Gaussian Process Dynamical Models for Human Motion

Jack M. Wang, David J. Fleet, *Member, IEEE,* Aaron Hertzmann

The authors are with the Department of Computer Science at the University of Toronto.

Digital Object Indentifier 10.1109/TPAMI.2007.1167

Abstract

We introduce Gaussian process dynamical models (GPDM) for nonlinear time series analysis, with applications to learning models of human pose and motion from high-dimensional motion capture data. A GPDM is a latent variable model. It comprises a low-dimensional latent space with associated dynamics, and a map from the latent space to an observation space. We marginalize out the model parameters in closed-form, using Gaussian process priors for both the dynamics and the observation mappings. This results in a non-parametric model for dynamical systems that accounts for uncertainty in the model. We demonstrate the approach, and compare four learning algorithms on human motion capture data in which each pose is 50-dimensional. Despite the use of small data sets, the GPDM learns an effective representation of the nonlinear dynamics in these spaces.

Index Terms

Machine learning, motion, tracking, animation, stochastic processes, time series analysis.

I. Introduction

Good statistical models for human motion are important for many applications in vision and graphics, notably, visual tracking, activity recognition, and computer animation. It is well known in computer vision that the estimation of 3D human motion from a monocular video sequence is highly ambiguous. Many recently-reported approaches have relied strongly on training prior models to constrain inference to plausible poses and motions [1], [2], [3], [4]. Specific activities could also be classified and recognized by evaluating the likelihood of the observation given models for multiple activities [5]. In computer animation, instead of having animators specify all degrees of freedom in a human-like character, the task of animating characters can be simplified by finding the most likely motion given sparse constraints [6], [7].

One common approach is to learn a probability distribution over the space of possible poses and motions, parameterized by the joint angles of the body, as well as its global position and orientation. Such a density function provides a natural measure of plausibility, assigning higher probabilities to motions that are similar to the training data. The task is challenging due to the high-dimensionality of human pose data, and to the complexity of the motion. However, poses from specific activities often lie near a nonlinear manifold with much lower dimensionality than the number of joint angles. Motivated by this property, a common approach to define the

generative model is to decouple the modeling of pose and motion. The motion is modeled by a dynamical process defined on a lower-dimensional latent space, and the poses are generated by an observation process from the latent space.

The current literature offers a number of generative models where the dynamics is not directly observed. Simple models such as hidden Markov models (HMM) and linear dynamical systems (LDS) are efficient and easily learned, but limited in their expressiveness for complex motions. More expressive models, such as switching linear dynamical systems (SLDS) and nonlinear dynamical systems (NLDS), are more difficult to learn, requiring many parameters that need to be hand-tuned and large amounts of training data.

In this article, we investigate a Bayesian approach to learning NLDS, averaging over model parameters rather than estimating them. Inspired by the fact that averaging over nonlinear regression models leads to a Gaussian process (GP) model, we show that integrating over NLDS parameters can also be performed in closed-form. The resulting Gaussian process dynamical model (GPDM) is fully defined by a set of low-dimensional representations of the training data, with both observation and dynamics processes learned from GP regression. As a natural consequence of GP regression, the GPDM removes the need to select many parameters associated with function approximators while retaining the power of nonlinear dynamics and observation.

Our approach is directly inspired by the Gaussian process latent variable model (GPLVM) [8]. The GPLVM models the joint distribution of the observed data and their corresponding representation in a low-dimensional latent space. It is not, however, a dynamical model; rather, it assumes that data are generated independently, ignoring temporal structure of the input. Here we augment the GPLVM with a latent dynamical model, which gives a closed-form expression for the joint distribution of the observed sequences and their latent space representations. The incorporation of dynamics not only enables predictions to be made about future data, but also helps to regularize the latent space for modeling temporal data in general (e.g., see [9]).

The unknowns in the GPDM consist of latent trajectories and hyperparameters. Generally, if the dynamics process defined by the latent trajectories is smooth, then the models tend to make good predictions. We first introduce a maximum *a posteriori* (MAP) algorithm for estimating all unknowns, and discuss cases where it fails to learn smooth trajectories. Generally, if the dynamics process defined by the latent trajectories is smooth, then the models tend to make good predictions. To learn smoother models we also consider three alternative learning algorithms,

namely, the balanced GPDM (B-GPDM) [10], manually specifying hyperparameters [11], and a two-stage MAP approach. These algorithms present trade-offs in efficiency, synthesis quality, and ability to generalize. We compare learned models based on the visual quality of generated motion, the learned latent space configuration, and their performance in predicting missing frames of test data.

II. Related Work

Dynamical modeling and dimensionality reduction are two essential tools for the modeling of high-dimensional time series data. The latter is often necessary before one can approach density estimation, while the former captures the temporal dependence in the data.

A. Dimensionality Reduction

Many tasks in statistics and machine learning suffer from the "curse of dimensionality." More specifically, the number of samples required to adequately cover a hypervolume increases exponentially with its dimension. Performance in various algorithms, both in terms of speed and accuracy, is often improved by first obtaining a low-dimensional representation of the data.

1) Linear Methods: A natural way to achieve dimensionality reduction is to represent the data in a linear subspace of the observation space. Probabilistic principal components analysis (PPCA) [12], [13], and factor analysis provide both a basis for the subspace and a probability distribution in the observation space. They are straightforward to implement and efficient, and are often effective as a simple preprocessing step before the application of more complex modeling techniques [14], [15], [16]. For purposes of density estimation, however, PCA is often unsuitable since many data sets are not well-modelled by a Gaussian distribution. For instance, images of objects taken over the surface of the viewsphere usually occupy a nonlinear manifold [17], as does human motion capture data (e.g., see Fig. 3(a)).

2) Geometrically Motivated Manifold Learning: Nonlinear dimensionality reduction techniques allow one to represent data points based on their proximity to each other on nonlinear manifolds. Locally linear embedding (LLE) [18] and the Laplacian eigenmap algorithm [19] obtain the embedding by observing that all smooth manifolds are locally linear with respect to sufficiently small neighbourhoods on the manifold. The Isomap algorithm [20] and its variants C-Isomap, L-Isomap [21], and ST-Isomap [22] extend multidimensional scaling by ensuring

the "dissimilarity" measure between pairs of data correspond to approximate geodesics on the manifold.

In applications such as data visualization and analysis [23], it is often sufficient to recover a low-dimensional latent representation of the data without closed-form mappings between the latent space and observation space. While manifold learning methods can be augmented with such mappings as a post-process, they do not provide a probability distribution over data. Techniques such as mixtures-of-Gaussians or the Parzen window method can be used to learn a density model in the lower dimensional space, but as observed in [6] with human pose data, estimation of mixture models is prone to overfitting and requires tuning a large number of parameters in practice. For LLE and Isomap, an additional problem is that they assume the observed data are densely sampled on the manifold, which is typically not true for human motion data.

3) Nonlinear Latent Variable Models: Nonlinear latent variable models (NLVM) are latent variable models that are capable of modeling data generated from a nonlinear manifold. NLVM methods treat the latent coordinates and the nonlinear mapping to observations as parameters in a generative model, which are typically learned using optimization or Monte Carlo simulation when needed. Compared to linear models such as PPCA, a lower number of dimensions can be used in the latent space without compromising reconstruction fidelity.

The Gaussian process latent variable model (GPLVM) [8] is a generalization of probabilistic PCA that allows for a nonlinear mapping from the latent space to the observation space. The model estimates the joint density of the data points and their latent coordinates. The estimates of the latent coordinates are used to represent a learned model, and can be directly used for data visualization. The GPLVM has the attractive property of generalizing reasonably well from small data sets in high-dimensional observation spaces [6], [24], and fast approximation algorithms for sparse GP regression can be used for learning [25], [26].

Except for ST-Isomap, neither manifold learning nor such NLVM methods are designed to model time series data. For applications in vision and graphics, the training data are typically video and motion capture sequences, where the frame-to-frame dependencies are important. Temporal models can also provide a prediction distribution over future data, which is important for tracking applications.

B. Dynamical Systems

The modeling of time-series data using dynamical systems is of interest to fields ranging from control engineering to economics. Given a probabilistic interpretation, state-space dynamical systems corresponding to the graphical model in Figure 1(a) provide a natural framework for incorporating dynamics into latent variable models. In Fig. 1(a), $\mathbf{x}_t$ represents the hidden state of the system at time t, while $\mathbf{y}_t$ represents the observed output of the system at time t. A dynamics function, parameterized by $\mathbf{A}$, and additive process noise govern the evolution of $\mathbf{x}_t$. An observation function, parameterized by $\mathbf{B}$, and measurement noise generate $\mathbf{y}_t$. The noise is assumed to be Gaussian, and the dynamical process is assumed to be Markov. Note that dynamical systems can also have input signals $\mathbf{u}_t$, which are useful for modeling control systems. We focus on the fully unsupervised case, with no system inputs.

Learning such models typically involves estimating parameters $\mathbf{A}$, $\mathbf{B}$, and the noise covariances, and is often referred to as system identification. In a maximum likelihood (ML) framework, the parameters (θ) are chosen to maximize

$$p(\mathbf{y}_{1\ldots N} \mid \theta) \;=\; \int p(\mathbf{y}_{1,\ldots,N}, \mathbf{x}_{1,\ldots,N} \mid \theta)\; d\mathbf{x}_{1,\ldots,N}, \tag{1}$$

as the states, $\mathbf{x}_{1\ldots N}$, are unobserved. The optimization can often be done using the expectation-maximization (EM) algorithm [27]. Once a system is identified, a probability distribution over sequences in the observation space is defined.

1) Linear Dynamical Systems: The simplest and most studied type of dynamical model is the discrete-time linear dynamical system (LDS), where the dynamical and observation functions are linear. The ML parameters can be computed iteratively using the EM algorithm [28], [29]. As part of the E-step, a Kalman smoother is used to infer the expected values of the hidden states. LDS parameters can also be estimated outside of a probabilistic framework; subspace state space system identification (4SID) methods [30] identify the system in closed-form, but are sub-optimal with respect to ML estimates [31].

While computations in LDS are efficient and are relatively easy to analyze, the model is not suitable for modeling complex systems such as human motion [32]. By definition, nonlinear variations in the state space are treated as noise in an LDS model, resulting in overly smoothed motion during simulation. The linear observation function suffers from the same shortcomings as linear latent variable models, as discussed in Section II-A.1.

2) Nonlinear Dynamical Systems: A natural way of increasing the expressiveness of the model is to turn to nonlinear functions. For example switching linear dynamical systems (SLDS) augment LDS with switching states to introduce nonlinearity, and appear to be better models for human motion [32], [33], [34]. Nevertheless, determining the appropriate number of switching states is challenging and such methods often require large amounts of training data as they contain many parameters.

Ijspeert et al. [35] propose an approach for modeling dynamics by observing that, in robotic control applications, the task of motion synthesis is often to make progress towards a goal state. Since this behavior is naturally exhibited by differential equations with well-defined attractor states or limit cycles, faster learning and more robust dynamics can be achieved by simply parameterizing the dynamical model as a differential equation.

The dynamics and observation functions can be modeled directly using nonlinear basis functions. Roweis and Ghahramani [36] use radial basis functions (RBF) to model the nonlinear functions, and identify the system using an approximate EM algorithm. The distribution over hidden states cannot be estimated exactly due to the nonlinearity of the system. Instead, extended Kalman filtering, which approximates the system using locally linear mappings around the current state, is used in the E-step.

In general, a central difficulty in modeling time series data is in determining a model that can capture the nonlinearities of the data without overfitting. Linear autoregressive models require relatively few parameters and allow closed-form analysis, but can only model a limited range of systems. In contrast, existing nonlinear models can model complex dynamics, but usually require many training data points to accurately learn models.

C. Applications

Our work is motivated by human motion modeling for video-based people tracking and data-driven animation. People tracking requires dynamical models in the form of transition densities in order to specify prediction distributions over new poses at each time instant. Similarly, data-driven computer animation can benefit from prior distributions over poses and motion.

1) Monocular Human Tracking: Despite the difficulties with linear subspace models mentioned above, PCA has been applied to video-based people tracking of humans and other vision applications [37], [3], [38], [5]. To this end, the typical data representation is the concatenation of

the entire trajectory of poses to form a single vector in observation space. The lower dimensional PCA subspace is then used as the state space. In place of explicit dynamics, a phase parameter which propagates forward in time can serve as an index to the prior distribution of poses.

