To appear at UAI2014

Bayesian Filtering with Online Gaussian Process Latent Variable Models

Yali Wang
Laval University
yali.wang.1@ulaval.ca

Marcus A. Brubaker
TTI Chicago
mbrubake@cs.toronto.edu

Brahim Chaib-draa
Laval University
chaib@ift.ulaval.ca

Raquel Urtasun
University of Toronto
urtasun@cs.toronto.edu

Abstract

In this paper we present a novel non-parametric approach to Bayesian filtering, where the prediction and observation models are learned in an online fashion. Our approach is able to handle multimodal distributions over both models by employing a mixture model representation with Gaussian Processes (GP) based components. To cope with the increasing complexity of the estimation process, we explore two computationally efficient GP variants, sparse online GP and local GP, which help to manage computation requirements for each mixture component. Our experiments demonstrate that our approach can track human motion much more accurately than existing approaches that learn the prediction and observation models offline and do not update these models with the incoming data stream.

1 INTRODUCTION

Many real world problems involve high dimensional data. In this paper we are interested in *modeling* and *tracking* human motion. In this setting, dimensionality reduction techniques are widely employed to avoid the curse of dimensionality.

Linear approaches such as principle component analysis (PCA) are very popular as they are simple to use. However, they often fail to capture complex dependencies due to their assumption of linearity. Non-linear dimensionality reduction techniques that attempt to preserve the local structure of the manifold (*e.g.*, Isomap [21, 8], LLE [19, 14]) can capture more complex dependencies, but often suffer when the manifold assumptions are violated, *e.g.*, in the presence of noise.

Probabilistic latent variable models have the advantage of being able to take the uncertainties into account when learning the latent representations. Perhaps the most successful model in the context of modelling human motion is the Gaussian process latent variable model (GPLVM) [12], where the non-linear mapping between the latent space and the high dimensional space is modeled with a Gaussian process. This provides powerful prior models, which have been employed for character animation [28, 26, 15] and human body tracking [24, 16, 25].

In the context of tracking, one is interested in estimating the state of a dynamic system. The most commonly used technique for state estimation is Bayesian filtering, which recursively estimates the posterior probability of the state of the system. The two key components in the filter are the prediction model, which describes the temporal evolution of the process, as well as the observation model which links the state and the observations. A parametric form is typically employed for both models.

Ko and Fox [10] introduced the GP-BayesFilter, which defines the prediction and observation models in a non-parametric way via Gaussian processes. This approach is well suited when accurate parametric models are difficult to obtain. Its main limitation, however, resides in the fact that it requires ground truth states (as GPs are supervised), which are typically not available. GPLVMs were employed in [11] to learn the latent space in an unsupervised manner, bypassing the need for labeled data. This, however, can not exploit the incoming stream of data available in the online setting as the latent space is learned offline. Furthermore, only unimodal prediction and observation models can be captured due to the fact that the models learned by GP are nonlinear but Gaussian.

In this paper we extend the previous non-parametric filters to learn the latent space in an online fashion as well as to handle multimodal distributions for both the prediction and observation models. Towards this goal, we employ a mixture model representation in the particle filtering framework. For the mixture components, we investigate two computationally efficient GP variants which can update the prediction and observation models in an online fashion, and cope with the growth in complexity as the number of data points increases over time. More specifically, the sparse online GP [3] selects the active set in a online fashion to efficiently maintain sparse approximations to the models. Alternatively, the local GP [26] reduces the computation by imposing local sparsity.

We demonstrate the effectiveness of our approach on a wide variety of motions, and show that both approaches perform better than existing algorithms. In the remainder of the paper we first present a review on Bayesian filtering and the GPLVM. We then introduce our algorithm and show our experimental evaluation followed by the conclusions.

2 BACKGROUND

In this section we review Bayesian filtering and Gaussian process latent variable models.

