the model parameters:

$$p(\mathbf{x}|m) = \int d\boldsymbol{\theta} p(\boldsymbol{\theta}|m) p(\mathbf{x}|\boldsymbol{\theta}, m). \quad (4)$$

However, such integration is typically intractable to compute in almost all interesting models. Thus, approximation techniques have been proposed for practical Bayesian learning.

Among various practical Bayesian learning techniques, the VB learning method is fast and efficient compared to sampling based methods [32], [33]. Furthermore, it avoids the problem of basis-dependence from which maximum a posteriori (MAP) and Laplace methods suffer [34]. The approximation is obtained by using a simpler parameterized distribution to approximate the true distribution. The parameters of the simpler distribution are adjusted during optimization to get the approximation as close as possible to the true distribution. When this approximation is

applied in the context of Bayesian learning, we call it the variational Bayesian (VB) learning method. In MacKay’s early work [35], [36], this method is also called “ensemble learning”. This is in contrast with ML learning and MAP learning in which a point estimate of the model parameters is optimized.

Applications of VB learning have been demonstrated in various areas; a list of related work is provided in [37]. The model we propose in this paper is a generalized form of switching linear dynamical system (SLDS) [3].

To derive VB learning algorithms for the proposed models, earlier works on the VB learning for hidden Markov models (HMM) [38], mixture of factor analyzers (MFA) [39] and state-space models (SSM) [40] are relevant. The proposed model belongs to the conjugate-exponential (CE) family, where conjugate priors [41] are available. The benefit of having a model with parameters and parameters in the CE family is that the VB learning algorithm derived for our proposed model has almost the same complexity as the maximum likelihood learning algorithm as shown in [32].

The VB learning framework is called VB expectation-maximization (VBEM) because it is derived from the standard maximum likelihood EM algorithm. Similarly, the resulting VBEM algorithm has two key steps, namely a variational Bayesian expectation (VBE) step and a variational Bayesian maximization (VBM) step. The VBE step for our model is based on the results from [40] where the VBE step is obtained by applying a modified version of forward-backward algorithm (a form of belief propagation). The VBM step is identical to the standard EM algorithm and has the same computational complexity.

### III. PROBLEM FORMULATION

In Table I, we define the variables that we will use in the derivation.

Based on the general formulation we introduced in the beginning of Section II, we propose to approximate  $f_{dyn}$  and  $f_{g \rightarrow x}$  using piecewise linear functions. The interactions among the linear functions are formulated in a graphical model as shown in Figure 1 (c). Simultaneous learning of  $f_{dyn}$  (dynamical process) and  $f_{g \rightarrow x}$  (and hence  $f_{x \rightarrow g}$  if such mapping is bi-directional) is formulated as the model parameter estimation problem in this graphical model.

The model defined in Figure 1 (c) is not just a simple temporal extension of the model defined in Figure 1 (b), which is a variant [42] of the globally coordinated mixture of factor analyzers

TABLE I  
VARIABLES USED IN THE MODEL.

Symbol	Size	Description
$\mathbf{x}_t$	$D \times 1$	observation vector at time $t$
$\mathbf{x}_{1:T}$	$D \times T$	sequence of observation vectors
$\mathbf{g}_t^{(s)}$	$d \times 1$	continuous state vector of the $s$ -th linear dynamical system (LDS) at time $t$
$\mathbf{g}_t$	$d \times S$	$\mathbf{g}_t = [\mathbf{g}_t^{(1)}, \dots, \mathbf{g}_t^{(s)}, \dots, \mathbf{g}_t^{(S)}]$
$\mathbf{g}_{1:T}$	$d \times T$	sequence of continuous state vectors
$\mathbf{s}_t$	$S \times 1$	switching state variable, where $\mathbf{s}_t = [s_t^{(1)}, \dots, s_t^{(s)}, \dots, s_t^{(S)}]^\top$ , $s_t^{(s)} \in \{0, 1\}$
$\mathbf{s}_{1:T}$	$S \times T$	sequence of switching state variables

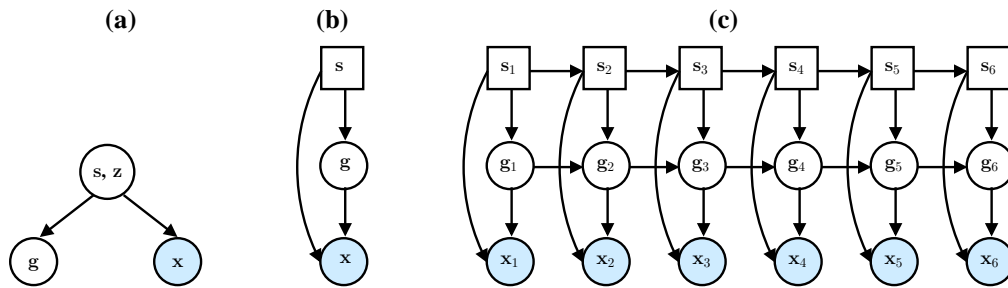


Fig. 1. Graphical models of MFA, GCMFA and our proposed model. (a). The original globally coordinated mixture of factor analyzers (MFA) proposed by Roweis *et al.* [23]. (b). The modified model for the globally coordinated MFA suggested by Verbeek in [42]. (c). Our proposed latent dynamical model [31]. Square nodes are hidden discrete states while the circle nodes are hidden nodes. The shaded nodes are observations. This model enables the idea of recovering the model parameters of the nonlinear dynamical process in concert with the manifold learning, and the nonlinearities of the processes are approximated by switching among multiple linear models in a coordinated way.

(MFA) proposed by Roweis *et al.* [23]. The key ideas are:

- 1) The nonlinear dynamical process is defined on the low-dimensional latent space; hence, it exploits temporal correlation during manifold learning and enables the recovery of the model parameters of the dynamical process in concert with the manifold learning.
- 2) By switching among linear dynamical processes and linear dimensionality reducers in a coordinated way, the proposed model is able to capture the nonlinearities that exist in both the dynamical process and the mappings between the latent space and the observation space.

In the following section, we will show the full derivation of  $f_{dyn}$  together with the mapping

functions of the proposed model shown in Figure 1 (c).

### A. Incorporating Dynamics

Given the model depicted in Figure 1(b), we can extend it incorporate temporal information in the form of the graphical model shown in Figure 1(c). Now the observations  $\{\mathbf{x}_t\}$  form a temporal sequence generated from the collaboration of the discrete Markov process  $\{s_t\}$  and continuous Markov process  $\{\mathbf{g}_t\}$ . Simultaneous learning of the nonlinear dynamics and nonlinear manifold is achieved by modeling the interactions among the linear models that define both the dynamics and the forward/backward mappings between  $\mathbf{g}_t$  and observation  $\mathbf{x}_t$ .

The discrete state variables are indicator vectors, the entry of corresponding state label is set to 1 and while rest are set to 0 For  $\mathbf{s}_{1:T} = \{\mathbf{s}_1, \dots, \mathbf{s}_T\}$ ,  $\mathbf{g}_{1:T} = \{\mathbf{g}_1, \dots, \mathbf{g}_T\}$  and  $\mathbf{x}_{1:T} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$ , the joint distribution for the graphical model shown in Figure 1(c) is defined as:

$$p(\mathbf{s}_{1:T}, \mathbf{g}_{1:T}, \mathbf{x}_{1:T}) = p(\mathbf{s}_1) \prod_{t=2}^T p(\mathbf{s}_t | \mathbf{s}_{t-1}) \times p(\mathbf{g}_1 | \mathbf{s}_1) \prod_{t=2}^T p(\mathbf{g}_t | \mathbf{g}_{t-1}, \mathbf{s}_t) \prod_{t=1}^T p(\mathbf{x}_t | \mathbf{g}_t, \mathbf{s}_t). \quad (5)$$

The dynamical system is defined on the globally coordinated space with the observations being tied to the individual factor analyzers. The following set of state-space equations describes the dynamical system:

$$\begin{aligned} \mathbf{g}_t &= \mathbf{F}(\mathbf{s}_t) \mathbf{g}_{t-1} + \mathbf{n}_{\mathbf{g},t}(\mathbf{s}_t), \quad t > 1, \\ \mathbf{g}_1 &= \mathbf{n}_1(\mathbf{s}_1), \quad t = 1, \\ \mathbf{x}_t &= \mathbf{\Lambda}(\mathbf{s}_t)(\mathbf{g}_t - \boldsymbol{\kappa}(\mathbf{s}_t)) + \boldsymbol{\mu}(\mathbf{s}_t) + \mathbf{n}_{\mathbf{x},t}, \quad \forall t. \end{aligned} \quad (6)$$

$\mathbf{\Lambda}(\mathbf{s}_t)$ ,  $\boldsymbol{\mu}(\mathbf{s}_t)$  and  $\boldsymbol{\kappa}(\mathbf{s}_t)$  are globally coordinated MFA parameters that parameterize the mapping  $f_{\mathbf{g} \rightarrow \mathbf{x}}$ , and  $\mathbf{F}(\mathbf{s}_t)$  represents the piecewise linear approximation of  $f_{dyn}$ . The corresponding noise processes are assumed to be independently distributed Gaussians, where  $\mathbf{n}_{\mathbf{g},t}(\mathbf{s}_t) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\mathbf{s}_t))$  for  $t > 1$ ,  $\mathbf{n}_1(\mathbf{s}_1) \sim \mathcal{N}(\mathbf{g}_1(\mathbf{s}_1), \boldsymbol{\Sigma}_1(\mathbf{s}_1))$  for  $t = 0$  and  $\mathbf{n}_{\mathbf{x},t} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$ ,  $\forall t$ . Therefore, the model parameters we need to estimate now are

$$\boldsymbol{\theta} = \{ \{ \mathbf{F}^{(s)}, \boldsymbol{\Sigma}^{(s)}, \mathbf{\Lambda}^{(s)}, \boldsymbol{\kappa}^{(s)}, \boldsymbol{\mu}^{(s)} \}_{s=1}^S, \boldsymbol{\Psi}, \boldsymbol{\Sigma}_1, \boldsymbol{\pi}, \mathbf{A} \}$$

for the model shown in Figure 1(c). Table II lists the details of individual model parameters.

TABLE II  
MODEL PARAMETERS.

Symbol	Size	Description
$\mathbf{F}^{(s)}$	$d \times d$	state dynamic matrix of the $s$ -th LDS
$\mathbf{\Lambda}^{(s)}$	$D \times d$	measurement/factor loading matrix of the $s$ -th LDS and factor analyzer (FA)
$\boldsymbol{\kappa}^{(s)}$	$d \times 1$	state mean of $s$ -th FA in the low-dimensional space
$\boldsymbol{\Sigma}^{(s)}$	$d \times d$	state covariance of $s$ -th FA in the low-dimensional space
$\boldsymbol{\mu}^{(s)}$	$D \times 1$	state mean of $s$ -th FA in the observation space
$\boldsymbol{\Psi}$	$D \times D$	observation noise
$\mathbf{A}$	$S \times S$	state transition matrix for the switching state
$\boldsymbol{\pi}$	$S \times 1$	initial state distribution for the switching state
$\boldsymbol{\Sigma}_1$	$d \times d$	initial state covariance matrix at $t = 1$

#### IV. VARIATIONAL MAXIMUM LIKELIHOOD LEARNING ALGORITHM OF THE MODEL

Without worrying about the model selection problem, the learning problem can be formulated in a maximum likelihood learning framework, *i.e.*, we need to solve the learning problem  $\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} \log p(\mathbf{x}_{1:T}|\boldsymbol{\theta})$ , and the inference problem  $p(\mathbf{s}_{1:T}, \mathbf{g}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta})$  by computing the joint distribution of the hidden state sequence  $\mathbf{s}_{1:T}$  and  $\mathbf{g}_{1:T}$  given the observation sequence  $\mathbf{x}_{1:T}$  and model parameters  $\boldsymbol{\theta}$ . However, exact inference is intractable for the graphical model defined Figure 1(c) [43].