Nonlinear dimensionality reduction techniques such as LLE have also been used in the context of human pose analysis. Elgammal and Lee [1] use LLE to learn activity based manifolds from silhouette data. They then use nonlinear regression methods to learn mappings from manifolds back to the silhouette space and to the pose space. Jenkins and Matarić [22] use ST-Isomap to learn embeddings of multi-activity human motion data and robot teleoperation data. Sminchisescu and Jepson [4] used spectral embedding techniques to learn an embedding of human motion capture data. They also learn a mapping back to pose space separately. None of the above approaches learn a dynamics function explicitly, and no density model is learned in [22]. In general, learning the embedding, the mappings, and the density function separately is undesirable.

2) Computer Animation: The applications of probabilistic models for animation revolve around motion synthesis, subject to sparse user constraints. Brand and Hertzmann [15] augment an HMM with stylistic parameters for style-content separation. Li et al. [7] model human motion using a two-level statistical model, combining linear dynamics and Markov switching dynamics. A GPLVM is applied to inverse kinematics by Grochow et al. [6], where ML is used to determine pose given kinematics constraints.

Non-parametric methods have also been used for motion prediction [39] and animation [40], [41], [42]. For example, in animation with motion graphs, each frame of motion is treated as a node in the graph. A similarity measure is assigned to edges in the graph, and can be viewed as transition probabilities in a first-order Markov process. Motion graphs are designed to be used with large motion capture databases, and the synthesis of new motions typically amounts to reordering poses already in the database. An important strength of motion graphs is the ability to synthesis high quality motions, but the need for a large amount of data is undesirable.

Motion interpolation techniques are designed to create natural looking motions relatively far from input examples. Typically, a set of interpolation parameters must be either well-defined (i.e., location of the right hand) or specified by hand (i.e., a number representing emotion) for each example. A mapping from the parameter space to the pose, or motion space is then learned using nonlinear regression [43], [44]. Linear interpolation between motion segments using the spatial-temporal morphable models is possible [45], [46], provided that correspondences

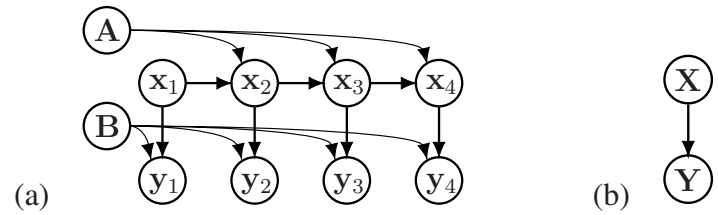


Fig. 1. Time-series graphical models. (a) Nonlinear latent-variable model for time series. (Hyperparameters $\bar{\alpha}$, $\bar{\beta}$ and $\mathbf{W}$ are not shown.) (b) GPDM model. Because the mapping parameters $\mathbf{A}$ and $\mathbf{B}$ have been marginalized over, all latent coordinates $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$ are jointly correlated, as are all poses $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^T$.

can be established between the available segments. More closely related to our work, Mukai and Kuriyama [43] employ a form of Gaussian process regression to learn the mapping from interpolation parameters to pose and motion. In particular, one can view the GPLVM and the GPDM introduced below as interpolation methods with learned interpolation parameters.

III. Gaussian Process Dynamics

The Gaussian process dynamical model (GPDM) is a latent variable model. It comprises a generative mapping from a latent space $\mathbf{x}$ to the observation space $\mathbf{y}$, and a dynamical model in the latent space (Fig. 1). These mappings are in general nonlinear. For human motion modeling, a vector $\mathbf{y}$ in the observation space corresponds to a pose configuration, and a sequence of poses defines a motion trajectory. The latent dynamical model accounts for the temporal dependence between poses. The GPDM is obtained by marginalizing out the parameters of the two mappings, and optimizing the latent coordinates of training data.

More precisely, our goal is to model the probability density of a sequence of vector-valued states $\mathbf{y}_1, \ldots, \mathbf{y}_t, \ldots, \mathbf{y}_N$, with discrete-time index t and $\mathbf{y}_t \in \mathbb{R}^D$. As a basic model, consider a latent variable mapping (3) with first-order Markov dynamics (2):

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}; \mathbf{A}) + \mathbf{n}_{x,t} \tag{2}$$

$$\mathbf{y}_t = g(\mathbf{x}_t; \mathbf{B}) + \mathbf{n}_{y,t}. \tag{3}$$

Here, $\mathbf{x}_t \in \mathbb{R}^d$ denotes the d-dimensional latent coordinates at time t, f and g are mappings parameterized by $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{n}_{x,t}$ and $\mathbf{n}_{y,t}$ are zero-mean, isotropic, white Gaussian noise processes. Fig. 1(a) depicts the graphical model.

While linear mappings have been used extensively in auto-regressive models, here we consider the more general nonlinear case for which f and g are linear combinations of (nonlinear) basis functions:

$$f(\mathbf{x}; \mathbf{A}) \;=\; \sum_i \mathbf{a}_i \; \phi_i(\mathbf{x}) \tag{4}$$

$$g(\mathbf{x}; \mathbf{B}) \;=\; \sum_j \mathbf{b}_j \; \psi_j(\mathbf{x}), \tag{5}$$

for basis functions ϕ_i and ψ_j, with weights $\mathbf{A} \equiv [\mathbf{a}_1, \mathbf{a}_2, \ldots]^T$ and $\mathbf{B} \equiv [\mathbf{b}_1, \mathbf{b}_2, \ldots]^T$. To fit this model to training data, one must select an appropriate number of basis functions, and one must ensure that there is enough data to constrain the shape of each basis function. After the basis functions are chosen, one might estimate the model parameters, $\mathbf{A}$ and $\mathbf{B}$, usually with an approximate form of EM [36]. From a Bayesian perspective, however, the uncertainty in the model parameters is significant, and because the specific forms of f and g are incidental, the parameters should be marginalized out if possible. Indeed, in contrast with previous NLDS models, the general approach we take in the GPDM is to estimate the latent coordinates while marginalizing over model parameters.

Each dimension of the latent mapping, g in (5), is a linear function of the columns of $\mathbf{B}$. Therefore, with an isotropic Gaussian prior on the columns of $\mathbf{B}$, and the Gaussian noise assumption above, one can show that marginalizing over g can be done in closed form [47], [48]. In doing so we obtain a Gaussian density over the observations, $\mathbf{Y} \equiv [\mathbf{y}_1, \ldots, \mathbf{y}_N]^T$, which can be expressed as a product of Gaussian processes (one for each of the D data dimensions):

$$p(\mathbf{Y} \mid \mathbf{X}, \bar{\beta}, \mathbf{W}) = \frac{|\mathbf{W}|^N}{\sqrt{(2\pi)^{ND}|\mathbf{K}_Y|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_Y^{-1}\mathbf{Y}\mathbf{W}^2\mathbf{Y}^T\right)\right), \tag{6}$$

where $\mathbf{K}_Y$ is a kernel matrix with hyperparameters $\bar{\beta}$ that are shared by all observation space dimensions, and hyperparameters $\mathbf{W}$. The elements of the kernel matrix, $\mathbf{K}_Y$, are defined by a kernel function, $(\mathbf{K}_Y)_{ij} \equiv k_Y(\mathbf{x}_i, \mathbf{x}_j)$. For the mapping g, we use the RBF kernel,

$$k_Y(\mathbf{x}, \mathbf{x}') \;=\; \exp\left(-\frac{\beta_1}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \beta_2^{-1}\delta_{\mathbf{x},\mathbf{x}'}. \tag{7}$$

The width of the RBF kernel function is controlled by β_1^{-1}, and β_2^{-1} is the variance of the isotropic additive noise in (3). The ratio of the standard deviation of the data and the additive noise also provides a signal-to-noise ratio (SNR) [11]; here, $SNR(\bar{\beta}) = \sqrt{\beta_2}$.

Following [6], we include D scale parameters, $\mathbf{W} \equiv \mathrm{diag}(w_1, \ldots, w_D)$, which model the variance in each observation dimension.[1] This is important in many data sets for which different dimensions do not share the same length scales, or differ significantly in their variability over time. The use of $\mathbf{W}$ in (6) is equivalent to a GP with kernel function $k_Y(\mathbf{x}, \mathbf{x}')/w_m^2$ for dimension m. That is, the hyperparameters $\{w_m\}_{m=1}^D$ account for the overall scale of the Gaussian processes in each data dimension. In effect, this assumes that each dimension of the input data should exert the same influence on the shared kernel hyperparameters, β_1 and β_2.

The dynamic mapping on the latent coordinates $\mathbf{X} \equiv [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$ is conceptually similar, but more subtle.[2] As above, one can form the joint density over the latent coordinates and the dynamics weights, $\mathbf{A}$, in (4). Then, one can marginalize over the weights $\mathbf{A}$ to obtain

$$p(\mathbf{X} \,|\, \bar{\alpha}) \;=\; \int p(\mathbf{X} \,|\, \mathbf{A}, \bar{\alpha})\, p(\mathbf{A} \,|\, \bar{\alpha})\, d\mathbf{A}, \tag{8}$$

where $\bar{\alpha}$ is a vector of kernel hyperparameters. Incorporating the Markov property (2) gives

$$p(\mathbf{X} \,|\, \bar{\alpha}) = p(\mathbf{x}_1) \int \prod_{t=2}^{N} p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{A}, \bar{\alpha}) p(\mathbf{A} \,|\, \bar{\alpha})\, d\mathbf{A}. \tag{9}$$

Finally, with an isotropic Gaussian prior on the columns of $\mathbf{A}$, one can show that (9) reduces to

$$\begin{aligned} &p(\mathbf{X} \,|\, \bar{\alpha}) \\ &\quad = \frac{p(\mathbf{x}_1)}{\sqrt{(2\pi)^{(N-1)d} |\mathbf{K}_X|^d}} \exp\left(-\frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{2:N} \mathbf{X}_{2:N}^T \right) \right), \end{aligned} \tag{10}$$

where $\mathbf{X}_{2:N} = [\mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, and $\mathbf{K}_X$ is the $(N-1) \times (N-1)$ kernel matrix constructed from $\mathbf{X}_{1:N-1} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N-1}]^T$. Below we also assume that $\mathbf{x}_1$ also has a Gaussian prior.

The dynamic kernel matrix has elements defined by a kernel function, $(\mathbf{K}_X)_{ij} \equiv k_X(\mathbf{x}_i, \mathbf{x}_j)$, for which a linear kernel is a natural choice; i.e.,

$$k_X(\mathbf{x}, \mathbf{x}') = \; \alpha_1 \mathbf{x}^T \mathbf{x}' + \alpha_2^{-1} \delta_{\mathbf{x},\mathbf{x}'}. \tag{11}$$

In this case, (10) is the distribution over state trajectories of length N, drawn from a distribution of auto-regressive models with a preference for stability [49]. While a substantial portion of

[1] With the addition of the scale parameters, $\mathbf{W}$, the latent variable mapping (3) becomes $\mathbf{y}_t = \mathbf{W}^{-1}\left(g(\mathbf{x}_t; \mathbf{B}) + \mathbf{n}_{y,t} \right)$.

[2] Conceptually, we would like to model each pair $(\mathbf{x}_t, \mathbf{x}_{t+1})$ as a training pair for regression with g. However, we cannot simply substitute them directly into the GP model of (6) as this leads to the nonsensical expression $p(\mathbf{x}_2, \ldots, \mathbf{x}_N \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_{N-1})$.

human motion (as well as many other systems) can be well-modeled by linear dynamical models, ground contacts introduce nonlinearity [32]. We found that the linear kernel alone is unable to synthesize good walking motions (e.g., see Fig. 3(h-i)). Therefore, we typically use a "linear + RBF" kernel:

$$k_X(\mathbf{x}, \mathbf{x}') = \alpha_1 \exp\left(-\frac{\alpha_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \alpha_3 \mathbf{x}^T \mathbf{x}' + \alpha_4^{-1} \delta_{\mathbf{x},\mathbf{x}'}. \tag{12}$$

The additional RBF term enables the GPDM to model nonlinear dynamics, while the linear term allows the system to regress to linear dynamics when predictions are made far from the existing data. Hyperparameters α_1, α_2 represent the output scale and the inverse width of the RBF terms, and α_3 represents the output scale of the linear term. Together, they control the relative weighting between the terms, while α_4^{-1} represents the variance of the noise term $\mathbf{n}_{x,t}$. The SNR of the dynamics process is given by $SNR(\bar{\alpha}) = \sqrt{(\alpha_1 + \alpha_3)\alpha_4}$.

It should be noted that, due to the marginalization over $\mathbf{A}$, the joint distribution of the latent coordinates is *not* Gaussian. One can see this in (10), where latent variables occur both inside the kernel matrix and outside of it; i.e., the log likelihood is not quadratic in $\mathbf{x}_t$. Moreover, the distribution over state trajectories in a nonlinear dynamical system is in general non-Gaussian.