2.1 BAYESIAN FILTERING

Bayesian filtering is a sequential inference technique typically employed to perform state estimation in dynamic systems. Specifically, the goal is to recursively compute the posterior distribution of the current hidden state $\mathbf{x}_t$ given the history of observations $\mathbf{y}_{1:t} = (\mathbf{y}_1, \ldots, \mathbf{y}_t)$ up to the current time step

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}$$

where $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the *prediction model* that represents the system dynamics, and $p(\mathbf{y}_t|\mathbf{x}_t)$ is the *observation model* that represents the likelihood of an observation $\mathbf{y}_t$ given the state $\mathbf{x}_t$.

One of the most fundamental Bayesian filters is the Kalman filter, which is a maximum-a-posteriori estimator for linear and Gaussian models. Unfortunately, it is often not applicable in practice since most real dynamical systems are non-linear and/or non-Gaussian. Two popular extensions for non-linear systems are the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) [9]. However, similar to the Kalman filter, the performance of EKF and UKF is poor when the models are multimodal [5].

In contrast, particle filters that are not restricted to linear and Gaussian models have been developed by using sequential Monte Carlo sampling to represent the underlying posterior $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ [5]. More specifically, at each time step, N_p particles of $\mathbf{x}_t$ are drawn from the prediction model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and then all the particles are weighted according to the observation model $p(\mathbf{y}_t|\mathbf{x}_t)$. The posterior $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is approximated using these N_p weighted particles. Finally, the N_p particles are resampled for the next step. Unfortunately, the parametric description of the dynamic models limits the estimation accuracy of Bayesian filters.

Recently, a number of GP-based Bayesian filters were proposed by learning the prediction and observation models using GP regression [10, 4]. This is a promising alternative as GPs are non-parametric and can capture complex mappings. However, training these methods requires access to ground truth data before filtering. Unfortunately, the inputs of the training set are the hidden states which are not always known in real-world applications. Two extensions were introduced to learn the hidden states of the training set via a non-linear latent variable model [11] or a sparse pseudo-input GP regression [22]. However, these methods require offline learning procedures, which are not able to exploit the incoming data streams. In contrast, we propose two non-parametric particle filters that are able to exploit the incoming data to learn better models in an online fashion.

2.2 GAUSSIAN PROCESS DYNAMICAL MODEL

The Gaussian Process Latent Variable Model (GPLVM) is a probabilistic dimensionality reduction technique, which places a GP prior on the observation model [12]. Wang et al. [28] proposed the Gaussian Process Dynamical Model (GPDM), which enriches the GPLVM to capture temporal structure by incorporating a GP prior over the dynamics in the latent space. Formally, the model is:

$$\begin{aligned} \mathbf{x}_t &= f_{\mathbf{x}}(\mathbf{x}_{t-1}) + \eta_{\mathbf{x}} \\ \mathbf{y}_t &= f_{\mathbf{y}}(\mathbf{x}_t) + \eta_{\mathbf{y}} \end{aligned}$$

where $\mathbf{y} \in \mathbb{R}^{D_\mathbf{y}}$ represents the observation and $\mathbf{x} \in \mathbb{R}^{D_\mathbf{x}}$ the latent state, with $D_\mathbf{y} \gg D_\mathbf{x}$. The noise processes are assumed to be Gaussian $\eta_\mathbf{x} \sim \mathcal{N}(0, \sigma_\mathbf{x}^2 I)$ and $\eta_\mathbf{y} \sim \mathcal{N}(0, \sigma_\mathbf{y}^2 I)$. The nonlinear functions $f_\mathbf{x}^i$ and $f_\mathbf{y}^i$ have GP priors, *i.e.*, $f_\mathbf{x}^i \sim \mathcal{GP}(0, k_x(\mathbf{x}, \mathbf{x}'))$ and $f_\mathbf{y}^i \sim \mathcal{GP}(0, k_y(\mathbf{x}, \mathbf{x}'))$ where $k_x(\cdot, \cdot)$ and $k_y(\cdot, \cdot)$ are the kernel functions. For simplicity, we denote the hyperparameters of the kernel functions by θ.