To make the learning tractable, we take a variational approach to learn the model parameters and optimize the lower bound of the log likelihood by applying Jensen's inequality,  $\log p(\mathbf{x}_{1:T}|\boldsymbol{\theta}) \geq \Phi$ , where

$$\begin{aligned}
\Phi &= \sum_{\mathbf{s}_{1:T}} \int q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta}) \log \left( \frac{p(\mathbf{s}_{1:T}, \mathbf{g}_{1:T}, \mathbf{x}_{1:T}|\boldsymbol{\theta})}{q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta})} \right) d\mathbf{g}_{1:T} \\
&= \sum_{\mathbf{s}_{1:T}} \left[ \int q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta}) \log p(\mathbf{g}_{1:T}, \mathbf{s}_{1:T}, \mathbf{x}_{1:T}|\boldsymbol{\theta}) d\mathbf{g}_{1:T} \right. \\
&\quad \left. - \int q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta}) \log q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta}) d\mathbf{g}_{1:T} \right]. \tag{7}
\end{aligned}$$

$q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta})$  is an approximation of  $p(\mathbf{g}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta})$ . Hence, the first term of Equation 7 approximates the expected log-likelihood of the standard EM algorithm. The second term can be regarded as a regularization term as it models the entropy of the approximate variational distribution. An outline of the learning algorithm is given in Algorithm 1.

---

**Algorithm 1 Variational EM Learning Algorithm**


---

1: **E-step:** Variational inference to obtain the approximate posterior distribution:

$$p(\mathbf{g}_{1:T}, \mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}^i) \approx q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}^i). \quad (8)$$

2: **M-step:** Maximize  $\Phi$  with respect to  $\boldsymbol{\theta}$ :

$$\boldsymbol{\theta}^{i+1} = \arg \max_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}^i). \quad (9)$$


---

While there are many possible approximations one could use for  $q$ , we focus on the following factorized form of  $q$  based on variational mean-field theory [44]:

$$q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}) = q(\mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}) q(\mathbf{g}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}). \quad (10)$$

As in mean field approximations, the original stochastically coupled system (Figure 1(c)) now is approximated by two decoupled models. One is a hidden Markov model (HMM) defined on  $\mathbf{s}_{1:T}$  with a set of variational parameters  $\eta_{\mathbf{s}_{1:T}} = \{p_1, \dots, p_T\}$ , where  $p_1, \dots, p_T$  are the output probabilities. The other is a linear dynamic system (LDS) defined on  $\mathbf{g}_{1:T}$  with a set of variational parameters:

$$\eta_{\mathbf{g}_{1:T}} = \{\hat{\mathbf{g}}_1, \hat{\boldsymbol{\Sigma}}_1, \hat{\boldsymbol{\Sigma}}_2, \dots, \hat{\boldsymbol{\Sigma}}_T, \hat{\mathbf{F}}_2, \dots, \hat{\mathbf{F}}_T, \hat{\boldsymbol{\Lambda}}_1, \dots, \hat{\boldsymbol{\Lambda}}_T, \hat{\boldsymbol{\mu}}_1, \dots, \hat{\boldsymbol{\mu}}_T\}.$$

The corresponding submodels are illustrated in Figure 2 (a) and (b). This approximation is based on the decomposition of the model structure and not all variables are uncoupled. The coupling of discrete hidden variables and continuous hidden variables is captured through the passing of variational parameters between the submodels.

In our model,  $\boldsymbol{\Psi}$  is considered as an observation noise covariance matrix and is the same for every dynamical process; therefore it is not included in  $\eta_{\mathbf{g}_{1:T}}$ . The  $\boldsymbol{\kappa}^{(s)}$ 's are also not included in  $\eta_{\mathbf{g}_{1:T}}$  as they are the inputs to the dynamical processes and can be calculated directly from the HMM submodel.

The expectation of the joint log likelihood  $\mathcal{L} = \log p(\mathbf{s}_{1:T}, \mathbf{g}_{1:T}, \mathbf{x}_{1:T})$  with respect to  $q(\mathbf{g}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta})$  has the form of the joint log-likelihood function of an HMM, and similarly the expectation of  $\mathcal{L}$  with respect to  $q(\mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta})$  has the form of the joint log-likelihood function of a time-varying LDS. As a result, we can derive alternating updates for  $\eta_{\mathbf{s}_{1:T}}$  and  $\eta_{\mathbf{g}_{1:T}}$ . Given the HMM sufficient statistics for  $p(\mathbf{s}_t)$ , we can obtain the time-varying LDS parameters  $\eta_{\mathbf{g}_{1:T}}$  and vice versa.

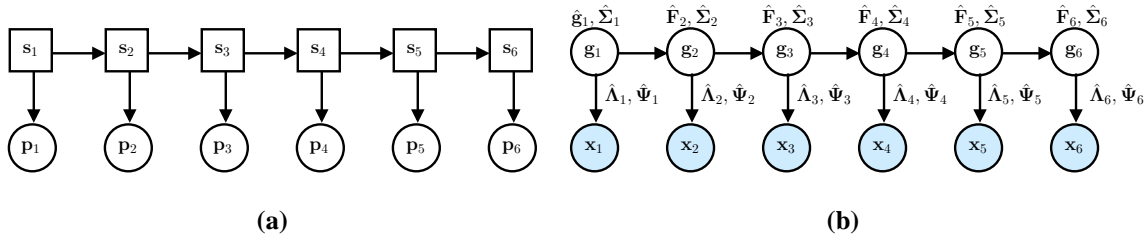


Fig. 2. The structural decomposition that corresponds to the factorization of  $q(\mathbf{g}_{1:T}, \mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}) = q(\mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta})q(\mathbf{g}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta})$ : (a). HMM submodel; (b). LDS submodel (Note that  $\boldsymbol{\mu}^{(s)}$  and  $\boldsymbol{\kappa}^{(s)}$  do not appear in the output mapping arrow because they are considered as the inputs to the LDS submodel.).

We only summarize the inference algorithm in Algorithm 2 as the detailed update equations can be derived by following the formulas provided in [45], [46]. However, the state space of the dynamical processes in our model is different. As the state resides in a low-dimensional space, the measurement function of the dynamical process now is the mapping function between the low-dimensional latent space and the high-dimensional observation space.

---

### Algorithm 2 Variational Inference Algorithm

---

- 1: error = inf;
  - 2: Initialize  $p(s_t | \boldsymbol{\pi}, \mathbf{A}, \eta_{\mathbf{s}_{1:T}})$ :
    - Apply the algorithm proposed in [13] to obtain the low-dimensional coordinates  $\mathbf{g}$ 's.
    - Do vector quantization to initialize the HMM as in [3].
  - 3: **while** error > maxError **do**
  - 4: Compute  $\eta_{\mathbf{g}_{1:T}}$  from  $p(\mathbf{s}_{1:T} | \boldsymbol{\pi}, \mathbf{A}, \eta_{\mathbf{s}_{1:T}})$ .
  - 5: Run LDS smoother to obtain  $p(\mathbf{g}_{1:T} | \mathbf{x}_{1:T}, \eta_{\mathbf{g}_{1:T}}, \{\boldsymbol{\kappa}^{(s)}\}_{s=1}^S, \boldsymbol{\Psi})$  using the method of [47];
  - 6: Update:  $q(\mathbf{g}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}) \leftarrow p(\mathbf{g}_{1:T} | \mathbf{x}_{1:T}, \eta_{\mathbf{g}_{1:T}}, \{\boldsymbol{\kappa}^{(s)}\}_{s=1}^S, \boldsymbol{\Psi})$ ;
  - 7: Compute  $\eta_{\mathbf{s}_{1:T}}$  from the LDS sufficient statistics.
  - 8: Compute  $p(\mathbf{s}_{1:T} | \boldsymbol{\pi}, \mathbf{A}, \eta_{\mathbf{s}_{1:T}})$  using the standard forward-backward algorithm for HMMs [48];
  - 9: Update:  $q(\mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}) \leftarrow p(\mathbf{s}_{1:T} | \boldsymbol{\pi}, \mathbf{A}, \eta_{\mathbf{s}_{1:T}})$ ;
  - 10: Update approximation error based on KL divergence between the updated  $q(\mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}) \cdot q(\mathbf{g}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta})$  and the  $q(\mathbf{s}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta}) \cdot q(\mathbf{g}_{1:T} | \mathbf{x}_{1:T}, \boldsymbol{\theta})$  before updating.
  - 11: **end while**
- 

## V. EXPERIMENTS WITH THE MODELS LEARNED VIA THE VARIATIONAL ML LEARNING ALGORITHM

Comparative studies with competing approaches [13], [3] are carried out in three sets of experiments to demonstrate the advantages of our approach.

We use DGCM to denote Lin *et al.*'s approach [2] and SLDS to denote the switching linear dynamic system proposed in [3]. We obtained the SLDS code from the authors of [3] and we implemented the DGCM proposed in [2]. All three approaches are implemented in un-optimized Matlab. As EM or coordinate ascent algorithms are used in all three approaches, proper initialization is necessary. To initialize the model in our approach and DGCM, the dimensionality of the latent space is chosen experimentally. To avoid over-fitting, we adopt a variational Bayesian approach [32] to choose the number of mixture components automatically for our approach and DGCM. We follow the technique proposed in [3] to initialize the SLDS. First order linear dynamical systems (LDS) are used in all three approaches.

The results reported are based on our implementations. Experiments are conducted on a PC with Intel dual-core 3.46 GHz CPU with 4 GB memory.

### A. Synthetic Data

TABLE III

SYNTHETIC DATA EXPERIMENTS SETUP (SECTION V-A). NUMBER OF STATES IN DGCM REFER TO THE NUMBER OF FACTOR ANALYZERS IN THE MIXTURE. IN OUR METHOD (MTD.), EACH STATE COMPRISES LINEAR DYNAMICAL SYSTEM WITH THE LATENT STATE SPACE.

	DGCM	Our Mtd.
Num. of Training Data	1500	1500
Dim. of Training Data $\mathbf{x}$	3	3
Dim. of Latent Coordinate $\mathbf{g}$	2	2
Num. of States	10	10
Training Time	$\sim 3$ min	$\sim 5$ min

TABLE IV

COMPARISON OF DIMENSIONALITY REDUCTION ( $f_{\mathbf{x} \rightarrow \mathbf{g}}$ ) AND RECONSTRUCTION ( $f_{\mathbf{g} \rightarrow \mathbf{x}}$ ). MSE STANDS FOR MEAN SQUARED ERROR AND  $\sigma$  STANDS FOR STANDARD DEVIATION OF MSE.