Following [8], we place uninformative priors on the kernel hyperparameters ($p(\bar{\alpha}) \propto \prod_i \alpha_i^{-1}$, and $p(\bar{\beta}) \propto \prod_i \beta_i^{-1}$). Such priors represent a preference for a small output scale (i.e., small α_1, α_3), a large width for the RBFs (i.e., small α_2, β_1), and large noise variances (i.e., small α_4, β_2). We also introduce a prior on the variances w_m that comprise the elements of $\mathbf{W}$. In particular, we use a broad half-normal prior on $\mathbf{W}$; i.e.,

$$p(\mathbf{W}) \;=\; \prod_{m=1}^{D} \frac{2}{\kappa\sqrt{2\pi}} \exp\left(-\frac{w_m^2}{2\kappa^2}\right), \tag{13}$$

where $w_m > 0$, and κ is set to 10^3 in the experiments below. Such a prior reflects our belief that every data dimension has a nonzero variance. This prior avoids singularities in the estimation of the parameters w_j (see Alg. 1), and prevents any one data dimension with an anomalously small variance from dominating the estimation of the remaining kernel parameters.

Taken together, the priors, the latent mapping, and the dynamics define a generative model for time-series observations (Fig. 1(b)):

$$\begin{aligned} &p(\mathbf{X}, \mathbf{Y}, \bar{\alpha}, \bar{\beta}, \mathbf{W}) \\ &\qquad = p(\mathbf{Y} \mid \mathbf{X}, \bar{\beta}, \mathbf{W})\, p(\mathbf{X} \mid \bar{\alpha})\, p(\bar{\alpha})\, p(\bar{\beta})\, p(\mathbf{W}). \end{aligned} \tag{14}$$

A. Multiple Sequences

This model extends naturally to multiple sequences $\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(P)}$, with lengths $N_1, \ldots, N_P$. Each sequence has associated latent coordinates, $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(P)}$, within a shared latent space. To form the joint likelihood, concatenate all sequences, and proceed as above with (6). A similar concatenation applies for the latent dynamical model (10), but accounting for the assumption that the first pose of sequence i is independent of the last pose of sequence $i-1$. That is, let

$$\mathbf{X}_{2:N} = \left[{\mathbf{X}^{(1)}_{2:N_1}}^T, \ldots, {\mathbf{X}^{(P)}_{2:N_P}}^T \right]^T \tag{15}$$

$$\mathbf{X}_{1:N-1} = \left[{\mathbf{X}^{(1)}_{1:N_1-1}}^T, \ldots, {\mathbf{X}^{(P)}_{1:N_P-1}}^T \right]^T. \tag{16}$$

The kernel matrix $\mathbf{K}_X$ is constructed with rows of $\mathbf{X}_{1:N-1}$ as in (10), and is of size $(N-P) \times (N-P)$. Finally, we place an isotropic Gaussian prior on the first pose of each sequence.

B. Higher-Order Features

The GPDM can be extended to model higher-order Markov chains, and to model velocity and acceleration in inputs and outputs. For example, a second-order dynamical model,

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}; \mathbf{A}) + \mathbf{n}_{x,t}, \tag{17}$$

can be used to explicitly model dependence on two past frames (or on velocity). Accordingly, the kernel function will depend on the current and previous latent positions,

$$\begin{aligned} & k_X(\,[\mathbf{x}_t, \mathbf{x}_{t-1}],\, [\mathbf{x}_\tau, \mathbf{x}_{\tau-1}]\,) \\ & \quad = \alpha_1 \exp\left(-\frac{\alpha_2}{2}||\mathbf{x}_t - \mathbf{x}_\tau||^2 - \frac{\alpha_3}{2}||\mathbf{x}_{t-1} - \mathbf{x}_{\tau-1}||^2\right) \\ & \qquad + \alpha_4\, \mathbf{x}_t^T \mathbf{x}_\tau + \alpha_5\, \mathbf{x}_{t-1}^T \mathbf{x}_{\tau-1} + \alpha_6^{-1} \delta_{t,\tau}. \end{aligned} \tag{18}$$

Similarly, the dynamics can be formulated to predict velocity in the latent space,

$$\mathbf{v}_{t-1} = f(\mathbf{x}_{t-1}; \mathbf{A}) + \mathbf{n}_{x,t}. \tag{19}$$

Velocity prediction may be more appropriate for modeling smooth motion trajectories. Using a first-order Taylor series approximation of position as a function of time, in the neighbourhood of $t-1$, with time-step Δt, we have $\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_{t-1}\Delta t$. The dynamics likelihood $p(\mathbf{X} \,|\, \bar{\alpha})$ can then be written by redefining $\mathbf{X}_{2:N} = [\mathbf{x}_2 - \mathbf{x}_1, \ldots, \mathbf{x}_N - \mathbf{x}_{N-1}]^T / \Delta t$ in (10). For a fixed time-step of $\Delta t = 1$, velocity prediction is analogous to using $\mathbf{x}_{t-1}$ as a "mean function" for predicting $\mathbf{x}_t$. Higher-order features have previously been used in GP regression as a way to reduce the prediction variance [50], [51].

C. Conditional GPDM

Thus far we have defined the generative model and formed the posterior distribution (14). Leaving the discussion of learning algorithms to Section IV, we recall here that the main motivation for the GPDM is to use it as a prior model of motion. A prior model needs to evaluate, or predict, whether a new observed motion is likely.

Given the learned model, $\Gamma = \left\{\mathbf{Y}, \mathbf{X}, \bar{\alpha}, \bar{\beta}, \mathbf{W}\right\}$, the distribution over a new sequence $\mathbf{Y}^{(*)}$ and its associated latent trajectory $\mathbf{X}^{(*)}$ is given by

$$p(\mathbf{Y}^{(*)}, \mathbf{X}^{(*)} \,|\, \Gamma) = p(\mathbf{Y}^{(*)} \,|\, \mathbf{X}^{(*)}, \Gamma)\; p(\mathbf{X}^{(*)} \,|\, \Gamma) \tag{20}$$

$$= \frac{p(\mathbf{Y}, \mathbf{Y}^{(*)} \,|\, \mathbf{X}, \mathbf{X}^{(*)}, \bar{\beta}, \mathbf{W})}{p(\mathbf{Y} \,|\, \mathbf{X}, \bar{\beta}, \mathbf{W})} \frac{p(\mathbf{X}, \mathbf{X}^{(*)} \,|\, \bar{\alpha})}{p(\mathbf{X} \,|\, \bar{\alpha})} \tag{21}$$

$$\propto\; p(\mathbf{Y}, \mathbf{Y}^{(*)} \,|\, \mathbf{X}, \mathbf{X}^{(*)}, \bar{\beta}, \mathbf{W})\; p(\mathbf{X}, \mathbf{X}^{(*)} \,|\, \bar{\alpha}), \tag{22}$$

where $\mathbf{Y}^{(*)}$ and $\mathbf{X}^{(*)}$ are $M \times D$ and $M \times d$ matrices respectively. Here, (20) factors the conditional density into a density over latent trajectories, and a density over poses conditioned on latent trajectories, which we refer to as the reconstruction and dynamics predictive distributions.

For sampling and optimization applications, we only need to evaluate (20) up to a constant. In particular, we can form the joint distribution over both new and observed sequences (22) by following the discussion in Section III-A. The most expensive operation in evaluating (22) is the inversion of kernel matrices of size $(N + M) \times (N + M)$.[3] When the number of training data is large, the computation cost can be reduced by evaluating (20) in terms of pre-computed block entries to the kernel matrices in (22).

Since the joint distribution over $\{\mathbf{Y}^{(*)}, \mathbf{Y}\}$ in (22) is Gaussian, it follows that $\mathbf{Y}^{(*)} \,|\, \mathbf{Y}$ is also Gaussian. More specifically, if the reconstruction kernel matrix in (22) is given by

$$\mathbf{K}_{Y,Y^{(*)}} \;=\; \begin{bmatrix} \begin{bmatrix} \\ \mathbf{K}_Y \\ \\ \end{bmatrix} & \begin{bmatrix} \\ \mathbf{A} \\ \\ \end{bmatrix} \\ \begin{bmatrix} \mathbf{A}^T \end{bmatrix} & \begin{bmatrix} \mathbf{B} \end{bmatrix} \end{bmatrix}, \tag{23}$$

[3]The dynamics kernel is only smaller by a constant subtraction.

where $(\mathbf{A})_{ij} = k_Y(\mathbf{x}_i, \mathbf{x}_j^{(*)})$ and $(\mathbf{B})_{ij} = k_Y(\mathbf{x}_i^{(*)}, \mathbf{x}_j^{(*)})$ are elements of $N \times M$ and $M \times M$ kernel matrices respectively, then

$$p(\mathbf{Y}^{(*)} \,|\, \mathbf{X}^{(*)}, \Gamma) = \frac{|\mathbf{W}|^M}{\sqrt{(2\pi)^{MD}|\mathbf{K}_{Y^{(*)}}|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{Y^{(*)}}^{-1}\mathbf{Z}_Y\mathbf{W}^2\mathbf{Z}_Y^T\right)\right), \tag{24}$$

where $\mathbf{Z}_Y = \mathbf{Y}^{(*)} - \mathbf{A}^T\mathbf{K}_Y^{-1}\mathbf{Y}$ and $\mathbf{K}_{Y^{(*)}} = \mathbf{B} - \mathbf{A}^T\mathbf{K}_Y^{-1}\mathbf{A}$. Here, $\mathbf{K}_Y$ only needs to be inverted once using the learned model. To evaluate (24) for new sequences, only $\mathbf{K}_{Y^{(*)}}$ must be inverted, which has size $M \times M$, and is not dependent on the size of the training data.

The distribution $p(\mathbf{X}^{(*)} \,|\, \Gamma) = \frac{p(\mathbf{X}, \mathbf{X}^{(*)} \,|\, \bar{\alpha})}{p(\mathbf{X} \,|\, \bar{\alpha})}$ is not Gaussian, but, by simplifying the quotient on the right hand side, an expression similar to (24) can be obtained. As above, if the dynamics kernel matrix in (22) is given by

$$\mathbf{K}_{X,X^{(*)}} = \begin{bmatrix} \begin{bmatrix} \\ \mathbf{K}_X \\ \\ \end{bmatrix} & \begin{bmatrix} \\ \mathbf{C} \\ \\ \end{bmatrix} \\ \begin{bmatrix} \mathbf{C}^T \end{bmatrix} & \begin{bmatrix} \mathbf{D} \end{bmatrix} \end{bmatrix}, \tag{25}$$

where $(\mathbf{C})_{ij} = k_X(\mathbf{x}_i, \mathbf{x}_j^{(*)})$ and $(\mathbf{D})_{ij} = k_X(\mathbf{x}_i^{(*)}, \mathbf{x}_j^{(*)})$ are elements of $(N-P) \times (M-1)$ and $(M-1) \times (M-1)$ kernel matrices respectively, then

$$p(\mathbf{X}^{(*)} \,|\, \Gamma) = \frac{p(\mathbf{x}_1^{(*)})}{\sqrt{(2\pi)^{(M-1)d}|\mathbf{K}_{X^{(*)}}|^d}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{X^{(*)}}^{-1}\mathbf{Z}_X\mathbf{Z}_X^T\right)\right), \tag{26}$$

where $\mathbf{Z}_X = \mathbf{X}_{2:N}^{(*)} - \mathbf{C}^T\mathbf{K}_X^{-1}\mathbf{X}_{2:N}$ and $\mathbf{K}_{X^{(*)}} = \mathbf{D} - \mathbf{C}^T\mathbf{K}_X^{-1}\mathbf{C}$. The matrices $\mathbf{X}_{2:N}$ and $\mathbf{X}_{2:N}^{(*)}$ are described in Section III-A. As with $\mathbf{K}_Y$ above, $\mathbf{K}_X$ only need to be inverted once. Also similar to $\mathbf{K}_{Y^{(*)}}$, the complexity of inverting $\mathbf{K}_{X^{(*)}}$ does not depend on the size of the training data.