Let $\mathbf{x}_{1:T_0} = (\mathbf{x}_1, \cdots, \mathbf{x}_{T_0})$ be the latent space coordinates from time $t = 1$ to time $t = T_0$. GPDM is typically learned by minimizing the negative log posterior $-\log(p(\mathbf{x}_{1:T_0}, \theta|\mathbf{y}_{1:T_0}))$ with respect to $\mathbf{x}_{1:T_0}$, and θ [28]. After $\mathbf{x}_{1:T_0}$ and θ are obtained, a standard GP prediction is used to construct the model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \theta, \mathcal{X}_{T_0})$ and $p(\mathbf{y}_t|\mathbf{x}_t, \theta, \mathcal{Y}_{T_0})$ with data $\mathcal{X}_{T_0} = \{(\mathbf{x}_{k-1}, \mathbf{x}_k)\}_{k=2}^{T_0}$ and $\mathcal{Y}_{T_0} = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{T_0}$. Tracking ($t > T_0$) is then performed assuming the model is fixed and can be done using, *e.g.*, a particle filter as described above. The major drawback of this approach is that it is not able to adapt to new observations during tracking. As shown in our experimental evaluation, this results in poor performance when the training set is small.

3 ONLINE GP PARTICLE FILTER

In order to solve the above-mentioned difficulties in learning and filtering with dynamic systems, we propose an *Online GP Particle Filter* framework to learn and refine the model during tracking, *i.e.*, the prediction $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and observation $p(\mathbf{y}_t|\mathbf{x}_t)$ models are updated online in the particle filtering framework. To account for multi-modality and the significant amount of uncertainty that can be present, we propose to represent the prediction and observation models by a mixture model. For each mixture component, we will investigate two different GP variants.

Let the prediction and observation models at $t-1$ be

$$p(\mathbf{x}_t|\mathbf{x}_{t-1},\Theta_{t-1,M}) = \tfrac{1}{R_M}\textstyle\sum_{i=1}^{R_M} p(\mathbf{x}_t|\mathbf{x}_{t-1},\Theta^i_{t-1,M}) \quad (1)$$

$$p(\mathbf{y}_t|\mathbf{x}_t,\Theta_{t-1,O}) = \tfrac{1}{R_O}\textstyle\sum_{i=1}^{R_O} p(\mathbf{y}_t|\mathbf{x}_t,\Theta^i_{t-1,O}) \quad (2)$$

where $\Theta^i_{t-1,M}$ and $\Theta^i_{t-1,O}$ represents the parameters of the i-th component, $\Theta_{t-1,M} = \{\Theta^i_{t-1,M}\}_{i=1}^{R_M}$ and $\Theta_{t-1,O} = \{\Theta^i_{t-1,O}\}_{i=1}^{R_O}$ are the parameters of all components. At the t-th time step, we run a standard particle filter to obtain a number of weighted particles. The latent space representations at time t can be obtained by resampling the weighted particles. Then, we assign each particle to the most likely mixture component of $p(\mathbf{x}_t|\mathbf{x}_{t-1},\Theta_{t-1,M})$ and $p(\mathbf{y}_t|\mathbf{x}_t,\Theta_{t-1,O})$ to capture the multi-modality of the prediction and observation models. Finally, we compute the mean latent states of the assigned particles and use this mean state to update the corresponding components parameters, $\Theta^i_{t,M}$ for the prediction (or motion) model and $\Theta^i_{t,O}$ for the observation model. The whole framework is summarized in Algorithm 1.

What remains is to specify how the parameters of individual components are represented and updated (lines 18 and 23 in Algorithm 1). As noted above, we aim to use a GP model for each mixture component. However, a standard implementation would require $O(t^3)$ operations and $O(t^2)$ memory. As t grows linearly over time, the particle filter will quickly become too computationally and memory intensive. Thus a primary challenge is how to efficiently update the GP mixture components in the prediction and observation models.