	2D		3D		3D	
	$f_{\mathbf{x} \rightarrow \mathbf{g}}$		$f_{\mathbf{g} \rightarrow \mathbf{x}}$		$f_{\mathbf{g} \rightarrow \mathbf{x}}(f_{\mathbf{x} \rightarrow \mathbf{g}})$	
	MSE	$\sigma$	MSE	$\sigma$	MSE	$\sigma$
DGCM	0.2958	0.2234	1.1993	0.7387	1.2347	0.7491
Our Mtd.	0.1102	0.0913	0.4854	0.2291	0.6507	0.6192

A synthetic data set is used in this experiment to quantify the information loss of dimensionality reduction and reconstruction. The data set is similar to the one used by [2]. 1500 2D data points are generated by a 2D random walk bounced off the boundaries in a patch  $[-2.5, 2.5] \times [-3, 3]$ . The bouncing at the boundaries introduces nonlinear motion. The 2D data

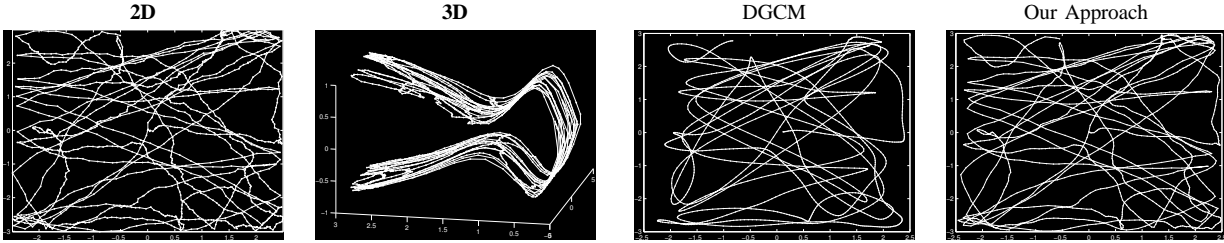


Fig. 3. Visualization of the ground truth synthetic data set.

Fig. 4. Visualization of the 2D trajectories obtained by applying  $f_{x \rightarrow g}$  learnt by DGCM and our approach on the ground truth 3D training data shown in Figure 3.

are then lifted to 3D by a mapping function  $f(x, y) = (x, |y|, \sin(\pi y)(y^2 + 1)^{-2} + 0.3y)$ . Figure 3 provides a visualization of the ground truth data set. The 1500 3D ground truth points are used as training data. We compare our approach with DGCM and Table III shows the setup of the experiment.

To quantitatively evaluate the mapping  $f_{x \rightarrow g}$ , we compute the mean squared error (MSE) between the ground truth 2D data and the inferred 2D data. Similarly, to evaluate  $f_{g \rightarrow x}$ , the MSE is computed between the ground truth 3D data and by applying  $f_{g \rightarrow x}$  on the ground truth 2D data to reconstruct the 3D sequence. Finally, to evaluate the bidirectional mapping,  $f_{g \rightarrow x}(f_{x \rightarrow g})$ , the MSE is computed on the 3D data by first applying  $f_{x \rightarrow g}$  to the ground truth 3D data and then applying  $f_{g \rightarrow x}$  to reconstruct 3D data from the inferred 2D data points. These error statistics are reported in Table IV. The mapping functions learnt by our approach are more accurate in terms of smaller MSE and standard deviation  $\sigma$ . In all cases, our approach cuts the MSE by more than half.

In the visualization (Figure 4) of inferred 2D trajectories. The  $f_{x \rightarrow g}$  learnt by our approach produces a 2D trajectory that is closer to the ground truth 2D trajectory (Figure 3). The mapping function learnt by DGCM produces an overly smoothed trajectory. This is because the switching of the linear dynamical models used in our approach is able to capture the sudden bouncing motion that occurs at the patch boundaries more accurately. This leads to the overall improvement in terms of smaller MSE and  $\sigma$  over DGCM.

TABLE V

EXPERIMENTAL SETUP OF EXPERIMENT (SECTION V-B)  
WITH DYNAMIC TEXTURE.

	DGCM	Our Approach
Len. of the Flag Seq.	250	250
Len. of the Wave Seq.	350	350
Dim. of $\mathbf{x}$	104256(= 288 × 362)	104256
Dim. $\mathbf{g}$	20	20
Num. of States	3	3
Training Time	~ 5 min	~ 8 min

TABLE VI

COMPARISON OF RECONSTRUCTION ERROR FROM TRAINING  
FRAMES.

	Flag Sequence		Wave Sequence	
	Mean Err.	$\sigma$	Mean Err.	$\sigma$
DGCM	0.0249	0.0378	0.0300	0.0316
Our approach	0.0161	0.0258	0.0210	0.0235

### B. Dynamic Texture

We can synthesize data on the manifold by using  $f_{dyn}$  to generate time series in the low-dimensional latent space. We can then use  $f_{\mathbf{g} \rightarrow \mathbf{x}}$  to map the low-dimensional time series back to the high-dimensional observation space. In this experiment, videos from a dynamic texture database [4] are used for training DGCM and our method. We then synthesize textures from the trained models. Table V shows the setup of the experiment.

Similar to Section V-A, we quantify the information loss by computing the MSE of normalized intensity values (range from 0 to 1) between the training video frames and the reconstructed frames. The reconstructed frames are obtained by first applying  $f_{\mathbf{x} \rightarrow \mathbf{g}}$  to the training video frames to get the coordinates in the latent space, and then applying  $f_{\mathbf{g} \rightarrow \mathbf{x}}$  to the latent coordinates. The error statistics of  $f_{\mathbf{x} \rightarrow \mathbf{g}}$  and  $f_{\mathbf{g} \rightarrow \mathbf{x}}$  are not evaluated separately as there is no ground truth low-dimensional data. The error statistics are shown in Table VI. We can see that the images reconstructed from the manifold learnt by our approach are closer to the training images in terms of smaller MSE and standard deviation  $\sigma$ . Our approach reduces the MSE by 35% and the  $\sigma$  by 31% for the flag sequence, and 30% and 26% for the wave sequence.

Sample synthesized frames from dynamic texture sequences are shown in Figure 5. The images synthesized by our approach are much crisper than those obtained by DGCM, especially when there is a sudden change of dynamics. Subtle details like the folds of the flag and foam on the wave are crisper in the images synthesized by our approach. This observation is consistent with the evaluation with the synthetic data, where our approach is able to handle sudden changes of motion through switching among multiple dynamical models.



Fig. 5. Comparison of texture synthesis results for the flag and wave sequences: **a.** DGCM, and **b.** our approach. The folds of the flag and the foam of the wave are crisper than those produced by DGCM.

TABLE VII

EXPERIMENTAL SETUP FOR HUMAN MOTION ANALYSIS (SECTION V-C). IN SLDS, NUMBER OF STATES REFER TO THE NUMBER OF DYNAMICAL MODELS USED.

	SLDS	DGCM	Our Approach
Len. of the Mocap Sequence	1500	1500	1500
Dim. of $\mathbf{x}$	28	28	28
Dim. of $\mathbf{g}$	-	3	3
Num. of States	17	12	12
Training Time	~ 45 min	~ 28 min	~ 33 min

### C. Human Motion Synthesis and Analysis

We test our approach on the tasks of human motion synthesis, classification and tracking to demonstrate the advantages of modeling dynamics on the low-dimensional manifold with multiple linear dynamical models. The *Boxing* sequences of S1 from the benchmark datasets [5] are used. In all three experiments, the motion capture sequence (containing multiple cycles of the boxing action) from Session 3 is used to train the model. Table VII summarizes the experimental setup.

1) *Human Motion Synthesis*: We compare our approach with DGCM for the task of reconstructing human body configurations from the learnt low-dimensional manifolds; *i.e.*, we compute the average joint angle error between the training data and the reconstructed data by applying  $f_{\mathbf{g} \rightarrow \mathbf{x}}(f_{\mathbf{x} \rightarrow \mathbf{g}})$  on the training data. Figure 6 shows the average reconstruction error of each joint

location of the upper body limbs over all the training frames. The errors reported on the left and right upperbody joints are not exactly the same due to asymmetrical limb movements. We can see that DGCM tends to make more errors at the shoulders. This can cause large errors for the joints at the elbows when we convert from the joint angle representation to the actual 3D human body joint positions. The errors made by our approach at the shoulder joints are at least one standard deviation smaller than those made by DGCM. This is because our motion model is able to capture the nonlinear limb movements more effectively in the latent space.

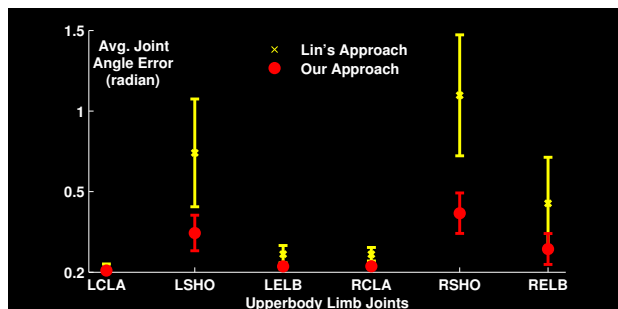


Fig. 6. Comparison of reconstruction error. Our approach has smaller reconstruction error (both mean and standard deviation) than DGCM (Lin’s approach), especially at the joints higher on the hierarchy of the kinematic chain. The short form joint labels are: LCLA (left clavicle), LSHO (left shoulder), LELB (left elbow) and RCLA, RSHO and RELB refer to the corresponding joints on the right upper body limbs

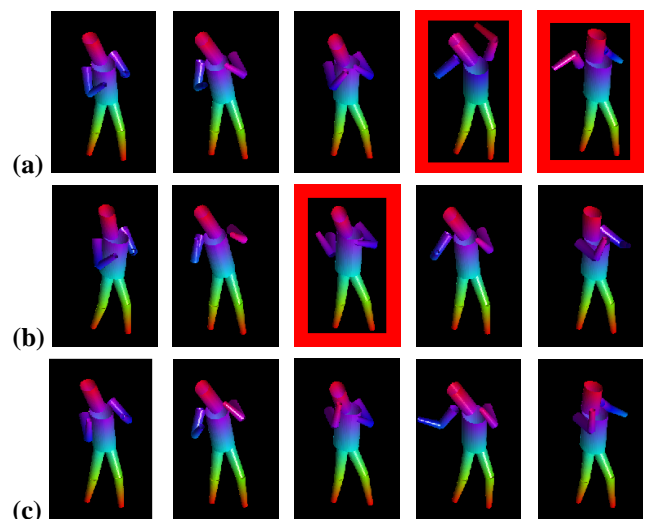


Fig. 7. Comparison of synthesis results. Row (a) shows synthesized frames using SLDS, row (b) DGCM and row (c) is our approach. Undesirable synthesized results are shown with red border.

We apply the learnt  $f_{dyn}$  and  $f_{g \rightarrow x}$  by DGCM and our approach to synthesize 100 frames. We also use SLDS [3] as a motion model to synthesize 100 frames. Sample synthesized frames are shown in Figure 7. The undesirable synthesis results are shown in red border and they are produced by SLDS and DGCM. We can see that the propagated error at the shoulder joints introduces unnatural configurations of the lower arms. In comparison, our approach is able to produce more natural boxing actions when compared to [3], [2] thanks to the temporally consistent learning of the low-dimensional manifold and effective modeling of nonlinear dynamics using interacting linear models.