IV. GPDM Learning

Learning the GPDM from measured data $\mathbf{Y}$ entails using numerical optimization to estimate some or all of the unknowns in the model $\{\mathbf{X}, \bar{\alpha}, \bar{\beta}, \mathbf{W}\}$. A model gives rise to a distribution over new poses and their latent coordinates (20). We expect modes in this distribution to correspond

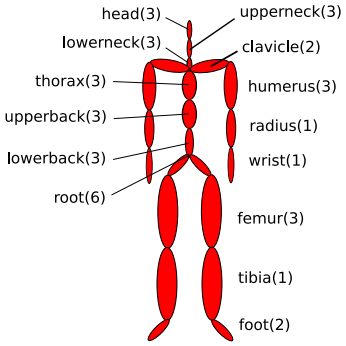


Fig. 2. The skeleton used in our experiments is a simplified version of the default skeleton in the CMU mocap database. The numbers in parentheses indicate the number of DOFs for the joint directly above the labeled body node in the kinematic tree.

to motions similar to the training data and their latent coordinates. In the following sections, we evaluate the models based on examining random samples drawn from the the models, and the models' performance in filling in missing frames. We find that models with visually smooth latent trajectories $\mathbf{X}$ not only better match our intuitions, but also achieve better quantitative results. However, care must be taken in designing the optimization method, including the objective function itself. We discuss four options: MAP, B-GPDM [10], hand-tuning $\bar{\alpha}$ [11], and two-stage MAP in this section.

The data used for all experiments are human motion capture data from the CMU motion capture database. As shown in Fig. 2, we use a simplified skeleton where each pose is defined by 44 Euler angles for joints, 3 global (torso) pose angles, and 3 global (torso) translational velocities.[4] The data are mean-subtracted, but otherwise we do not apply preprocessing such as time synchronization or time warping.

A. MAP Estimation

A natural learning algorithm for the GPDM is to minimize the joint negative log-posterior of the unknowns, $-\ln p(\mathbf{X}, \bar{\alpha}, \bar{\beta}, \mathbf{W} \mid \mathbf{Y})$, that is given, up to an additive constant, by

$$\mathcal{L} = \mathcal{L}_{\mathcal{Y}} + \mathcal{L}_{\mathcal{X}} + \sum_j \ln \beta_j + \frac{1}{2\kappa^2} \mathrm{tr}\left(\mathbf{W}^2\right) + \sum_j \ln \alpha_j, \tag{27}$$

[4]For each frame, the global velocity is set to the difference between the next and the current frame. Velocity for the last frame is copied from the second to last frame.

where

$$\mathcal{L}_{\mathcal{Y}} = \frac{D}{2}\ln|\mathbf{K}_Y| + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_Y^{-1}\mathbf{Y}\mathbf{W}^2\mathbf{Y}^T\right) - N\ln|\mathbf{W}| \tag{28}$$

$$\mathcal{L}_{\mathcal{X}} = \frac{d}{2}\ln|\mathbf{K}_X| + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_X^{-1}\mathbf{X}_{2:N}\mathbf{X}_{2:N}^T\right) + \frac{1}{2}\mathbf{x}_1^T\mathbf{x}_1. \tag{29}$$

As described in Alg. 1, we alternate between minimizing $\mathcal{L}$ with respect to $\mathbf{W}$ in closed form[5], and with respect to $\{\mathbf{X}, \bar{\alpha}, \bar{\beta}\}$ using scaled conjugate gradient (SCG). The latent coordinates are initialized using a subspace projection onto the first d principal directions given by PCA applied to mean-subtracted data $\mathbf{Y}$. In our experiments, we fix the number of outer loop iterations as $I = 100$ and the number of SCG iterations per outer loop as $J = 10$.

Figure 3 shows a GPDM on a 3D latent space, learned using MAP estimation. The training data comprised two gait cycles of a person walking. The initial coordinates provided by PCA are shown in Fig. 3(a). Fig. 3(c) shows the MAP latent space. Note that the GPDM is significantly smoother than a 3D GPLVM (i.e., without dynamics), shown in Fig. 3(b).

Figure 5(b) shows a GPDM latent space learned from walking data of four different walkers. In contrast to the model learned with a single walker in Fig. 3, the latent trajectories here are not smooth. There are small clusters of latent positions separated by large jumps in the latent space. While such models produce good reconstructions from latent positions close to the training data, they often produce poor dynamical predictions. For example, neither the sample trajectories show in Fig. 5(d) nor the reconstructed poses in Fig. 10(a) resemble the training data particularly well.

B. Balanced GPDM

Since the $\mathcal{L}_{\mathcal{X}}$ term in MAP estimation penalizes unsmooth trajectories, one way to encourage smoothness is to increase the weight on $\mathcal{L}_{\mathcal{X}}$ during optimization. Urtasun et al. [10] suggest replacing $\mathcal{L}_{\mathcal{X}}$ in (27) with $\frac{D}{d}\mathcal{L}_{\mathcal{X}}$, thereby "balancing" the objective function based on the ratio between dimensions of data and latent spaces ($\frac{D}{d}$). Learned from the same data as that in Fig. 5(b), Fig. 6(a) shows a model learned using the balanced GPDM (B-GPDM). It is clear that the latent model is now much smoother. Furthermore, random samples drawn from the model yield

[5]The update for w_k shown in Alg. 1 is a MAP estimate, given the current values of $\{\mathbf{X}, \bar{\alpha}, \bar{\beta}\}$. It is bounded by $\kappa\sqrt{N}$, which is due to our choice of prior on $\mathbf{W}$ (13). Note that a prior of $p(w_k) \propto w_k^{-1}$ would not regularize the estimation of w_k since its MAP estimate then becomes undefined when $\mathbf{d}^T\mathbf{K}_Y^{-1}\mathbf{d} = 0$.

Algorithm 1 MAP estimation of $\left\{\mathbf{X}, \bar{\alpha}, \bar{\beta}, \mathbf{W}\right\}$

Require: Data $\mathbf{Y}$. Integers $\{d, I, J\}$.

Initialize $\mathbf{X}$ with PCA on $\mathbf{Y}$ with d dimensions.

Initialize $\bar{\alpha} \Leftarrow (0.9, 1, 0.1, e)$, $\bar{\beta} \Leftarrow (1, 1, e)$, $\{w_k\} \Leftarrow 1$.

for $i = 1$ to I **do**

  for $j = 1$ to D **do**

    $\mathbf{d} \Leftarrow \left[(\mathbf{Y})_{1j}, \ldots, (\mathbf{Y})_{Nj}\right]^T$

    $w_j^2 \Leftarrow N\left(\mathbf{d}^T \mathbf{K}_Y^{-1} \mathbf{d} + \frac{1}{\kappa^2}\right)^{-1}$

  end for

  $\left\{\mathbf{X}, \bar{\alpha}, \bar{\beta}\right\} \Leftarrow$ optimize (27) w.r.t. $\left\{\mathbf{X}, \bar{\alpha}, \bar{\beta}\right\}$ using SCG for J iterations.

end for

better walking simulations, and it has proved to be successful as a prior for 3D people tracking [52], [10]. Though simple and effective, the weighting constant in the B-GPDM does not have a valid probabilistic interpretation, however, similar variations on time-series analysis have been used successfully in speech recognition with HMMs [53], [54].

C. Manually Specified Hyperparameters

The B-GPDM manipulates the objective function to favour smooth latent trajectories. A more principled way of achieving this is by ensuring that $p(\mathbf{X} \,|\, \bar{\alpha})$ represents a strong preference for smooth trajectories, which can be achieved by selecting $\bar{\alpha}$ by hand instead of optimizing for it. One way to select a suitable $\bar{\alpha}$ is to examine samples from $p(\mathbf{X} \,|\, \bar{\alpha})$ [11]. If a sufficiently strong prior is selected, models with smooth trajectories can be learned. Figure 7(a) shows a four-walker model learned with such a smoothness prior. We set $\bar{\alpha} = [0.009, 0.2, 0.001, 1e6]^T$ inspired by observations from [11].[6] It is conceivable that a better choice of $\bar{\alpha}$ could give a very different set of latent trajectories, and better results in our experiments.

D. Two-Stage MAP Estimation

[6]Note, the model in [11] used velocity prediction (cf. Section III-B), and an RBF kernel (rather than linear + RBF).

Algorithm 2 MAP estimation of $\{\bar{\alpha}, \bar{\beta}, \mathbf{W}\}$ using MCEM

Require: Data matrix $\mathbf{Y}$. Integers $\{d, R, I, J, K\}$.

Initialize $\bar{\alpha} \Leftarrow (0.9, 1, 0.1, e)$, $\bar{\beta} \Leftarrow (1, 1, e)$, $\{w_k\} \Leftarrow 1$.

for $i = 1$ to I **do**

Generate $\{\mathbf{X}^{(r)}\}_{r=1}^{R} \sim p(\mathbf{X} \mid \mathbf{Y}, \bar{\alpha}, \bar{\beta}, \mathbf{W})$ using hybrid Monte Carlo sampling.

Construct $\{\mathbf{K}_Y^{(r)}, \mathbf{K}_X^{(r)}\}_{r=1}^{R}$ from $\{\mathbf{X}^{(r)}\}_{r=1}^{R}$.

for $j = 1$ to J **do**

for $k = 1$ to D **do**

$\mathbf{d} \Leftarrow [(\mathbf{Y})_{1k}, \ldots, (\mathbf{Y})_{Nk}]^T$

$w_k^2 \Leftarrow N \left(\mathbf{d}^T \left(\frac{1}{R} \sum_{r=1}^{R} \left(\mathbf{K}_Y^{(r)} \right)^{-1} \right) \mathbf{d} + \frac{1}{\kappa^2} \right)^{-1}$

end for

$\{\bar{\alpha}, \bar{\beta}\} \Leftarrow$ minimize (31) w.r.t. $\{\bar{\alpha}, \bar{\beta}\}$ using SCG for K iterations.

end for

end for

Both the B-GPDM and hand-tuning $\bar{\alpha}$ are practical ways to encourage smoothness. However, MAP learning is still prone to overfitting in high-dimensional spaces.[7] When we seek a MAP estimate, we are looking to approximate the posterior distribution with a delta function. Here, as there are clearly a multiplicity of posterior modes, the estimate may not represent a significant proportion of the posterior probability mass [47]. To avoid this problem, we could aim to find a mode of the posterior that effectively represents a significant proportion of local probability mass. In effect this amounts to minimizing the expected loss with respect to different loss functions (cf. [55]).

Toward this end, we consider a two-stage algorithm for estimating unknowns in the model: first, estimate the hyperparameters $\Theta = \{\bar{\alpha}, \bar{\beta}, \mathbf{W}\}$ with respect an unknown distribution of latent trajectories $\mathbf{X}$, and then estimate $\mathbf{X}$ while holding Θ fixed. Because $\mathbf{X}$ comprises the vast majority of the unknown model parameters, by marginalizing over $\mathbf{X}$ and therefore taking its uncertainty into account while estimating Θ, we are finding a solution that is more representative of the

[7]We're optimizing in a space with dimension over $N \times d$ since there is one latent point for every training pose.

posterior distribution on average. This is also motivated by the fact that parameter estimation algorithms for NLDS typically account for uncertainty in the latent space [36]. Thus, in the first step, we find an estimate of Θ that maximizes $p(\mathbf{Y} \mid \Theta) = \int p(\mathbf{Y}, \mathbf{X} \mid \Theta) d\mathbf{X}$. The optimization is approximated using a variant of EM [27], [56] called Monte Carlo EM (MCEM) [57].

In the E-step of the i^{th} iteration, we compute the expected complete negative log likelihood[8] $-\ln p(\mathbf{Y}, \mathbf{X} \mid \Theta)$ under $p(\mathbf{X}|\mathbf{Y}, \Theta^i)$, the posterior given the current estimate of hyperparameters,

$$\mathcal{L}_{\mathcal{E}}(\Theta) \;=\; -\int_{\mathbf{X}} p(\mathbf{X} \mid \mathbf{Y}, \Theta^i) \ln p(\mathbf{Y}, \mathbf{X} \mid \Theta) d\mathbf{X}. \tag{30}$$

In the M-step, we seek a set of hyperparameters Θ^{i+1}, that minimizes $\mathcal{L}_{\mathcal{E}}$. In MCEM, we numerically approximate (30) by sampling from $p(\mathbf{X}|\mathbf{Y}, \Theta^i)$ using hybrid Monte Carlo (HMC) [47]:[9]

$$\mathcal{L}_{\mathcal{E}}(\Theta) \;\approx\; -\frac{1}{R} \sum_{r=1}^{R} \ln p(\mathbf{Y}, \mathbf{X}^{(r)} \mid \Theta), \tag{31}$$

where $\{\mathbf{X}^{(r)}\}_{r=1}^{R} \sim p(\mathbf{X}|\mathbf{Y}, \Theta^i)$. The derivative with respect to the hyperparameters is given by

$$\frac{\partial \mathcal{L}_{\mathcal{E}}}{\partial \Theta} \;\approx\; -\frac{1}{R} \sum_{r=1}^{R} \frac{\partial}{\partial \Theta} \ln p(\mathbf{Y}, \mathbf{X}^{(r)} \mid \Theta). \tag{32}$$

The approximations are simply sums of the derivatives of the complete log likelihood, which we used for optimizing (14). Algorithm 2 describes the estimation in pseudocode. We set $R = 50, I = 10, J = 10, K = 10$ in our experiments.