In order to efficiently update $\Theta^i_{t,M}$ and $\Theta^i_{t,O}$ in an online manner, we consider two fast GP-based strategies: Sparse Online GP (SOGP) and Local GPs (LGP) in which the reduction in memory and/or computation is achieved by an online sparsification and a local experts mechanism respectively. A detailed review of fast GP approaches can be found in [1, 18].

The specific contents of $\Theta^i_{t,M}$ or $\Theta^i_{t,O}$ will vary depending on the method used. In the case of SOGP it will contain some computed quantities and the active set while for LGP it will simply be the set of all training points. While we will focus on these two strategies, we note that in principle any similar update strategy could be used instead, such as informative vector machines [13] or local regression approaches [7, 6, 20]. In what follows, to avoid confusion with the notations of the latent state and observation, we will use $\mathbf{a}$ and $\mathbf{b}$ to indicate the input and output when we describe SOGP and LGP regression in which we consider modeling a generic function $\mathbf{b} = f(\mathbf{a}) + \xi$, with $\xi \sim \mathcal{N}(0, \sigma^2 I)$.

Algorithm 1 Online GP-Particle Filter

```
1:  Initialize model parameters $\Theta$ based on $\mathbf{y}_{1:T_0}$
2:  Initialize particle set $\mathbf{x}_{T_0}^{(1:N_P)}$ based on $\mathbf{y}_{1:T_0}$
3:  for $t = T_0 + 1$ to $T$ do
4:      for $i = 1$ to $N_p$ do
5:          $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \Theta_{t-1,M})$
6:          $\hat{w}_t^{(i)} = p(\mathbf{y}_t|\mathbf{x}_t^{(i)}, \Theta_{t-1,O})$
7:      end for
8:      Normalize weights $w_t^{(i)} = \hat{w}_t^{(i)}/(\sum_{i=1}^{N_p}\hat{w}_t^{(i)})$
9:      Resample particle set with probabilities $w_t^{(1:N_p)}$
10:     for $i = 1$ to $N_p$ do
11:         $\eta_M^i = \arg\max_j p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, \Theta^j_{t-1,M})$
12:         $\eta_O^i = \arg\max_j p(\mathbf{y}_t|\mathbf{x}_t^{(i)}, \Theta^j_{t-1,O})$
13:     end for
14:     for $j = 1$ to $R_M$ do
15:         $n^j_{t-1} = \sum_{i=1}^{N_p}\delta(\eta_M^i = j)$
16:         $\bar{\mathbf{x}}^j_{t-1} = \frac{1}{n^j_{t-1}}\sum_{i=1}^{N_p}\delta(\eta_M^i = j)\mathbf{x}_{t-1}^{(i)}$
17:         $\bar{\mathbf{x}}^j_t = \frac{1}{n^j_{t-1}}\sum_{i=1}^{N_p}\delta(\eta_M^i = j)\mathbf{x}_t^{(i)}$
18:         Update $\Theta^j_{t,M}$ with $(\bar{\mathbf{x}}^j_{t-1}, \bar{\mathbf{x}}^j_t)$
19:     end for
20:     for $j = 1$ to $R_O$ do
21:         $n^j_{t-1} = \sum_{i=1}^{N_p}\delta(\eta_O^i = j)$
22:         $\bar{\mathbf{x}}^j_t = \frac{1}{n^j_{t-1}}\sum_{i=1}^{N_p}\delta(\eta_O^i = j)\mathbf{x}_t^{(i)}$
23:         Update $\Theta^j_{t,O}$ with $(\bar{\mathbf{x}}^j_t, \mathbf{y}_t)$
24:     end for
25: end for
```

3.1 SPARSE ONLINE GAUSSIAN PROCESS

The Sparse Online Gaussian Process (SOGP) of [3, 27] is a well-known algorithm for online learning of GP models. To cope with the fact that data arrives in an online manner, SOGP trains a GP model sequentially by updating the posterior mean and covariance of the latent function values of the training set. This online procedure is coupled with a sparsification strategy which iteratively selects a fixed-size subset of points to form the active set, preventing the otherwise unbounded growth of the computation and memory load.