2) *Human Motion Classification*: As we approximate  $f_{dyn}$  with multiple linear motion models, we can do motion classification when we associate each model with a class label. This experiment with the boxing sequence demonstrates such classification capability. The test sequence comprises 300 frames in this experiment. We compare our model with the SLDS model proposed in [3]. Note that in the SLDS model, the observation and hidden states of the continuous layer are of the same dimensionality (28), while the hidden states of the continuous layer in our model are of much lower dimension (3 in the current setup). In our approach, the 6 states for the forward punch are considered as one class and the 6 states for the upward punch are considered as another class. Similarly, for the 17 states used for the SLDS model, the 7 states being labeled as forward punch are considered as one class and the other 10 are considered the upward punch class. SLDS state labels are set to maximize the classification accuracy.

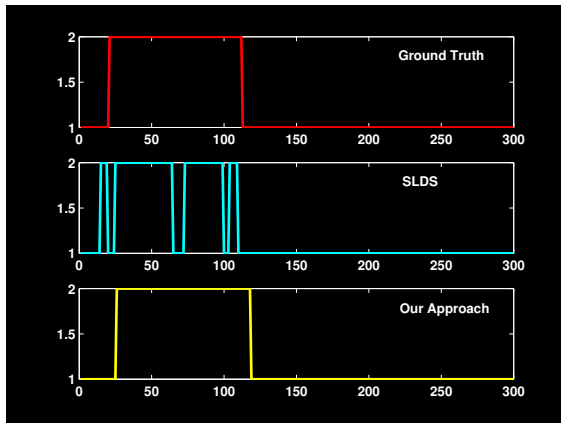


Fig. 8. Comparison of classification results. The horizontal axis shows the frame indices, while the vertical axis shows class labels: 1 refers to upward punch and 2 refers to forward punch. Our approach produces accurate classification results, whereas there is an abrupt change of class labels in SLDS.

As shown in Figure 8, our approach achieves 95% classification accuracy. At the state transition, our approach tends to delay the transition a little bit more for about 5-10 frames. This can be explained by the global coordination mechanism which counteracts the abrupt switching. As there is no such mechanism in the SLDS model and the high-dimensional states are less discriminative, SLDS tends to switch among different classes more frequently and hence has a lower classification accuracy of 90.3% for this data set.

3) *Human Motion Tracking*: In this experiment, we use the learnt  $f_{dyn}$  and  $f_{g \rightarrow x}$  to provide prior information for 3D human motion tracking. The tracker we use here is similar to [13], [8]. We test the Boxing sequences from Sessions 1 and 2 of S1 and evaluate the tracker accuracy

TABLE VIII

COMPARISON OF TRACKER ERRORS AND PROCESSING TIME PER FRAME. OUR APPROACH TAKES SLIGHTLY MORE TIME PER FRAME COMPARED TO DGCM WITH AN IMPROVED ACCURACY OF 50% BOTH IN TERMS OF MEAN AND STANDARD DEVIATION OF THE MARKER ERROR.

	SLDS	DGCM	Our Approach
Mean marker error (mm)	569.90	380.02	187.50
$\sigma$ (mm)	209.18	74.97	39.73
Processing time per frame (second)	$\sim 120$	$\sim 32$	$\sim 41$

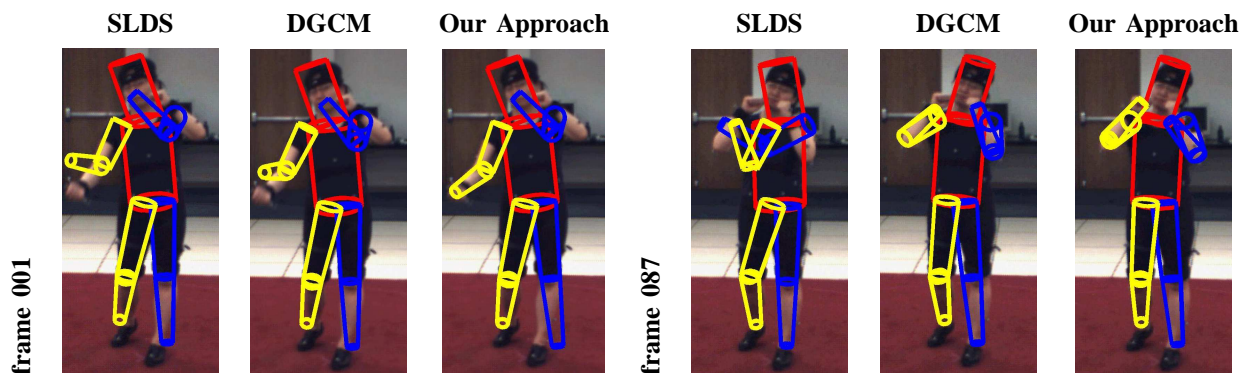


Fig. 9. Sample tracked frames. Both SLDS and DGCM fail to lock on the right lower arm in frame 1. SLDS fails to track both arms in frame 87.

from the online evaluation tool provided by [5]. The tracker errors reported in Table VIII are computed based on the criteria defined in [5]. From Table VIII, we can see that the mean error for recovered virtual joint marker positions (see [5] for detail) is within 250 mm which is less than half the error reported for SLDS and DGCM. Sample tracked frames are shown in Figure 9. We can see that the tracker that uses the priors from our approach is able to lock on to the limbs over time while the other two approaches fail. Our tracker is also able to generalize fairly well for motion with slight variation from the training data as the training sequence and testing sequences are captured at different times with the same test subject. The promising results show that the proposed model can be used effectively in tracking applications.

TABLE IX

MODEL PARAMETER PRIORS.  $\mathcal{N}(\cdot, \cdot)$  STANDS FOR NORMAL DISTRIBUTION;  $\mathcal{G}(\cdot, \cdot)$  REFERS TO GAMMA DISTRIBUTION AND  $\mathcal{D}(\cdot)$  REPRESENTS DIRICHLET DISTRIBUTION. WE EXCLUDE THE PRIOR DISTRIBUTION OF THE OBSERVATION NOISE,  $\mathbf{R}$ , SINCE IT DOES NOT GROW WITH THE MODEL COMPLEXITY. THE PARAMETERS IN DIFFERENT COMPONENTS ARE

$$\text{INDEPENDENT, e.g., } p(\mathbf{F}|\beta_{\mathbf{F}}^*) = \prod_{s=1}^S p(\mathbf{F}^{(s)}|\beta_{\mathbf{F}^{(s)}}^*).$$

Prior	Distribution
$p(\mathbf{F}^{(s)} \beta_{\mathbf{F}^{(s)}}^*)$	$([\mathbf{F}^{(s)}]_{i\cdot})^\top \sim \mathcal{N}(\mathbf{0}, [\text{diag}(\beta_{\mathbf{F}^{(s)}}^*)]^{-1}), \forall i \in \{1, \dots, d\}$
$p(\boldsymbol{\kappa}^{(s)} \mu_{\boldsymbol{\kappa}^{(s)}}^*, \beta_{\boldsymbol{\kappa}^{(s)}}^*)$	$[\boldsymbol{\kappa}^{(s)}]_i \sim \mathcal{N}([\mu_{\boldsymbol{\kappa}^{(s)}}^*]_i, [\beta_{\boldsymbol{\kappa}^{(s)}}^*]_i^{-1}), \forall i \in \{1, \dots, d\}$
$p(\Sigma^{(s)} a_{\Sigma^{(s)}}^*, \mathbf{b}_{\Sigma^{(s)}}^*)$	$\beta^{(s)} \sim \mathcal{G}(a_{\Sigma^{(s)}}^*, \mathbf{b}_{\Sigma^{(s)}}^*), \text{ where } \Sigma^{(s)} = [\text{diag}(\beta^{(s)})]^{-1}$
$p(\Lambda^{(s)} \rho_{\Lambda^{(s)}}^*, \beta_{\Lambda^{(s)}}^*)$	$([\Lambda^{(s)}]_{i\cdot}^\top) \sim \mathcal{N}(\mathbf{0}, [\rho_{\Lambda^{(s)}}^*]^{-1}[\text{diag}(\beta_{\Lambda^{(s)}}^*)]^{-1}), \forall i \in \{1, \dots, D\}$
$p(\rho_{\Lambda^{(s)}} a^*, \mathbf{b}^*)$	$[\rho_{\Lambda^{(s)}}]_i \sim \mathcal{G}(a^*, [\mathbf{b}^*]_i), \forall i \in \{1, \dots, d\}$
$p(\mu^{(s)} \mu^*, \beta^*)$	$\mu^{(s)} \sim \mathcal{N}(\mu^*, [\text{diag}(\beta^*)]^{-1})$
$p(\boldsymbol{\pi} \mathbf{u}_{\boldsymbol{\pi}}^*)$	$\boldsymbol{\pi} \sim \mathcal{D}(\mathbf{u}_{\boldsymbol{\pi}}^*), \text{ where } \mathbf{u}_{\boldsymbol{\pi}}^* = [(\mathbf{u}_{\boldsymbol{\pi}}^*)^1, \dots, (\mathbf{u}_{\boldsymbol{\pi}}^*)^{(S)}]$
$p(\mathbf{A} \mathbf{u}_{\mathbf{A}}^*)$	$[\mathbf{A}]_s \sim \mathcal{D}(\mathbf{u}_{\mathbf{A}}^*), \text{ where } \mathbf{u}_{\mathbf{A}}^* = [(\mathbf{u}_{\mathbf{A}}^*)^1, \dots, (\mathbf{u}_{\mathbf{A}}^*)^{(S)}]$

## VI. VARIATIONAL BAYESIAN FORMULATION

So far we have proposed a general method for efficient simultaneous learning of a nonlinear low-dimensional manifold and a nonlinear dynamical model for high-dimensional time series. The proposed solution exploits the coordinated piecewise linear models to overcome these difficulties. Our experiments verify the efficiency and effectiveness of the proposed solution. However, the number of states is chosen independently of the dynamical models using a variational Bayesian approach. The dimensionality of the latent space is chosen empirically. In this section, we will derive a full-fledged variational Bayesian formulation to choose the optimal model setup, *i.e.*, the number of components and the dimensionality of the latent space.

### A. Model priors

To derive the VB formulation for our proposed model, we first define the prior distribution of the model parameters in Table IX.

Given an observation sequence  $\mathbf{x}_{1:T}$ , an exact Bayesian treatment of our proposed model would require computing the marginals of the posterior over the model parameters,  $\boldsymbol{\theta}$  and the hidden variables  $\mathbf{s}_{1:T}$  and  $\mathbf{g}_{1:T}$ . The coupling of the model parameters and the hidden variables makes the integration over them analytically intractable. However, since our proposed model belongs

to the conjugate exponential family, we can apply the findings in [40] to derive a variational Bayesian Expectation Maximization (VBEM) learning algorithm. The set of precision parameters  $\rho_{\Lambda^{(s)}}$  defined on the covariance matrices of the factor loading matrices  $\Lambda^{(s)}$  is used to infer the dimensionality of the latent space through automatic relevance determination (ARD). The intuition is that a particular dimension is “turned off” if the value of the corresponding entry in  $\rho_{\Lambda^{(s)}}$  becomes very large.