In the second stage, we maximize $\ln p(\mathbf{X}, \Theta \mid \mathbf{Y})$ with respect to $\mathbf{X}$ using SCG. The resulting trajectories estimated by two-stage MAP on walking data are shown in Fig. 8(a). In contrast with previous methods, data from the four walking subjects are placed in separate parts of the latent space. On the golf swings data set (Fig. 9(a)), smoother trajectories are learned compared to the MAP model in Figure 4(a).

V. Evaluation of Learned Models

The computational bottleneck for the learning algorithms above is the inversion of the kernel matrices, which is necessary to evaluate the likelihood function and its gradient. Learning using

[8]In practice, we compute the expected value of the log of (14), which is regularized by the priors on the hyperparameters.

[9]We initialize the sampler using SCG to find a mode in $p(\mathbf{X}|\mathbf{Y}, \Theta^i)$, and 50 samples in total are returned to compute the expectation. We use 10 burn-in samples, take 100 steps per trajectory, and the step size is adjusted so that an acceptance rate of 0.6 to 0.95 is achieved on the first 25 samples.

MAP estimation, the B-GPDM, and fixed hyperparameters $\bar{\alpha}$, requires approximately 6000 inversions of the kernel matrices, given our specified number of iterations. These algorithms take approximately 500 seconds for a data set of 289 frames. The two-stage MAP algorithm is more expensive to run, as both the generation of samples in the E-step and the averaging of samples in the M-step require evaluation of the likelihood function. The experiments below used approximately 400,000 inversions, taking about 9 hours for the same data set of 289 frames. Note that our implementation is written in Matlab, with no attempts made to optimize performance, nor is sparsification exploited (e.g., see [25]).

In the rest of this section, we discuss visualizations and comparisons of GPDM models. We first consider visualization methods on a single-walker model and golf swing models learned using MAP and two-stage MAP, then we discuss the failure of MAP in learning a four-walker model. Finally, we compare four-walker models learned using the different methods above. The comparison is based on visually examining samples from the distribution over new motions, and errors in the task of filling in missing frames of data.

A. Single-Walker Model

Figure 3 shows 3D latent models learned from data comprising two walk cycles from a single subject.[10] In all experiments here we use a 3D latent space. Learning with more than three latent dimensions significantly increases the number of latent coordinates to be estimated. Conversely, in two dimensions the latent trajectories often intersect which makes learning difficult. In particular, GPs are function mappings, providing one prediction for each latent position. Accordingly, learned 2D GPDMs often contain large "jumps" in latent trajectories as the optimization breaks the trajectory to avoid nearby positions requiring inconsistent temporal predictions.

Figure 3(b) shows a 3D GPLVM (i.e., without dynamics) learned from walking data. Note that, without the dynamical model, the latent trajectories are not smooth; there are several locations where consecutive poses in the walking sequence are relatively far apart in the latent space. In contrast, Fig. 3(c) shows that the GPDM produces a much smoother configuration of latent positions. Here the GPDM arranges the latent positions roughly in the shape of a saddle.

[10]CMU database file 07_01.amc, frames 1 to 260, down-sampled by a factor of 2.

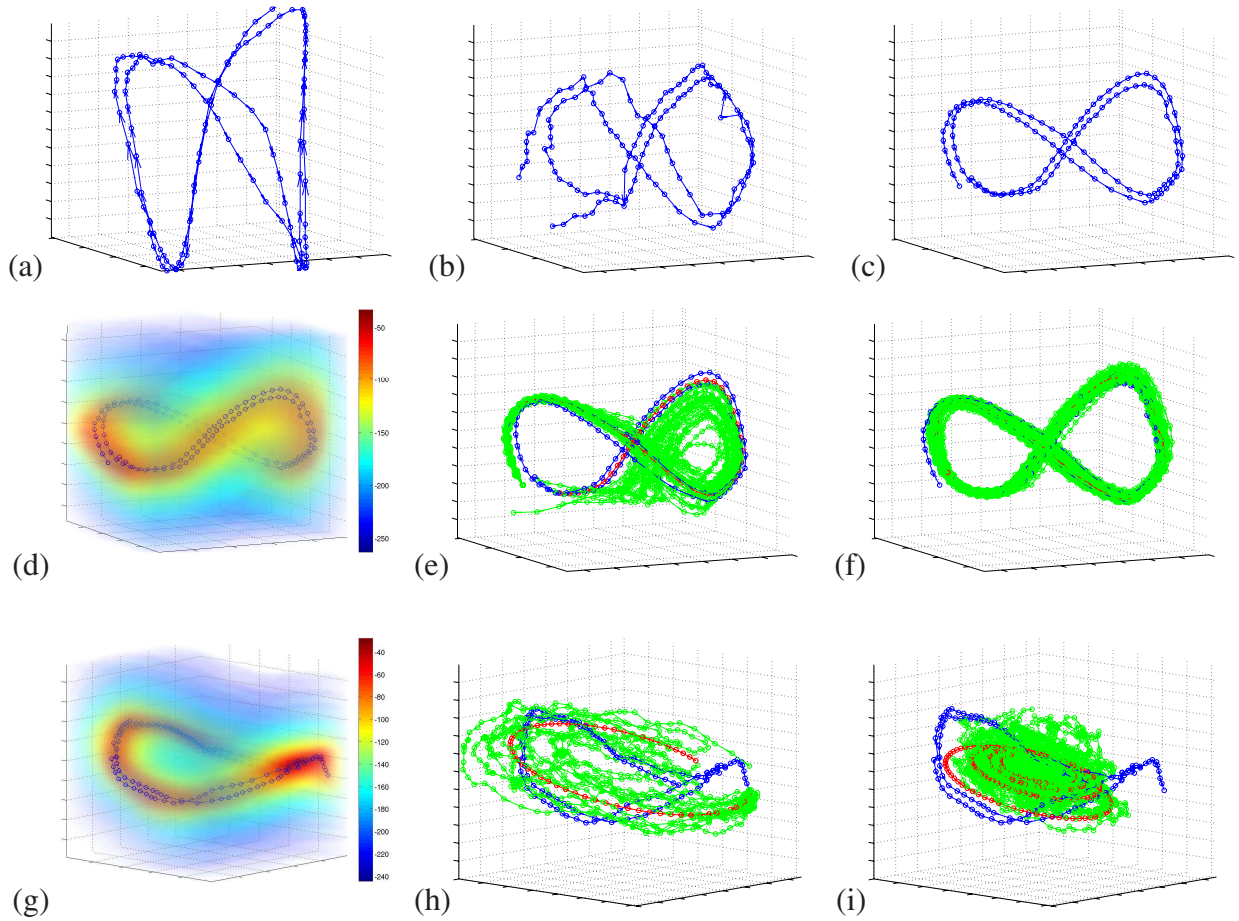


Fig. 3. Models learned from a walking sequence comprising two gait cycles. The PCA initializations (a), latent coordinates learned with a GPLVM (b) and GPDM (c) are shown in blue. Vectors depict the temporal sequence. (d) $-\ln$ variance for reconstruction shows positions in latent space that are reconstructed with high confidence. (e) Random trajectories drawn from the dynamic predictive distribution using hybrid Monte Carlo are green, the red trajectory is the mean-prediction sample. (f) Longer random trajectories drawn from the dynamics predictive distribution. (g-i) $-\ln$ variance for reconstruction, random trajectories, and longer random trajectories created in the same fashion as (d-f), using a model learned with the linear dynamics kernel. Note that the samples do not follow the training data closely, and longer trajectories are attracted to the origin.

Figure 3(d) shows a volume visualization of the value $\ln p\left(\mathbf{x}^{(*)}, \mathbf{y}^{(*)} = \mu_Y(\mathbf{x}^{(*)}) \,|\, \Gamma\right)$, where $\mu_Y(\mathbf{x}^{(*)})$ is the mean of the Gaussian process for pose reconstruction [47], as a function of the latent space position $\mathbf{x}^{(*)}$; i.e.,

$$\mu_Y(\mathbf{x}) \;=\; \mathbf{Y}^T \mathbf{K}_Y^{-1} \mathbf{k}_Y(\mathbf{x}) \tag{33}$$

$$\sigma_Y^2(\mathbf{x}) \;=\; k_Y(\mathbf{x}, \mathbf{x}) - \mathbf{k}_Y(\mathbf{x})^T \mathbf{K}_Y^{-1} \mathbf{k}_Y(\mathbf{x}). \tag{34}$$

The prediction variance is $\sigma_Y^2(\mathbf{x})$. The colour in the figure depicts the variance of the reconstructions; i.e., it is proportional to $-\ln \sigma_Y^2(\mathbf{x})$. This plot depicts the confidence with which the model reconstructs a pose as a function of latent position $\mathbf{x}$. The GPDM reconstructs the pose with high confidence in a "tube" around the region occupied by the training data.

To further illustrate the dynamical process, we can draw samples from the dynamic predictive distribution. As noted above, because we marginalize over the dynamic weights $\mathbf{A}$, the resulting density over latent trajectories is non-Gaussian. In particular, it cannot factored into a sequence of low-order Markov transitions (Fig. 1(a)). Hence, one cannot properly draw samples from the model in a causal fashion, one state at a time from a transition density, $p(\mathbf{x}_t^{(*)} \mid \mathbf{x}_{t-1}^{(*)})$.

Instead, we draw fair samples of entire trajectories using a Markov chain Monte Carlo sampler. The Markov chain was initialized with what we call a *mean-prediction sequence*, generated from $\mathbf{x}_1^{(*)}$ by simulating the dynamical process one frame at a time. That is, the density over $\mathbf{x}_t^{(*)}$ conditioned on $\mathbf{x}_{t-1}^{(*)}$ is Gaussian:

$$\mathbf{x}_t^{(*)} \sim \mathcal{N}(\mu_X(\mathbf{x}_{t-1}^{(*)}); \sigma_X^2(\mathbf{x}_{t-1}^{(*)})\mathbf{I}) \tag{35}$$

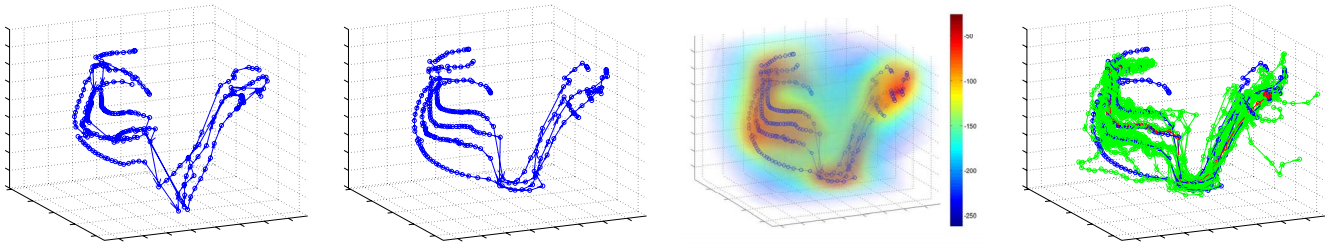
$$\mu_X(\mathbf{x}) = \mathbf{X}_{2:N}^T \mathbf{K}_X^{-1} \mathbf{k}_X(\mathbf{x}) \tag{36}$$

$$\sigma_X^2(\mathbf{x}) = k_X(\mathbf{x}, \mathbf{x}) - \mathbf{k}_X(\mathbf{x})^T \mathbf{K}_X^{-1} \mathbf{k}_X(\mathbf{x}), \tag{37}$$

where $\mathbf{k}_X(\mathbf{x})$ is a vector containing $k_X(\mathbf{x}, \mathbf{x}_i)$ in the i-th entry and $\mathbf{x}_i$ is the i^{th} training vector. At each step of mean-prediction, we set the latent position to be the mean latent position conditioned on the previous step: $\mathbf{x}_t^{(*)} = \mu_X(\mathbf{x}_{t-1}^{(*)})$.

Given an initial mean-prediction sequence, a Markov chain with several hundred samples is generated using hybrid Monte Carlo (HMC).[11] Figure 3(e) shows 23 fair samples from the latent dynamics of the GPDM. All samples are conditioned on the same initial state, $\mathbf{x}_1^{(*)}$, and each has a length of 62 time steps (i.e., drawn from $p(\mathbf{X}_{2:62}^{(*)} \mid \mathbf{x}_1^{(*)}, \Gamma)$). The length was chosen to be just less than a full gait cycle for ease of visualization. The resulting trajectories are smooth and roughly follow the trajectories of the training sequences. The variance in latent position tends to grow larger when the latent trajectories corresponding to the training data are farther apart, and toward the end of the simulated trajectory.