### B. Derivation

Based on the VB learning theory introduced in [32], we can obtain the VB lower bound by introducing variational distribution  $q(\cdot)$  over the model parameters and hidden variables and applying Jensen’s inequality as follows:

$$\begin{aligned}
\ln p(\mathbf{x}_{1:T}) &= \ln \int d\boldsymbol{\theta} \sum_{\mathbf{s}_{1:T}} \int d\mathbf{g}_{1:T} p(\mathbf{x}_{1:T}, \mathbf{g}_{1:T}, \mathbf{s}_{1:T} | \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\
&\geq \int d\boldsymbol{\theta} \sum_{\mathbf{s}_{1:T}} \int d\mathbf{g}_{1:T} q(\boldsymbol{\theta}, \mathbf{s}_{1:T}, \mathbf{g}_{1:T}) \ln \frac{p(\mathbf{x}_{1:T}, \mathbf{g}_{1:T}, \mathbf{s}_{1:T} | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}, \mathbf{s}_{1:T}, \mathbf{g}_{1:T})} \\
&\triangleq \int d\boldsymbol{\theta} q(\boldsymbol{\theta}) \sum_{\mathbf{s}_{1:T}} q(\mathbf{s}_{1:T}) \int d\mathbf{g}_{1:T} q(\mathbf{g}_{1:T}) \ln \frac{p(\mathbf{x}_{1:T}, \mathbf{g}_{1:T}, \mathbf{s}_{1:T} | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}) q(\mathbf{s}_{1:T}) q(\mathbf{g}_{1:T})} \\
&= \mathcal{F}^{(VB)}. \tag{11}
\end{aligned}$$

The only assumption we make during the derivation is that the variational distribution over model parameters and hidden variables  $q(\boldsymbol{\theta}, \mathbf{s}_{1:T}, \mathbf{g}_{1:T})$  can be factorized as  $q(\boldsymbol{\theta}, \mathbf{s}_{1:T}, \mathbf{g}_{1:T}) \triangleq q(\boldsymbol{\theta}) q(\mathbf{s}_{1:T}) q(\mathbf{g}_{1:T})$ .

1) *Deriving the lower bound  $\mathcal{F}^{(VB)}$* : Based on the dependence among the model parameters, the VB lower bound from (11) can be rewritten as follows:

$$\begin{aligned}
\mathcal{F}^{(VB)} &= -KL(q(\boldsymbol{\pi}) || p(\boldsymbol{\pi})) - KL(q(\mathbf{A}) || p(\mathbf{A})) - KL(q(\mathbf{F}) || p(\mathbf{F})) - KL(q(\boldsymbol{\mu}) || p(\boldsymbol{\mu})) \\
&\quad - KL(q(\boldsymbol{\beta}) || p(\boldsymbol{\beta})) + \int d\boldsymbol{\beta} q(\boldsymbol{\beta}) \int d\boldsymbol{\Lambda} q(\boldsymbol{\Lambda} | \boldsymbol{\beta}) \ln \frac{p(\boldsymbol{\Lambda} | \boldsymbol{\beta})}{q(\boldsymbol{\Lambda} | \boldsymbol{\beta})} \\
&\quad - KL(q(\boldsymbol{\Sigma}) || p(\boldsymbol{\Sigma})) + \int d\boldsymbol{\Sigma} q(\boldsymbol{\Sigma}) \int d\boldsymbol{\kappa} q(\boldsymbol{\kappa} | \boldsymbol{\Sigma}) \ln \frac{p(\boldsymbol{\kappa} | \boldsymbol{\Sigma})}{q(\boldsymbol{\kappa} | \boldsymbol{\Sigma})} \\
&\quad + \left\langle \int d\boldsymbol{\theta} q(\boldsymbol{\theta}) \ln p(\mathbf{x}_{1:T}, \mathbf{g}_{1:T}, \mathbf{s}_{1:T} | \boldsymbol{\theta}) - \ln q(\mathbf{s}_{1:T}) - \ln q(\mathbf{g}_{1:T}) \right\rangle_{q(\mathbf{s}_{1:T}) q(\mathbf{g}_{1:T})} \tag{12}
\end{aligned}$$

The iterative variational Bayesian Expectation Maximization (VBEM) algorithm used to optimize  $\mathcal{F}^{(VB)}$  is summarized in Algorithm 3:

**Algorithm 3** Variational Bayesian learning algorithm for our proposed model

- 
- 1: Initialize parameters. Initialize hidden variables and state priors.
  - 2: **for**  $n_1=1:\text{max\_iter}_1$  **do**
  - 3:   **VBM Step:**
  - 4:     Compute the expected natural parameters of  $q(\theta)$ .
  - 5:   **VBE Step:**
  - 6:     Compute the sufficient statistics of hidden variable distributions  $q(\mathbf{s}_{1:T})$  and  $q(\mathbf{g}_{1:T})$ .
  - 7:   **Optimize hyperparameters.**
  - 8: **end for**
- 

2) *The VBM step:* The VBM step is obtained by taking functional derivatives of  $\mathcal{F}^{(VB)}$  with respect to the variational distributions of model parameters and equating them to zero. As the derivations are mainly algebraic manipulation, we refer the readers to [45] for detailed derivations to maintain the clarity of this paper. The main update equations are summarized as follows:

- The expected values of initial state distribution are computed:

$$\tilde{\boldsymbol{\pi}} = \exp \left[ \psi \left( \mathbf{u}_{\boldsymbol{\pi}}^{(s)} \right) - \psi \left( \sum_{s=1}^S \mathbf{u}_{\boldsymbol{\pi}}^{(s)} \right) \right], \quad (13)$$

where  $\mathbf{u}_{\boldsymbol{\pi}}^{(s)} = (\mathbf{u}_{\boldsymbol{\pi}}^*)^{(s)} + \langle \mathbf{s}_1^{(s)} \rangle_{q(\mathbf{s}_{1:T})}$  and  $\psi$  is the digamma function.

- The expected values of switching state transition distribution are:

$$\tilde{\mathbf{A}} = \exp \left[ \psi \left( \mathbf{u}_{\mathbf{A}}^{ss'} \right) - \psi \left( \sum_{s=1}^S \mathbf{u}_{\mathbf{A}}^{ss'} \right) \right], \quad (14)$$

where  $\mathbf{u}_{\mathbf{A}}^{ss'} = (\mathbf{u}_{\mathbf{A}}^*)^{(s)} + \langle \mathbf{s}_{t-1}^{s'} \mathbf{s}_t^{(s)} \rangle_{q(\mathbf{s}_{1:T})}$ .

- The distribution of  $\boldsymbol{\Sigma}^{(s)}$  where  $\boldsymbol{\Sigma}^{(s)} = [\text{diag}(\boldsymbol{\beta}^{(s)})]^{-1}$  follows a Gamma distribution as:

$\boldsymbol{\beta}^{(s)} \sim \mathcal{G} \left( a_{\boldsymbol{\Sigma}^{(s)}}^*, \mathbf{b}_{\boldsymbol{\Sigma}^{(s)}}^* \right)$ , with the updates for  $a_{\boldsymbol{\Sigma}^{(s)}}$  and  $\mathbf{b}_{\boldsymbol{\Sigma}^{(s)}}^*$  as following:

$$a_{\boldsymbol{\Sigma}^{(s)}} = a_{\boldsymbol{\Sigma}^{(s)}}^* + \frac{1}{2} \bar{N}^{(s)}, \quad [\mathbf{b}_{\boldsymbol{\Sigma}^{(s)}}]_i = [\mathbf{b}_{\boldsymbol{\Sigma}^{(s)}}^*]_i + \frac{1}{2} \left( [\bar{\mathbf{c}}^{(s)}]_i + \frac{\bar{N}^{(s)} * [\boldsymbol{\Sigma}^{(s)}]_{ii}^{-1}}{\bar{N}^{(s)} + [\boldsymbol{\Sigma}^{(s)}]_{ii}^{-1}} ([\bar{\mathbf{g}}^{(s)}]_i - [\boldsymbol{\mu}_{\boldsymbol{\kappa}^{(s)}}]_i)^2 \right), \quad (15)$$

where

$$\bar{N}^{(s)} = \sum_{t=1}^T \langle s_t^{(s)} \rangle, \quad \bar{\mathbf{g}}^{(s)} = \frac{1}{\bar{N}^{(s)}} \sum_{t=1}^T \langle s_t^{(s)} \rangle \langle \mathbf{g}_t \rangle, \quad [\bar{\mathbf{c}}^{(s)}]_i = \sum_{t=1}^T \langle s_t^{(s)} \rangle ([\langle \mathbf{g}_t \rangle]_i - [\bar{\mathbf{g}}^{(s)}]_i)^2,$$

and  $i = 1, \dots, d_{max}$ .

- The distribution of  $\boldsymbol{\kappa}^{(s)}$  follows a Gaussian distributions as:  $\boldsymbol{\kappa}^s \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\kappa}^{(s)}}, \boldsymbol{\Sigma}_{\boldsymbol{\kappa}^{(s)}})$ , where the expected sufficient statistics are:

$$\begin{aligned} [\boldsymbol{\Sigma}_{\boldsymbol{\kappa}^{(s)}}]_{ii}^{-1} &= [\boldsymbol{\beta}_{\boldsymbol{\kappa}^{(s)}}^*]_i + \sum_{t=1}^T [\boldsymbol{\Sigma}^{(s)}]_{ii}^{-1} \langle s_t^{(s)} \rangle, \\ [\boldsymbol{\mu}_{\boldsymbol{\kappa}^{(s)}}]_i &= [\boldsymbol{\Sigma}_{\boldsymbol{\kappa}^{(s)}}]_{ii} \left( \sum_{t=1}^T \langle s_t^{(s)} \rangle [\boldsymbol{\Sigma}^{(s)}]_{ii}^{-1} [\langle \mathbf{g}_t \rangle]_i + [\boldsymbol{\beta}_{\boldsymbol{\kappa}^{(s)}}^*]_i [\boldsymbol{\mu}_{\boldsymbol{\kappa}^{(s)}}^*]_i \right). \end{aligned} \quad (16)$$

- The dynamic matrix or the state evolution matrix  $\mathbf{F}^{(s)}$  for component  $s$  is normally distributed:  $[\mathbf{F}^s]_i \sim \mathcal{N}(\boldsymbol{\mu}_{[\mathbf{F}^s]_i}, \boldsymbol{\Sigma}_{[\mathbf{F}^s]_i})$  where

$$\begin{aligned} \boldsymbol{\mu}_{[\mathbf{F}^s]_i} &= \mathbf{S}_{[\mathbf{F}^{(s)}]_i} \boldsymbol{\Sigma}_{[\mathbf{F}^{(s)}]_i}^{-1} \sum_{t=1}^T \langle s_t^{(s)} \rangle [\langle \mathbf{g}_{t-1} \mathbf{g}_t^T \rangle]_{ii}, \\ [\boldsymbol{\Sigma}_{\mathbf{F}^{(s)}}]^{-1} &= [\text{diag}(\boldsymbol{\beta}_{\mathbf{F}^{(s)}})]^{-1} + \sum_{t=2}^T \langle \mathbf{g}_{t-1} \mathbf{g}_{t-1}^T \rangle. \end{aligned} \quad (17)$$

- The precision  $\boldsymbol{\rho}_{\boldsymbol{\Lambda}^{(s)}}$  for each row of factor loading matrix  $\boldsymbol{\Lambda}^{(s)}$  is Gamma-distributed:  $[\boldsymbol{\rho}_{\boldsymbol{\Lambda}^{(s)}}] \sim \mathcal{G}(a_{\boldsymbol{\Lambda}^{(s)}}, \mathbf{b}_{\boldsymbol{\Lambda}^{(s)}})$ , with

$$a_{\boldsymbol{\Lambda}^{(s)}} = a^* + \frac{1}{2} \bar{N}^{(s)}, \quad [\mathbf{b}_{\boldsymbol{\Lambda}^{(s)}}]_i = [\mathbf{b}^*]_i + \frac{1}{2} \sum_{j=1}^D \langle [\boldsymbol{\Lambda}^{(s)}]_{ji}^2 \rangle_{q(\boldsymbol{\Lambda}^{(s)})}, \quad (18)$$

and the expected value of  $[\boldsymbol{\rho}_{\boldsymbol{\Lambda}^{(s)}}]_i$  being:  $\langle [\boldsymbol{\rho}_{\boldsymbol{\Lambda}^{(s)}}]_i \rangle = [\bar{\boldsymbol{\rho}}_{\boldsymbol{\Lambda}^{(s)}}]_i = \frac{a_{\boldsymbol{\Lambda}^{(s)}}}{[\mathbf{b}_{\boldsymbol{\Lambda}^{(s)}}]_i}$ .

- The  $j$ -th row of factor loading/measurement matrix  $\boldsymbol{\Lambda}^{(s)}$  for a given precision matrix  $[\boldsymbol{\rho}_{\boldsymbol{\Lambda}^{(s)}}]_j$ ,  $[\boldsymbol{\Lambda}^{(s)}]_j^T$  follows a Gaussian distribution with the following updates:

$$\begin{aligned} \boldsymbol{\mu}_{[\boldsymbol{\Lambda}^{(s)}]_j} &= \mathbf{S}_{[\boldsymbol{\Lambda}^{(s)}]_j} \boldsymbol{\Sigma}_{[\boldsymbol{\Lambda}^{(s)}]_j}^{-1} \sum_{t=1}^T \langle s_t^{(s)} \rangle \mathbf{x}_t \langle \mathbf{g}_t^T \rangle \Big|_j, \\ [\boldsymbol{\Sigma}_{[\boldsymbol{\Lambda}^{(s)}]_j}]^{-1} &= \text{diag}(\boldsymbol{\beta}_{\boldsymbol{\Lambda}^{(s)}}^*) + \sum_{t=1}^T \langle s_t^{(s)} \rangle \langle \mathbf{g}_t \mathbf{g}_t^T \rangle. \end{aligned} \quad (19)$$

- The updates for the prior over the centers  $\{\boldsymbol{\mu}^s\}_{s=1}^S$  of each component are:

$$\begin{aligned} [\boldsymbol{\Sigma}_{\boldsymbol{\mu}^{(s)}}]_{jj}^{-1} &= [\boldsymbol{\beta}^*]_j + [\boldsymbol{\Psi}]_{ii}^{-1} \sum_{t=1}^T s_t^{(s)}, \\ \boldsymbol{\mu}_j^{(s)} &= [\boldsymbol{\Sigma}_{\boldsymbol{\mu}_s}]_{jj} \left( [\boldsymbol{\Psi}]_{jj}^{-1} \sum_{t=1}^T \langle s_t^{(s)} \rangle [\mathbf{x}_t]_j + [\boldsymbol{\beta}^*]_j [\boldsymbol{\mu}^*]_j \right), \quad j \in \{1, \dots, D\}. \end{aligned} \quad (20)$$

3) *The VBE step*: Based on the theoretical results from [40], we propose to further use structural approximation of our proposed model to make the inference, the VBE step, tractable. Given the sufficient statistics and expected distributions of the model parameters we computed from the VBM step, we can reuse the inference algorithm derived for the variational maximum-likelihood (ML) learning algorithm (Algorithm 2). To make this section self-contained, we briefly summarize the inference algorithm for the VBE step here:

---

**Algorithm 4** Inference algorithm (VBE step)

---

```

1: for  $n_2=1:\text{max\_iter}_2$  do
2:   Do inference on the HMM submodel:
3:     Compute  $\langle \mathbf{s}_t \rangle$  using the expected distributions and sufficient statistics of the
       model parameters and  $\mathbf{g}_t$ .
4:   Do inference on the LDS submodel:
5:     Run the variational Kalman smoother [40] using the expected distributions and
       sufficient statistics of the model parameters and  $\mathbf{s}_t$ .
6: end for

```

---

4) *Hyperparameter optimization*: The hyperparameters for our proposed model are:

$$\Theta = \{\beta_{\mathbf{F}^{(s)}}^*, \mu_{\mathbf{K}^{(s)}}^*, \beta_{\mathbf{K}^{(s)}}^*, a_{\Sigma^{(s)}}^*, \mathbf{b}_{\Sigma^{(s)}}^*, \beta_{\Lambda^{(s)}}^*, a^*, \mathbf{b}^*, \mu^*, \beta^*, \mathbf{u}_{\pi}^*, \mathbf{u}_{\mathbf{A}}^*, \Psi\},$$

and  $\Theta$  can be optimized by taking derivatives of  $\mathcal{F}^{(VB)}$  with respect to the individual hyperparameter. The updates for  $\mu_{\mathbf{K}^{(s)}}^*, \beta_{\mathbf{K}^{(s)}}^*, \beta_{\Lambda^{(s)}}^*, a^*, \mathbf{b}^*, \mu^*, \beta^*, \mathbf{u}_{\pi}^*$  and  $\Psi$  are the same as the derivations in [45] for the GCMFA model. And the updates for  $\mathbf{u}_{\mathbf{A}}^*$  are the same as the updates derived for HMM in [49]. The only extra set of updates is  $\beta_{\mathbf{F}^{(s)}}^*$  where

$$[\beta_{\mathbf{F}^{(s)}}^*]_i^{-1} = \text{diag} \left( \Sigma_{\mathbf{F}^{(s)}} \right) + \frac{1}{d} \left[ \Sigma_{\mathbf{F}^{(s)}} \mathbf{S}_{\mathbf{F}^{(s)}} \mathbf{S}_{\mathbf{F}^{(s)}}^T \Sigma_{\mathbf{F}^{(s)}} \right]_{ii}. \quad (21)$$

5) *Model Splitting*: We extend the splitting strategy proposed in [50] and start with a single FA with a single LDS. We let it split and keep track of the value of  $\mathcal{F}^{(VB)}$ . We conduct the splitting at the HMM submodel level. All possible splittings are attempted and the one that gives the best improvement of  $\mathcal{F}^{(VB)}$  is chosen. This is similar to the Variational Bayesian learning framework proposed in [49]. There are other state-splitting strategies for HMMs [51], [52]; however, we choose the splitting strategy adopted in [50] because it takes temporal information into consideration during splitting and it gives that best overall results

## VII. EXPERIMENTS WITH THE MODELS LEARNED VIA THE VB ALGORITHM

In this section, we report the results of three sets of experiments that evaluate our VB formulation, wherein the model structure is automatically selected during the VB training. The first set of experiments involves the modeling and synthesis of dynamic textures. The second set of experiments evaluates our VB formulation in 3D human motion tracking from 2D video sequences. In these first two sets of experiments, the results attained using the VB formulation (where the model parameters and structure are automatically determined) compare favorably with those attained using the ML formulation of Section IV (where the model parameters and structure are hand-tuned to yield best possible performance). In the final experiment, the model learned via the VB algorithm is demonstrated in labeling the actions in human motion sequence data. This final experiment demonstrates how actions can be associated with specific sets of states in the dynamical process model.

### A. Dynamic Texture

TABLE X

EXPERIMENTAL SETUP OF EXPERIMENTS (SECTION V-B) WITH DYNAMIC TEXTURE. WE USE “ML APPROACH” TO REFER TO THE VARIATIONAL ML LEARNING ALGORITHM PROPOSED IN ALGORITHM 1. THE DIMENSIONALITY OF  $\mathbf{g}$  AND THE NUMBER OF STATES ARE CHOSEN BY THE ALGORITHM FOR THE VB APPROACH WHILE THE SAME QUANTITIES ARE CHOSEN EMPIRICALLY FOR THE ML APPROACH.

	ML approach	VB approach
Len. of the Flag Sequence	250	250
Len. of the Wave Sequence	350	350
Dim. of $\mathbf{x}$	104256(= 288 × 362)	104256
Dim. of $\mathbf{g}$	20	Flag: 18 Wave: 19
Num. of States	3	Flag: 3 Wave: 2
Training Time	~ 8 min	~ 60 min

TABLE XI

COMPARISON OF RECONSTRUCTION ERROR FROM TRAINING FRAMES.

	Flag Sequence		Wave Sequence	
	Mean Err.	$\sigma$	Mean Err.	$\sigma$
ML approach	0.0161	0.0258	0.0210	0.0235
VB approach	0.0177	0.0252	0.0184	0.0193

We use the same texture datasets used in Section V-B as the input training data for the VB

learning algorithm. We then synthesize textures from the trained models. Table X shows the setup of the experiments. Instead of choosing the number of states and the dimensionality of the latent space empirically as was done with the ML approach in Section V, the VB approach is able to determine the model structure during learning.

We adopt the same measure used in Section V-B to quantify the information loss. It computes the MSE of normalized intensity values (range from 0 to 1) between the training video frames and the reconstructed frames. The statistics of the reconstruction error are shown in Table XI. From the reported error statistics, we can see that the images reconstructed from the models learned by the VB approach have comparable mean errors and standard deviations. However, the models with the empirically chosen structure have a higher complexity both in terms of the dimensionality  $g$  and the number of states. In comparison, the models obtain via the VB approach have a relatively simpler structure. In the flag sequence, the VB approach increases the MSE by 10% and reduces standard deviation  $\sigma$  by 2.3% compared to these of the ML approach. In the wave sequence, the VB approach reduces both the MSE and the  $\sigma$  by 12.4% and 17.8%.

In Figure 10, there is no detectable difference of the visual qualities of the synthesized frames for both the flag sequence and the wave sequence.

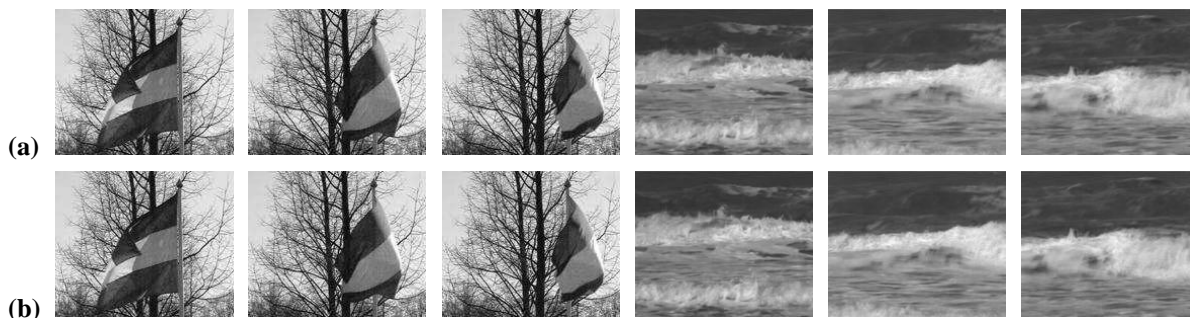


Fig. 10. Comparison of texture synthesis results for the flag sequence and wave sequence. The frames shown in the first row (a) are obtained from the model learned using the ML approach; and the frames shown in the second row (b) are obtained from the model learned using the VB approach.