[11]We allow for 40 burn-in samples, and set the HMC parameters to obtain a rejection rate of about 20%.



(a) (b) (c) (d)

Fig. 4. Models learned from four golf swings from the same golfer. The latent coordinates learned with a GPLVM (a) and GPDM (b) are shown in blue. Vectors depict the temporal sequence. (c) $-\ln$ variance for reconstruction shows positions in latent space that are reconstructed with high confidence. (d) Random trajectories drawn from the dynamic predictive distribution using hybrid Monte Carlo are green, the red trajectory is the mean of the samples.

It is also of interest to see samples generated that are much longer than a gait cycle. Figure 3(f) shows one sample from an HMC sampler that is approximately four cycles in length. Notice that longer trajectories are also smooth, generating what look much like limit cycles in this case. To see why this process generates motions that look smooth and consistent, note that the variance of pose $\mathbf{x}_{t+1}^{(*)}$ is determined in part by $\sigma_X^2(\mathbf{x}_t^{(*)})$. This variance will be lower when $\mathbf{x}_t^{(*)}$ is nearer to other samples in the training data or the new sequence. As a consequence, the likelihood of $\mathbf{x}_{t+1}^{(*)}$ can be increased by moving $\mathbf{x}_t^{(*)}$ closer to latent positions of other poses in the model.

Figure 3(g-i) show a GPDM with only a linear term in the dynamics kernel (12). Here the dynamical model is not as expressive, and there is more process noise. Hence random samples from the dynamics do not follow the training data closely (Fig. 3(h)). The longer trajectories in Fig. 3(i) are attracted towards the origin.

B. Golf Swing Model

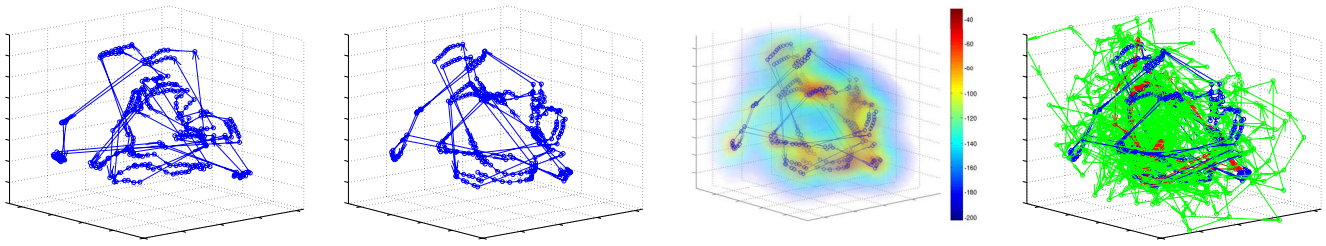
The GPDM can be applied to both cyclic motions (like walking above) and acyclic motions. Fig. 4 shows a GPDM learned from four swings of a golf club, all by the same subject.[12] Figures 4(a) and 4(b) show a 3D GPLVM and a 3D GPDM on the same golf data. The swings all contain periods of high acceleration; consequently, the spacing between points in latent space are more

[12]CMU database files 64_01.amc, frames 120 to 400; 64_02.amc, frames 170 to 420; 64_03.amc, frames 100 to 350; 64_04.amc, frames 80 to 315; all down-sampled by a factor of 4.

TABLE I

KERNEL PROPERTIES FOR FOUR-WALKER MODELS

	MAP	B-GPDM	Fixed α	Two-stage MAP
$SNR(\bar{\alpha})$	6.47	940	100	18.0
$CLS(\bar{\alpha})$	0.32	1.44	2.24	0.75
$SNR(\bar{\beta})$	34.0	5.23	9.32	45.0
$CLS(\bar{\beta})$	0.54	0.77	1.43	1.34



(a) (b) (c) (d)

Fig. 5. Models learned from walking sequences from four different subjects. The latent coordinates learned with a GPLVM (a) and GPDM (b) are shown in blue. (c) $-\ln$ variance plot shows clumpy high confidence regions. (d) Samples from the dynamic predictive distribution are shown in green, while the mean-prediction sample is shown in red. The samples do not stay close to the training data.

varied compared to the single walker data. While the GPLVM latent space contains an abrupt jump near the bottom of the figure, the GPDM is much smoother. Figure 4(c) shows the volume visualization, and 4(d) shows samples drawn from the dynamics predictive distribution.

While the GPDM learned with MAP estimation is better behaved than the GPLVM, an even smoother model can be learned using two-stage MAP. For example, Fig. 9(a,b) shows the GPDM learned with two-stage MAP. Random samples from its predictive dynamical model, shown in Fig. 9(c), nicely follow the training data and produce animations that are of visually higher quality than samples from the MAP model in Fig. 4(d).

C. Four-Walker Models

The MAP learning algorithm produces good models for the the single walker and the golf swings data. However, as discussed above, this is not the case with model learned with four

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

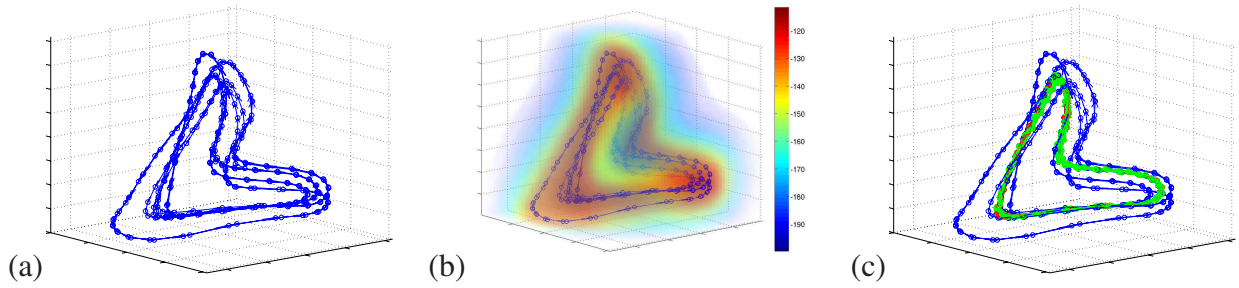


Fig. 6. B-GPDM models learned from walking sequences from three different subjects. (a) The learned latent coordinates shown in blue. (b) $-\ln$ variance plot shows smooth high confidence regions, but the variance near data is larger than in Figure 5(c). (c) Samples from the dynamic predictive distribution are shown in green, while the mean-prediction sample is shown in red.

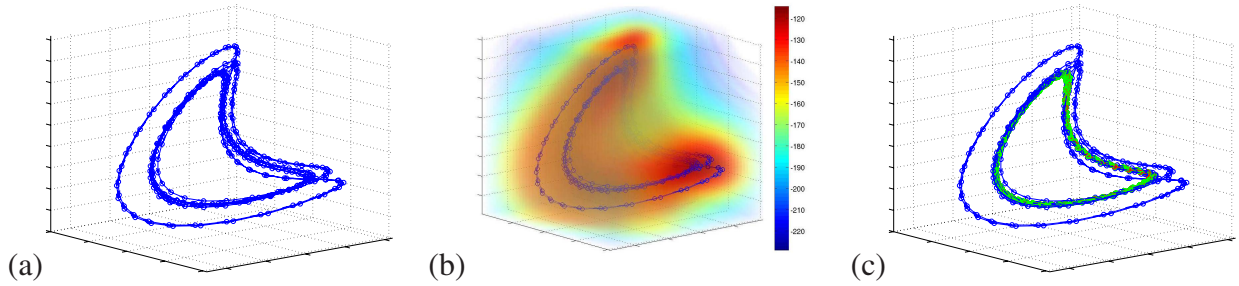


Fig. 7. Models learned with fixed $\bar{\alpha}$ from three different walking subjects. (a) The learned latent coordinates shown in blue. (b) $-\ln$ variance plot shows smooth high confidence regions, but the variance near data is larger than in Fig. 5(c), similar to B-GPDM. (c) Typical samples from the dynamic predictive distribution are shown in green, while the mean-prediction sample is shown in red.

walkers (Fig. 5(b)).[13] In contrast to the GPDM learned for the single walk data (Fig. 3), the latent positions for the training poses in the four-walker GPDM consist of small clumps of points connected by large jumps. The regions with high reconstruction certainty are similarly clumped (Fig. 5(c)); only in the vicinity of these clumps is pose reconstructed reliably. Also note that the latent positions estimated for the GPDM are very similar to those estimated by the GPLVM on the same dataset (Fig. 5(a)). This suggests that the dynamical term in the objective function (27) is overwhelmed by the data reconstruction term during learning, and therefore has a negligible impact on the resulting model.

[13]CMU database files 35_02.amc, frames 55 to 338; 10_04.amc, frames 222 to 499; 12_01.amc, frames 22 to 328; 16_15.amc, frames 62 to 342; all down-sampled by a factor of 4.

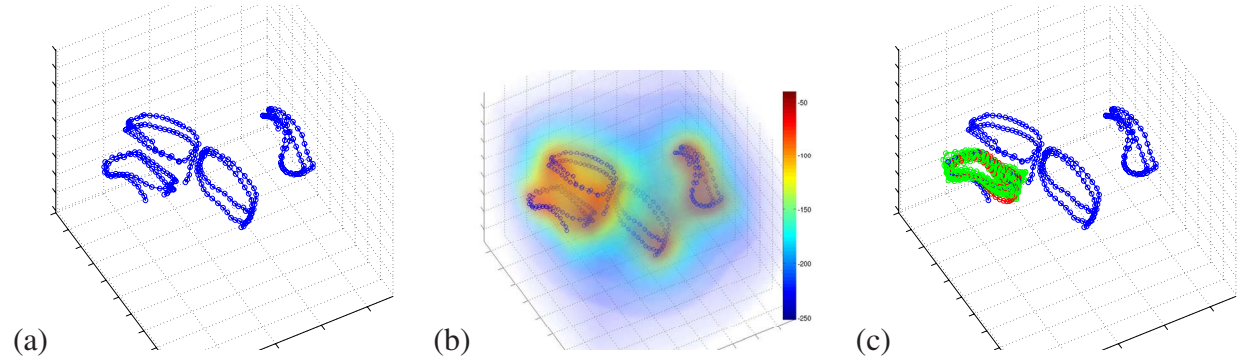


Fig. 8. Models learned with two-stage MAP from four different walking subjects. (a) The learned latent coordinates shown in blue, note the walkers are separated into distinct portions of the latent space. (b) $-\ln$ variance plot shows smooth high confidence regions, and the variance near data is similar to Fig. 5(c). (c) Typical samples from the dynamic predictive distribution are shown in green, while the mean-prediction sample is shown in red.

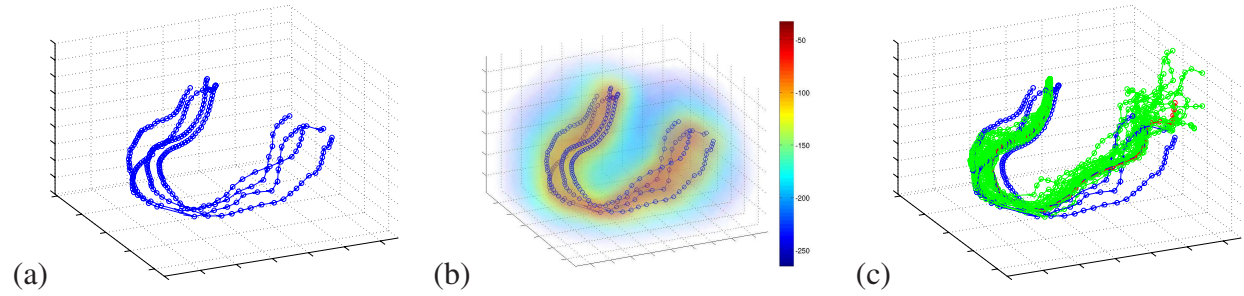


Fig. 9. Models learned with two-stage MAP from four golf swings from the same golfer. (a) The learned latent coordinates shown in blue. (b) $-\ln$ variance for reconstruction shows positions in latent space that are reconstructed with high confidence. (c) Random trajectories drawn from the dynamic predictive distribution using hybrid Monte Carlo are green, the red trajectory is the mean-prediction sample. The distribution is conditioned on starting from the beginning of a golf swing.