As demonstrated in the experiments with dynamic texture, we can see that the models obtained from the VB approach with the optimal model structure determined by the learning algorithm can achieve comparable performance to the models obtained from the ML approach. However, in the ML approach, great care needs to be taken to choose the model structure; oftentimes a

lot of experiments are needed in order to find a good model structure and we have no way to gauge whether we are overfitting the data. The VB approach provides a principled solution to choose the optimal model structure during the learning of model parameters.

### B. Human Motion Tracking

In this set of experiments, we use the learned  $f_{dyn}$  and  $f_{\mathbf{g} \rightarrow \mathbf{x}}$  to propagate hypotheses and provide prior information for 3D human motion tracking. The tracker is the same as the tracker used in Section V-C.

We use the name ML-tracker to refer to the tracker that employs the model learned using the ML approach and VB-tracker to represent the tracker that uses the model learned via the VB approach. The training data is the motion capture sequence from Session 3 of S1. Table XII shows the setup of this experiment. Given that the image silhouettes are relatively noisy, we make use of videos from camera C1-C3.

TABLE XII

EXPERIMENTAL SETUP FOR 3D HUMAN MOTION TRACKING ON THE BOXING SEQUENCE. FOR ML-TRACKER, THE DIMENSIONALITY OF  $\mathbf{g}$  AND THE NUMBER OF STATES ARE CHOSEN EMPIRICALLY WHILE THE SAME QUANTITIES ARE OBTAINED TOGETHER WITH THE MODEL PARAMETERS FROM THE LEARNING STEP OF THE VB-TRACKER. THE LEARNED MODEL IS USED FOR TRACKING AND THE TRACKING ERROR IS REPORTED TOGETHER WITH THE MODEL PARAMETERS.

	ML-tracker	VB-tracker
Length of the Mocap Sequence	1500	1500
Dimensionality of $\mathbf{x}$	28	28
Dimensionality of $\mathbf{g}$	3	5
Number of States	12	6
Training Time	~ 33 min	~ 2.5 hrs
Mean marker error (mm)	152.31	110.95
$\sigma$ (mm)	53.86	44.07

We test both trackers on the boxing video sequences from Session 1 and 2 of S1 and evaluate the tracker accuracy from the online evaluation tool. The goal of this experiment is to compare the results obtained from the ML-tracker and the VB-tracker. Given that both are stochastic trackers, the mean and standard deviation of the tracker errors [5] are averaged over 20 trials

and 200 frames. The error statistics are shown in Table XII. Compared to ML-tracker, the VB-tracker on average has an improved accuracy of 27% in terms of mean marker error and 18% in terms of standard deviation of the marker error. Sample tracked frames are shown in Figure 11 and visually demonstrate the accuracy of the VB tracker as the lower limbs are tracked more reliably.

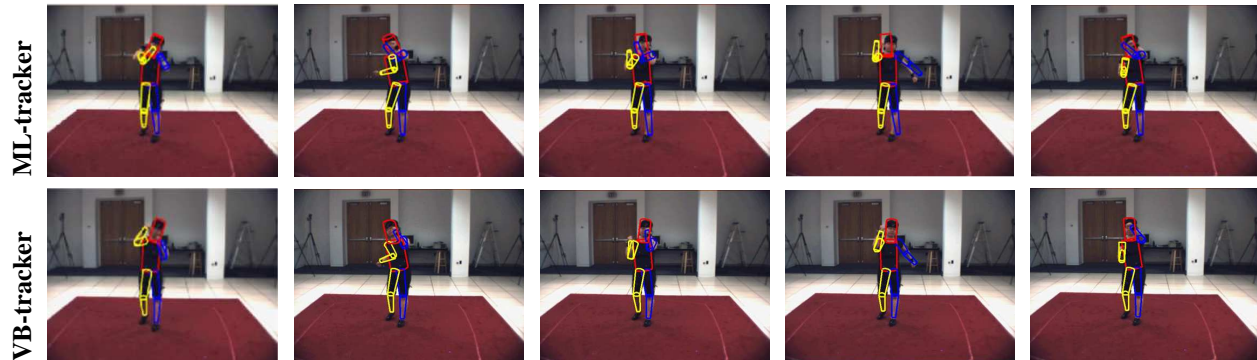


Fig. 11. Qualitative comparison base on the sample tracked frames from both the ML-tracker and the VB-tracker.

We also test our VB-tracker on more sequences (gestures, throw/catch, boxing) from Session 1 and 2 of subject S2. The number of states and the dimensionality of the latent space inferred by the VB learning algorithm from the training motion capture sequence (data from Session 3) are listed in Table XIII. We can see we need a more complex model in order to capture the leg movement involved in the boxing motion performed by S2. While in the boxing action performed by S1, the subject mostly stood still and only moved upper limbs. Given that the lower limbs do not have large movement and the upper limb motion is relatively smooth, *i.e.*, the upper limbs in these two motions do not move as fast as the boxing motion, simpler models are inferred for the gesture and the throw/catch motion by our VB learning algorithm.

Sample tracked frames are shown in Figure 12 and the tracker error statistics over multiple trials are reported in Table XIII. The error statistics reported here are comparable with the best recently published results in Poppe's pose estimation work [53]. Overall, our stochastic tracker produces comparable results to what have been reported in [53] on all the test sets. However, an exhaustive search of the (pose, image feature) pair database is used in [53] for pose estimation. Therefore, it maybe not be suitable for tracking if the database gets too large. Furthermore, given that it is not a stochastic algorithm, the results reported are for just one run of algorithm in [53]

TABLE XIII

INFERRED NUMBER OF STATES AND THE DIMENSIONALITY OF  $\mathbf{g}$  FOR MOTION CAPTURE DATA OF S2. THE LEARNED MODELS ARE USED FOR TRACKING AND THEIR CORRESPONDING VB-TRACKER ERRORS ON THE BOXING, GESTURE AND THROW/CATCH SEQUENCES OF S2 ARE REPORTED TOO.

	Boxing	Gesture	Throw/Catch
dimensionality of $\mathbf{g}$	8	4	4
number of states	7	4	3
Mean marker error (mm)	112.35	79.42	61.58
$\sigma$ (mm)	46.73	35.20	27.94

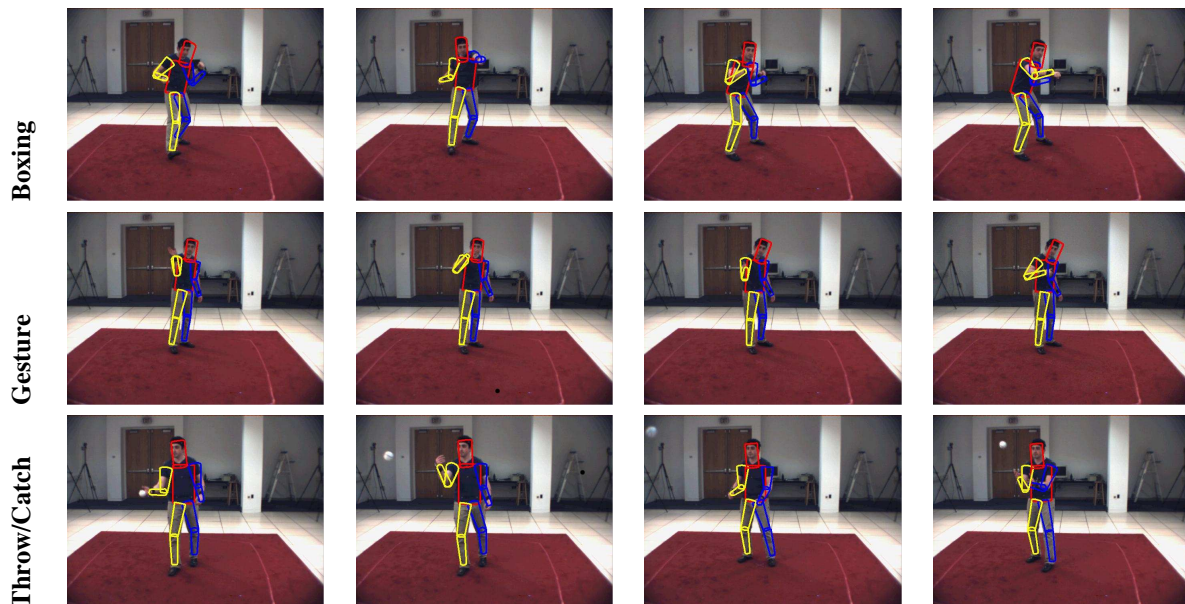


Fig. 12. Sample tracked frames from the VB-tracker.

while our error statistics are obtained via 20 runs of the tracking algorithm.

### C. Model Identification

In this experiment, we use our VB approach to obtain a model by training with a long sequence of human motion capture data. This long sequence is indexed as subject 13 and trial 30 from [54]. The reason we choose to use a capture sequence from a different database is that the long

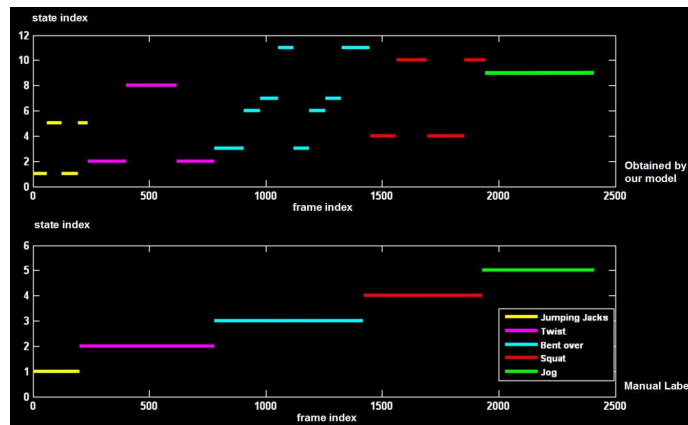


Fig. 13. Demonstration of labeling of the activities in a human motion sequence: jumping jacks, twist, bent-over, squat, jog. The bottom graph shows the ground truth labeling obtained through human labeling of the sequence. The horizontal axis of the graph depicts the time axis, and the vertical axis gives the activity label. The top graph shows the labeling obtained at each frame in the sequence using our model. The line segments in the graph represent dynamically similar regimes that can be described using a corresponding linear dynamical system. Except for jogging, all other actions need more than one linear dynamical model. The two segments in jumping jacks correspond to the upward movement of the arms and the downward movement of the arms. The three segments in the twist motion correspond to twist left, middle and twist right. The bent-over activity has the most number of segments due to complexity of the arm and leg movements. The four segments correspond to left elbow down and right knee up, left elbow up and right knee down, right elbow down and left knee up, right elbow up and left knee down. For jogging, the subject is performing stationary jogging with rather small limb movements. As a result, one component is sufficient to describe such motion.

sequences in the HumanEva-I datasets have many broken segments as the markers cannot be tracked reliably over a long time in their setup [5].