To better understand this GPDM, it is instructive to examine the estimated kernel hyperparameters. Following [11], Table V-C shows the signal-to-noise ratio (SNR) and the characteristic length scale (CLS) of the kernels. The SNR depends largely on the variance of the additive process noise. The CLS is defined as square root of the inverse RBF width, that is $CLS(\bar{\beta}) = \alpha_2^{-0.5}$ and $CLS(\bar{\alpha}) = \beta_1^{-0.5}$. The CLS is directly related to the smoothness of the mapping [11], [47]. Table V-C reveals that dynamical hyperparameters $\bar{\alpha}$ for the four-walker GPDM has both a low SNR and a low CLS. Not surprisingly, random trajectories the dynamical model (see Fig. 5(d)) show larger variability than any of the other three models shown. The trajectories do not stay close to regions of high reconstruction certainty, and therefore yield poor pose reconstructions and unrealistic walking motions; in particular, note the feet locations in the fourth pose from

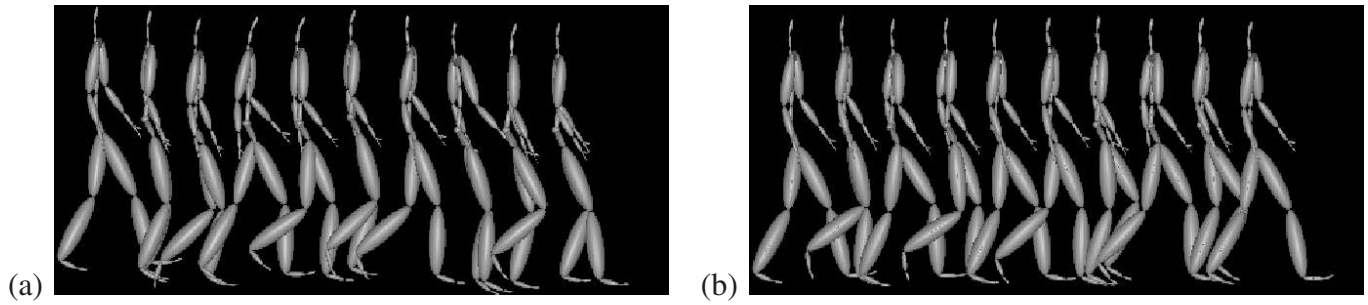


Fig. 10. Walks synthesized by taking the mean of the predictive distribution, conditioned on a starting point in latent space. (a) The walk produced by the MAP model is unrealistic, and does not resemble the training data. (b) High-quality walk produced by a model learned using two-stage MAP.

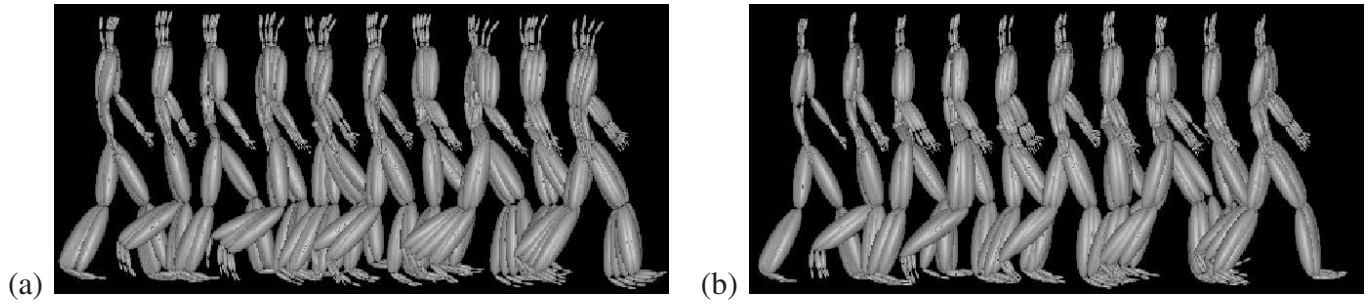


Fig. 11. Six walks synthesized by sampling from the predictive distribution, conditioned on a starting point in latent space. (a) Walks generated from B-GPDM. (b) Walks generated from a two-stage MAP model; note the difference in variance.

the left in Fig. 10(a).

Figure 6 shows the balanced GPDM (B-GPDM) learned from the four-walker data. Note that the learned trajectories are smooth and poses are not clumped. Sample trajectories from the model dynamics stay close to the training data (Fig. 6(c)). On the other hand, Fig. 6(b) shows that the B-GPDM exhibits higher reconstruction uncertainty near the training data (as compared to Fig. 5(c)). The emphasis on smoothness when learning the B-GPDM yields hyperparameters that give small variance to dynamical predictions, but large variance in pose reconstruction predictions. For example, in Table V-C note that the dynamics kernel $\bar{\alpha}$ has a high SNR of 940, while the SNR of reconstruction kernel $\bar{\beta}$ is only 5.23. Because of the high reconstruction variance, fair pose samples from the B-GPDM (20) are noisy and do not resemble realistic walks (see Fig. 11(a)). Nevertheless, unlike the MAP model, mean motions produced from the B-GPDM from different starting states usually correspond high-quality walking motions.

Figure 7 shows how a model learned with fixed hyperparameters $\bar{\alpha}$ (Section IV-C) also

produces smooth learned trajectories. The samples from the dynamics predictive distribution (see Fig. 7(c)) have low variance, staying close to the training data. Like the B-GPDM, the pose reconstructions have high variance (Fig. 7(b)). One can also infer this from the low SNR of 9.32 for the reconstruction kernel $\bar{\beta}$. Hence, like the B-GPDM, sample poses from random trajectories are noisy.

Figure 8 shows a model learned using two-stage MAP (Section IV-D). The latent trajectories are smooth, but not as smooth as those in Fig. 6 and 7. Notably, the walk cycles from the four subjects are separated in the latent space. Random samples from the latent dynamical model tend to stay near the training data (Fig. 8(c)), like the other smooth models.

In contrast to other smooth models, the hyperparameters $\bar{\beta}$ for the two-stage MAP model have a higher SNR of 45.0. One can see this with the reconstruction uncertainty near training data in Fig. 8(b)). On the other hand, the SNR for the dynamics kernel parameters $\bar{\alpha}$ is 18.0, lower than those those for the B-GPDM and the model learned with fixed $\bar{\alpha}$, but higher than that for the MAP model. Random samples generated from this model (e.g., Fig. 11(b)) have smaller variance and produce realistic walking motions.

We should stress here that the placement of the latent trajectories is not strictly a property of the learning algorithm. The B-GPDM, for example, does sometimes produce separate walk cycles when applied to other data sets [52]. We have observed the same behavior when $\bar{\alpha}$ are fixed at various settings to encourage smoothness. Conversely, the two-stage MAP model learned on the golf swing data does not separate the swings in the latent space (Fig. 9). One conclusion that can be made about the algorithms is on the smoothness of the individual trajectories. For the same data sets, models learned from MAP tend to have the least smooth trajectories, while models learned from B-GPDM and fixed $\bar{\alpha}$ tend to produce the smoothest trajectories. Models learned from two-stage MAP are somewhere in-between.

D. Missing Data

To further examine the models, we consider the task of filling in missing frames of new data using the four-walker models. We take 50 frames of new data, remove 31 frames in the middle, and attempt to recover the missing frames by optimizing (22).[14] We set $\mathbf{y}^{(*)} = \mu_Y\left(\mathbf{x}^{(*)}\right)$ for the

[14]The observed data roughly correspond to 1.5 cycles, of which nearly 1 cycle was missing.

TABLE II

MISSING FRAMES RMS ERRORS

	MAP	B-GPDM	Fix. α	T. MAP	KNN-15	Spline
35-03	58.06	12.15	16.32	20.74	17.88	62.98
12-02	48.51	37.06	30.95	26.60	33.99	67.35
16-21	72.28	32.87	73.46	53.13	47.26	100.10
12-03	57.46	40.40	26.00	23.16	30.89	64.27
07-01	84.16	65.38	42.41	68.69	75.28	90.83
07-02	85.58	64.47	56.70	64.43	65.93	93.45
08-01	87.77	70.05	114.86	72.61	90.57	139.75
08-02	97.15	72.11	102.12	90.80	83.14	128.01
AVG.	73.87	49.31	57.85	52.52	55.62	93.34

missing frames.

Table II compares the RMS error per frame of each of the learned models, the result of optimizing a K-nearest neighbour (KNN) least squares objective function on eight test sets,[15] as well as direct cubic spline interpolation in the pose space. Each error is an average of 12 experiments on different windows of missing frames (i.e., missing frames 5-35, 6-36, ... , 16-46). None of the test data were used for training, however the first four rows in Table II are test motions from the same subjects as were used to learn the models. The last four rows in Table II are for test motions of new subjects.

Other than spline interpolation, the unsmooth model learned from MAP performed the worst on average. As expected, test results on subjects whose motions were used to learn the models show significantly smaller errors than for test motions from subjects not seen in the training set. None of the models consistently perform well in the latter case.

The B-GPDM achieved the best average error with relatively small variability across sequences. One possible explanation is that models with high variance in the reconstruction process, such as the B-GPDM, do a better job at constraining poses far from the training data. This is consistent with results from related work that it can be used as a prior for human tracking[10], where observations are typically distinct from the training data.

Nevertheless, the visual quality of the animations do not necessarily correspond to the RMS

[15]We tried K = [3, 6, 9, 15, 20], with 15 giving the lowest average error.

error values. The two-stage MAP model tends to fill in missing data by pulling the corresponding latent coordinates close to one of the training walk cycles. Consequently, the resulting animation contains noticeable "jumps" when transitioning from observed test frames to missing test frames, especially for subjects not seen in the training set. Both the B-GPDM and models learned with fixed $\bar{\alpha}$ rely more on the smoothness of the latent trajectories to fill in the missing frames in latent space. As a result, the transitions from observed to missing frames tend to be smoother in the animation. For subjects not seen in the training set, the models with hand-specified $\bar{\alpha}$ tend to place all of the test data (both observed and missing) far from them training data in latent space. This amounts to filling in missing frames only using newly observed frames, which does not have enough information to produce high quality walks. Severe footskate is observed even in cases with small RMS errors (such as on data set 07-01).

VI. Discussion and Extensions

We have presented the Gaussian process dynamical model, a non-parametric model for high-dimensional dynamical systems that accounts for uncertainty in the model parameters. The model is applied to 50-dimensional motion capture data, and four learning algorithms are investigated. We showed that high-quality motions can be synthesized from the model without postprocessing, as long as the learned latent trajectories are reasonably smooth. The model defines a density function over new motions, which can be used to predict missing frames.

The smoothness of the latent trajectories and the corresponding inverse variance plots tell us a lot about the quality of the learned models. With the single walker data set for example, if the learned latent coordinates define a low-variance tube around the data, then new poses along the walk cycle (in phases not in the training data) can be reconstructed. This is not true if the latent space contains clumps of low-variance regions associated with an unsmooth trajectory. One of the main contributions of the GPDM is the ability to incorporate a soft smoothness constraint on the latent space for the family of GPLVMs.

In addition to the smoothness, the placement of latent trajectories is also informative. When trajectories are placed far apart in the latent space with no low-variance region between them, little or no structure between the trajectories is learned. That is not unreasonable, however, as observed in related work [58], the intra-trajectory distance between poses is often much smaller than the inter-trajectory distance. That is, it may well better reflect the data. The GPDM does not

explicitly constrain the placement of individual trajectories, and incorporating prior knowledge to enable the modeling of inter-trajectory structure is an interesting area of future work. One potential approach is adapting a mixture of GPs [59] to the latent variable model framework.

Performance is a major issue in applying GP methods to larger datasets. Previous approaches prune uninformative vectors from the training data [8]. This is not straightforward when learning a GPDM, however, because each timestep is highly correlated with the steps before and after it. For example, if we hold $\mathbf{x}_t$ fixed during optimization, then it is unlikely that the optimizer will make much adjustment to $\mathbf{x}_{t+1}$ or $\mathbf{x}_{t-1}$. The use of higher-order features provides a possible solution to this problem. Specifically, consider a dynamical model of the form $\mathbf{v}_t = f(\mathbf{x}_{t-1}, \mathbf{v}_{t-1})$. Since adjacent time-steps are related only by the velocity $\mathbf{v}_t \approx (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$, we can handle irregularly-sampled data points by adjusting the timestep Δt, possibly using a different Δt at each step. Another intriguing approach for speeding up the GPDM learning is through the use of pseudo-inputs [25], [60], [26].

A number of further extensions to the GPDM are possible. It would be straightforward to include an input signal $\mathbf{u}_t$ in the dynamics $f(\mathbf{x}_t, \mathbf{u}_t)$, which could potentially be incorporated into existing frameworks for GPs in reinforcement learning as a tool for model identification of system dynamics [61]. The use of a latent space in the GPDM may be particularly relevant for continuous problems with high-dimensional state-action spaces.

It would also be interesting to improve the MCEM algorithm used for two-stage MAP. The algorithm currently used is only a crude approximation and does not utilize samples efficiently. Methods such as ascent-based MCEM [62] can potentially be used to speed up the two-stage learning algorithm.

For applications in animation, animator constraints could be specified in pose space to synthesize entire motion sequences by constrained optimization. Such a system would be a generalization of the interactive posing application presented by Grochow et al. [6]. However the speed of the optimization would certainly be an issue, due to the dimensionality of the state space.

A more general direction of future work is the learning and inference of motion models from long, highly variable motion sequences such as a dance score. A latent variable representation of such sequences must contain a variety of loops and branches, which the current GPDM cannot learn regardless of performance issues. Modeling branching in latent space requires taking non-Gaussian process noise into account in the dynamics. Alternatively, one could imagine building

a hierarchical model, where a GPDM is learned on segment(s) of motion and connected through a higher level Markov model [7].

ACKNOWLEDGMENTS

An earlier version of this work appeared in [63]. The authors would like to thank Neil Lawrence for his comments on the manuscript, as well as his publicly available GPLVM code. JMW would like to thank Ryan Schmidt for assisting in producing the supplemental video. The volume rendering figures were generated using Joe Conti's code on www.mathworks.com. This project is funded in part by the Alfred P. Sloan Foundation, Canadian Institute for Advanced Research, Canada Foundation for Innovation, Microsoft Research, NSERC Canada, and Ontario Ministry of Research and Innovation. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

REFERENCES

[1] A. Elgammal and C.-S. Lee, "Inferring 3D body pose from silhouettes using activity manifold learning." in *Proc. IEEE CVPR'04*, vol. 2, Washington, DC, June/July 2004, pp. 681–688.

[2] N. R. Howe, M. E. Leventon, and W. T. Freeman, "Bayesian reconstruction of 3D human motion from single-camera video." in *Adv. Neural Info. Proc. Sys. 12*. The MIT Press, 2000, pp. 820–826, Proc. NIPS'99.

[3] H. Sidenbladh, M. J. Black, and D. J. Fleet, "Stochastic tracking of 3D human figures using 2D image motion," in *Comput. Vis. – ECCV'00*. Springer, 2000, vol. 2, pp. 702–718.

[4] C. Sminchisescu and A. D. Jepson, "Generative modeling for continuous non-linearly embedded visual inference." in *Proc. ICML'04*, Banff, Canada, July 2004, pp. 759–766.

[5] Y. Yacoob and M. J. Black, "Parameterized modeling and recognition of activities." *Comput. Vis. Image Understanding*, vol. 73, no. 2, pp. 232–247, Feb. 1999.

[6] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 522–531, Aug. 2004, Proc. SIGGRAPH.

[7] Y. Li, T. Wang, and H.-Y. Shum, "Motion texture: A two-level statistical model for character motion synthesis," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 465–472, July 2002, Proc. SIGGRAPH.

[8] N. D. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models," *J. Machine Learning Res.*, vol. 6, pp. 1783–1816, Nov. 2005.

[9] A. Rahimi, B. Recht, and T. Darrell, "Learning appearance manifolds from video." in *Proc. IEEE CVPR'05*, vol. 1, San Diego, CA, June 2005, pp. 868–875.

[10] R. Urtasun, D. J. Fleet, and P. Fua, "3D people tracking with Gaussian process dynamical models." in *Proc. IEEE CVPR'06*, vol. 1, New York, NY, June 2006, pp. 238–245.

[11] N. D. Lawrence, "The Gaussian process latent variable model." Department of Computer Science, University of Sheffield, Tech. Rep. CS-06-03, Jan. 2006.

[12] S. T. Roweis, "EM algorithms for PCA and SPCA." in *Adv. Neural Info. Proc. Sys. 10*. The MIT Press, 1998, pp. 626–632, Proc. NIPS'97.

[13] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *J. Roy. Stat. Soc. B*, vol. 61, no. 3, pp. 611–622, 1999.

[14] R. Bowden, "Learning statistical models of human motion." in *Proc. IEEE Workshop on Human Modeling, Analysis and Synthesis*, Hilton Head Island, SC, June 2000, pp. 10–17.

[15] M. Brand and A. Hertzmann, "Style machines," in *Proc. ACM SIGGRAPH'00*, New Orleans, LA, July 2000, pp. 183–192.

[16] L. Molina-Tanco and A. Hilton, "Realistic synthesis of novel human movements from a database of motion capture examples." in *Proc. HUMO'00*, Austin, TX, Dec. 2000, pp. 137–142.

[17] H. Murase and S. Nayar, "Visual learning and recognition of 3D objects from appearance," *Int. J. Comput. Vis.*, vol. 14, no. 1, pp. 5–24, Jan. 1995.

[18] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, December 2000.

[19] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation." *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, June 2003.

[20] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[21] V. de Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction." in *Adv. Neural Info. Proc. Sys. 15*. The MIT Press, 2003, pp. 705–712, Proc. NIPS'02.

[22] O. C. Jenkins and M. J. Matarić, "A spatio-temporal extension to Isomap nonlinear dimension reduction." in *Proc. ICML'04*, Banff, Canada, July 2004, pp. 441–448.

[23] R. Pless, "Image spaces and video trajectories: Using Isomap to explore video sequences." in *Proc. IEEE ICCV'03*, vol. 2, Nice, France, Oct. 2003, pp. 1433–1440.

[24] R. Urtasun, D. J. Fleet, A. Hertzmann, and P. Fua, "Priors for people tracking from small training sets." in *Proc. IEEE ICCV'05*, vol. 1, Beijing, China, Oct. 2005, pp. 403–410.

[25] N. D. Lawrence, "Learning for larger datasets with the Gaussian process latent variable model." in *Proc. AISTATS'07*, San Juan, Puerto Rico, Mar. 2007.

[26] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs." in *Adv. Neural Info. Proc. Sys. 18*. The MIT Press, 2006, pp. 1257–1264, Proc. NIPS'05.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.

[28] Z. Ghahramani and G. E. Hinton, "Parameter estimation for linear dynamical systems," Department of Computer Science, University of Toronto, Tech. Rep. CRG-TR-96-2, Feb. 1996.

[29] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *J. Time Series Anal.*, vol. 3, no. 4, pp. 253–264, 1982.

[30] P. Van Overschee and B. De Moor, "N4SID : Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, no. 1, pp. 75–93, Jan. 1994.

[31] G. A. Smith and A. J. Robinson, "A comparison between the EM and subspace identification algorithms for time-invariant linear dynamical systems." Cambridge University Engineering Department, Tech. Rep. CUED/F-INFENG/TR.345, Nov. 2000.

[32] A. Bissacco, "Modeling and learning contact dynamics in human motion." in *Proc. IEEE CVPR'05*, vol. 1, San Diego, CA, June 2005, pp. 421–428.

[33] S. M. Oh, J. M. Rehg, T. R. Balch, and F. Dellaert, "Learning and inference in parametric switching linear dynamical systems." in *Proc. IEEE ICCV'05*, vol. 2, Beijing, China, Oct. 2005, pp. 1161–1168.

[34] V. Pavlović, J. M. Rehg, and J. MacCormick, "Learning switching linear models of human motion." in *Adv. Neural Info. Proc. Sys. 13*. The MIT Press, 2001, pp. 981–987, Proc. NIPS'00.

[35] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives." in *Adv. Neural Info. Proc. Sys. 15*. The MIT Press, 2002, pp. 1523–1530, Proc. NIPS'01.

[36] S. T. Roweis and Z. Ghahramani, "Learning nonlinear dynamical systems using the expectation-maximization algorithm." in *Kalman Filtering and Neural Networks*. Wiley, 2001, pp. 175–220.

[37] D. Ormoneit, H. Sidenbladh, M. J. Black, and T. Hastie, "Learning and tracking cyclic human motion." in *Adv. Neural Info. Proc. Sys. 13*. The MIT Press, 2001, pp. 894–900, Proc. NIPS'00.

[38] R. Urtasun, D. J. Fleet, and P. Fua, "Temporal motion models for monocular and multiview 3D human body tracking," *Comput. Vis. Image Understanding*, vol. 104, no. 2, pp. 157–177, Nov. 2006.

[39] H. Sidenbladh, M. J. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking." in *Comput. Vis. – ECCV'02*. Springer, 2002, vol. 2, pp. 784–800.

[40] O. Arikan and D. A. Forsyth, "Interactive motion generation from examples," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 483–490, July 2002, Proc. SIGGRAPH.

[41] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 473–482, July 2002, Proc. SIGGRAPH.

[42] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard, "Interactive control of avatars animated with human motion data," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 491–500, July 2002, Proc. SIGGRAPH.

[43] T. Mukai and S. Kuriyama, "Geostatistical motion interpolation," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1062–1070, July 2005, Proc. SIGGRAPH.

[44] C. Rose, M. Cohen, and B. Bodenheimer, "Verbs and adverbs: Multidimensional motion interpolation." *IEEE Comput. Graph. Appl.*, vol. 18, no. 5, pp. 32–40, Sept./Oct. 1998.

[45] M. A. Giese and T. Poggio, "Morphable models for the analysis and synthesis of complex motion patterns," *Int. J. Comput. Vis.*, vol. 38, no. 1, pp. 59–73, June 2000.

[46] W. Ilg, G. H. Bakir, J. Mezger, and M. Giese, "On the representation, learning and transfer of spatio-temporal movement characteristics," *Int. J. Humanoid Robotics*, vol. 1, no. 4, pp. 613–636, Dec. 2004.

[47] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. New York, NY: Cambridge University Press, 2003.

[48] R. M. Neal, *Bayesian Learning for Neural Networks*. Secaucus, NJ: Springer-Verlag New York, Inc., 1996.

[49] K. Moon and V. Pavlović, "Impact of dynamics on subspace embedding and tracking of sequences." in *Proc. IEEE CVPR'06*, vol. 1, New York, NY, June 2006, pp. 198–205.

[50] R. Murray-Smith and B. A. Pearlmutter, "Transformations of Gaussian process priors." in *Deterministic and Statistical Methods in Machine Learning*. Springer, 2005, pp. 110–123, Proc. Int. Workshop.

[51] E. Solak, R. Murray-Smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen, "Derivative observations in Gaussian process models of dynamic systems." in *Adv. Neural Info. Proc. Sys. 15*. The MIT Press, 2003, pp. 1033–1040, Proc. NIPS'02.

[52] R. Urtasun, "Motion models for robust 3D human body tracking," Ph.D. dissertation, EPFL, Switzerland, 2006.

[53] A. Ogawa, K. Takeda, and F. Itakura, "Balancing acoustic and linguistic probabilities." in *IEEE ICASSP'98*, vol. 1, Seattle, WA, May 1998, pp. 181–184.

[54] A. Rubio, J. Diaz-Verdejo, and J. S. P. Garcia, "On the influence of frame-asynchronous grammar scoring in a CSR system." in *IEEE ICASSP'97*, vol. 1, Munich, Germany, Apr. 1997, pp. 895–898.

[55] D. H. Brainard and W. T. Freeman, "Bayesian color constancy," *J. Optical Soc. America, A*, vol. 14, no. 7, pp. 1393–1411, July 1997.

[56] R. M. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*. The MIT Press, 1999, pp. 355–368.

[57] G. Wei and M. Tanner, "A Monte-Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms." *J. American Stat. Assoc.*, vol. 85, no. 411, pp. 699–704, 1990.

[58] A. Elgammal and C.-S. Lee, "Separating style and content on a nonlinear manifold." in *Proc. IEEE CVPR'04*, vol. 1, Washington, DC, June/July 2004, pp. 478–485.

[59] J. Q. Shi, R. Murray-Smith, and D. M. Titterington, "Hierarchical Gaussian process mixtures for regression," *Statistics and Computing*, vol. 15, pp. 31–41, 2005.

[60] N. D. Lawrence and J. Q. Candela, "Local distance preservation in the GP-LVM through back constraints." in *Proc. ICML'06*, Pittsburgh, PA, June 2006, pp. 513–520.

[61] C. E. Rasmussen and M. Kuss, "Gaussian processes in reinforcement learning." in *Adv. Neural Info. Proc. Sys. 16*. The MIT Press, 2004, pp. 751–759, Proc. NIPS'03.

[62] B. S. Caffo, W. Jank, and G. L. Jones, "Ascent-based Monte Carlo expectation-maximization," *J. Roy. Stat. Soc. B*, vol. 67, no. 2, pp. 235–251, Apr. 2005.

[63] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models." in *Adv. Neural Info. Proc. Sys. 18*. The MIT Press, 2006, pp. 1441–1448, Proc. NIPS'05.