There are five different types of motions in this sequence from the CMU Mocap database: jumping jacks, side twists, bends over, squat and jog. The dimensionality of the training data is 56 and 2405 frames are used for training. Training takes about 6 hours to complete on the same 3.45 GHz PC with 4 GB memory. The dimensionality of the low-dimensional space is 7 and the number of components is 11. The dimensionality of the low-dimensional space and the number of components are inferred by applying the VB learning algorithm. Each component here refers to a locally linear submodel, which corresponds to a dynamically similar regime for the corresponding activities. We give each component a label and compare the labeled sequence with the manually labeled sequence in Figure 13.

In most actions, our model identifies the multiple segments for most of the activities. However,

with the last segment where the subject is jogging, only one component is identified by the VB solution. A closer look at the video reveals that in this segment, the subject is performing stationary jogging with rather small limb movements. As a result, one component is sufficient to describe such motion as demonstrated by the labeling output from our model.

This result shows that our VB learning algorithm is able to determine the number of linear components in the model and these linear components correspond to actions that can be captured by a linear dynamical model in the latent space. As a result, our VB learning algorithm can be used for the task of automatic segmentation of time sequences. And if we give semantic labels to the segments, further processing could lead to automatic annotation of the time sequences.

### VIII. CONCLUSIONS AND FUTURE WORK

The key message in this paper is that we should place equal importance on reducing the dimensionality and modeling dynamics of the high-dimensional time series. Most existing techniques mainly solve these modeling problems independently or by oversimplifying one of them. By placing equal importance on these two problems and solving them simultaneously in one coherent framework, we are able to model the high-dimensional time series with a few parameters from the trained model accurately. We have demonstrated the advantages of having such models in a variety of applications where the state-of-the-art accuracy and efficiency have been reported.

Nonetheless, our approach does have its limitations. One limitation with our approach is that we need to have strong knowledge of the model prior distributions in our VB solution so that suitable approximating variational distributions can be used. If inappropriate approximations are used, then the trained model could be useless as the lower bound would deviate too far from the true marginal likelihood. One way to discover the discrepancies introduced by the variational distributions is to do an analysis of the tightness of the lower bound as it indicates how much we are conceding when using such approximations. Similar to [32], a full analysis of the tightness of the VB bound could be done by sampling from our proposed model.

Another limitation is specific to the demonstration application. In the 3D human motion tracking application, it is possible that the observed actual body pose might deviate very far from the training poses. Consequently, none of the hypotheses generated from the trained model would be similar to the actual body pose we wish to estimate. One solution to this problem would be to incorporate a failure detection mechanism in our existing model. When the actual

body pose deviates very far from the training poses, the failure detection mechanism should signal to the tracker that the trained model cannot be used and more computationally expensive detection-based algorithms like [55], [56] should be activated to estimate the 3D body pose from the 2D images more reliably. Another possible solution to overcome this limitation is to use a similar framework by Urtasun, et al. [10]. In this work, the image likelihood computation is based on the inputs from a robust 2D tracker [57]. The idea is to increase the weight for the image likelihood term in the objective function so that the 3D pose obtained from the local optimization step respects the 2D image observations.

## REFERENCES

- [1] D. MacKay, “Bayesian interpolation,” *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.
- [2] R.-S. Lin, C.-B. Liu, M.-H. Yang, N. Ahuja, and S. Levinson, “Learning nonlinear manifolds from time series,” in *Proc. European Conf. on Computer Vision (ECCV)*, 2006, pp. 245–256.
- [3] V. Pavlovic, J. Rehg, and J. MacCormick, “Learning switching linear models of human motion,” in *Advances in Neural Information Processing Systems*. The MIT Press, 2000, pp. 981–987.
- [4] <http://www.cwi.nl/projects/dyntex/>.
- [5] L. Sigal and M. Black, “HumanEva: Synchronized video and motion capture dataset for evaluation of articulated human motion,” Brown University, Tech. Rep. CS-06-08, 2006.
- [6] A. Elgammal and C.-S. Lee, “Inferring 3D body pose from silhouettes using activity manifold learning,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 681–688.
- [7] K. Grochow, S. Martin, A. Hertzman, and Z. Popovic, “Style-based inverse kinematics,” *ACM Computer Graphics (SIGGRAPH)*, pp. 522–531, 2004.
- [8] C. Sminchisescu and A. Jepson, “Generative modeling for continuous non-linearly embedded Visual Inference,” in *Proc. IEEE International Conf. on Machine Learning (ICML)*, 2004, pp. 96–103.
- [9] T.-P. Tian, R. Li, and S. Sclaroff, “Articulated pose estimation in a learned smooth space of feasible solutions,” in *CVPR Learning Workshop*, 2005, pp. 50–57.
- [10] R. Urtasun, D. Fleet, A. Hertzman, and P. Fua, “Priors for people tracking from small training sets,” in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2005, pp. 403–410.
- [11] M. Belkin and P. Niyogi, “Laplacian Eigenmaps and spectral techniques for embedding and clustering,” in *Advances in Neural Information Processing Systems*, 2001, pp. 585–591.
- [12] O. Jenkins and M. Matarić, “A spatio-temporal extension to Isomap nonlinear dimension reduction,” in *Proc. IEEE International Conf. on Machine Learning (ICML)*, 2004, pp. 56–63.
- [13] R. Li, M.-H. Yang, S. Sclaroff, and T.-P. Tian, “Monocular tracking of 3D human motion with a coordinated mixture of factor analyzers,” in *Proc. European Conf. on Computer Vision (ECCV)*, 2006, pp. 137–150.
- [14] S. Roweis and L. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [15] V. Silva and J. Tenenbaum, “Global versus local methods in nonlinear dimensionality reduction,” in *Advances in Neural Information Processing Systems*. The MIT Press, 2003, pp. 705–712.

- [16] J. Tenenbaum, V. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
- [17] M. Law, N. Zhang, and A. Jain, "Nonlinear manifold learning for data stream," in *SIAM data mining*, 2004, pp. 33–44.
- [18] H. Zha and Z. Zhang, "Isometric embedding and continuum ISOMAP," in *Proc. IEEE International Conf. on Machine Learning (ICML)*, 2003, pp. 864–871.
- [19] C. Bishop, M. Svensén, and C. Williams, "GTM: The generative topographic mapping," *Neural Computation*, vol. 10, no. 1, pp. 215–234, 1998.
- [20] N. Lawrence, "Gaussian process latent variable models for visualization of high dimensional data," in *Advances in Neural Information Processing Systems*, vol. 16. The MIT Press, 2004, pp. 329–336.
- [21] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.
- [22] M. Brand, "Charting a manifold," in *Advances in Neural Information Processing Systems*. The MIT Press, 2002, pp. 961–968.
- [23] S. Roweis, L. Saul, and G. Hinton, "Global coordination of local linear models," in *Advances in Neural Information Processing Systems*, vol. 14. The MIT Press, 2001, pp. 889–896.
- [24] Y. W. Teh and S. Roweis, "Automatic alignment of hidden representations," in *Advances in Neural Information Processing Systems*, vol. 15. The MIT Press, 2002, pp. 841–848.
- [25] K. Moon and V. Pavlovic, "Impact of dynamics on subspace embedding and tracking of sequences," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 198–205.
- [26] J. Wang, D. Fleet, and A. Hertzman, "Gaussian process and dynamical models for human motion," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 2, pp. 283–298, 2008.
- [27] Z. Ghahramani and S. Roweis, "Learning nonlinear dynamical systems using an EM algorithm," in *Advances in Neural Information Processing Systems*. The MIT Press, 1998, pp. 599–605.
- [28] L. Ralaivola and F. d'Alché Buc, "Dynamical modeling with kernels for nonlinear time series prediction," in *Advances in Neural Information Processing Systems*. The MIT Press, 2004, pp. 129–136.
- [29] M. Varsta, J. Heikkonen, J. Lampinen, and J. Milln, "Temporal Kohonen map and the recurrent self-organizing map: Analytical and experimental comparison," *Neural processing letters*, vol. 13, no. 3, pp. 237–251, 2001.
- [30] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, vol. 18. The MIT Press, 2006, pp. 1257–1264.
- [31] R. Li, T.-P. Tian, and S. Sclaroff, "Simultaneous learning of nonlinear manifold and dynamical models for high-dimensional time," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2007, pp. 1–8.
- [32] M. Beal, "Variational algorithms for approximate bayesian inference," Ph.D. dissertation, University College London, 2003.
- [33] P. Dellaportas and J. Forster, "Markov chain Monte Carlo model determination for hierarchical and graphical log-linear models," *Biometrika*, vol. 86, pp. 615–633, 1996.
- [34] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2004.
- [35] —, "Free energy minimization algorithm for decoding and cryptanalysis," *Electronics Letters*, vol. 31, no. 6, pp. 446–447, 1995.
- [36] —, "Ensemble learning and evidence maximization," 1995.
- [37] <http://www.variational-bayes.org/vbpapers.html>.

- [38] D. MacKay, "Ensemble learning for hidden Markov models," University of Cambridge, Cavendish Laboratory, Tech. Rep., 1997.
- [39] Z. Ghahramani and M. Beal, "Variational inference for Bayesian mixture of factor analysers," in *Advances in Neural Information Processing Systems*. The MIT Press, 1999, pp. 449–455.
- [40] —, "Propagation algorithms for variational Bayesian learning," in *Advances in Neural Information Processing Systems*. The MIT Press, 2000, pp. 507–513.
- [41] A. Gelman, J. Carlin, H. Stern, and D. Rubin, *Bayesian Data Analysis*. Chapman & Hall/CRC Press, 1995.
- [42] J. Verbeek, "Learning non-linear image manifolds by combining local linear models," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 28, no. 8, pp. 1236–1250, 2006.
- [43] Z. Ghahramani, "Learning dynamic Bayesian networks," in *Adaptive Processing of Sequences and Data Structures*. Springer-Verlag, 1998, pp. 168–197.
- [44] R. Feynman, *Statistical Mechanics*. Addison-Wesley, 1972.
- [45] R. Li, "Simultaneous learning of non-linear manifold and dynamical models for high-dimensional time series," Ph.D. dissertation, Boston University, 2009.
- [46] S.-M. Oh, A. Ranganathan, J. Rehg, and F. Dellaert, "A variational inference method for switching linear dynamic system," Georgia Institute of Technology, Tech. Rep. GIT-GVU-05-16, 2005.
- [47] H. Rauch, F. Tung, and C. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AAIA journal*, vol. 3, no. 8, pp. 1445–1450, 1965.
- [48] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, pp. 4–15, 1986.
- [49] T. Jitsuhiro and S. Nakamura, "Variational Bayesian approach for automatic generation of HMM topologies," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2003, pp. 77–82.
- [50] S. Siddiqi, G. Gordon, and A. Moore, "Fast state discovery for HMM model selection and learning," in *11th International Conference on Artificial Intelligence and Statistics (AI-STATS)*, 2007, pp. 492–499.
- [51] H. Singer and M. Ostendorf, "Maximum likelihood successive state splitting," in *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1996, pp. 1890–1897.
- [52] J. Takami and S. Sagayama, "A successive state splitting algorithm for efficient allophone modeling," in *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1992, pp. 573–576.
- [53] R. Poppe, "Evaluating example-based pose estimation: Experiments on the HumanEva sets," 2007.
- [54] <http://mocap.cs.cmu.edu/>.
- [55] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.
- [56] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard, "Tracking loose-limbed people," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 421–428.
- [57] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust on-line appearance models for vision tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, no. 10, pp. 1296–1311, 2003.