The key of SOGP is to maintain the joint posterior over the latent function values of the fixed-size active set $\mathcal{D}_{t-1}$, i.e.,

$\mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$, by recursively updating μ_{t-1} and Σ_{t-1}. When a new observation $(\mathbf{a}_t, \mathbf{b}_t)$ [1] is available, we perform the following update to take the new data point into account [27]

$$
\begin{aligned}
\mathbf{q}_t &= Q_{t-1}\mathbf{k}_{t-1}(\mathbf{a}_t) && (3)\\
\rho_t^2 &= k(\mathbf{a}_t, \mathbf{a}_t) - \mathbf{k}_{t-1}(\mathbf{a}_t)^T Q_{t-1}\mathbf{k}_{t-1}(\mathbf{a}_t) && (4)\\
\hat{\sigma}_t^2 &= \sigma^2 + \rho_t^2 + \mathbf{q}_t^T \Sigma_{t-1}\mathbf{q}_t && (5)\\
\delta_t &= \begin{bmatrix} \Sigma_{t-1}\mathbf{q}_t \\ \rho_t^2 + \mathbf{q}_t^T\Sigma_{t-1}\mathbf{q}_t \end{bmatrix} && (6)\\
\mu_t &= \begin{bmatrix} \mu_{t-1} \\ \mathbf{q}_t^T \mu_{t-1} \end{bmatrix} + \hat{\sigma}_t^{-2}(\mathbf{b}_t - \mathbf{q}_t^T \mu_{t-1})\delta_t && (7)\\
\Sigma_t &= \begin{bmatrix} \Sigma_{t-1} & \Sigma_{t-1}\mathbf{q}_t \\ \mathbf{q}_t^T\Sigma_{t-1} & \rho_t^2 + \mathbf{q}_t^T\Sigma_{t-1}\mathbf{q}_t \end{bmatrix} - \hat{\sigma}_t^{-2}\delta_t\delta_t^T && (8)
\end{aligned}
$$

where $\mathbf{k}_{t-1}(\mathbf{a}_t)$ is the kernel vector which is constructed from $\mathbf{a}_t$ and the active set $\mathcal{D}_{t-1}$, and Q_{t-1} is the inverse kernel matrix of the active set $\mathcal{D}_{t-1}$.

One of the key steps in this algorithm is how to decide when to add the new point to the active set. We employed the strategy suggested by [3, 27], and ignore the new point with $\rho_t^2 < \epsilon$ for some small value of ϵ (we use $\epsilon = 10^{-6}$). In this case, the μ_t, Σ_t are updated as $\mu_t \leftarrow [\mu_t]_{-i}$, $\Sigma_t \leftarrow [\Sigma_t]_{-i,-i}$ where $i = t$ is the index of the new point, $[\cdot]_{-i}$ removes the i-th entry of a vector, and $[\cdot]_{-i,-i}$ removes the i-th row and column of a matrix. Additionally, the inverse kernel matrix is simply $Q_t = Q_{t-1}$ because the new point is not included in the active set. When $\rho_t^2 \geq \epsilon$, we add the new point to the active set $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{a}_t, \mathbf{b}_t)\}$. The μ_t, Σ_t are then the same as Eq.$\tilde{}$(7) and (8), and the inverse kernel matrix is updated to be [27]

$$
Q_t = \begin{bmatrix} Q_{t-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + \rho_t^{-2} \begin{bmatrix} \mathbf{q}_t\mathbf{q}_t^T & -\mathbf{q}_t \\ -\mathbf{q}_t^T & 1 \end{bmatrix} \qquad (9)
$$

When the size of the active set is larger than the fixed size N_A because a new point was added, we must remove a point. This is done by selecting the one which minimally affects the predictions according to the squared predicted error. Following [3, 27], we remove the j-th data point with

$$
j = \arg\min_j \left(\frac{[Q_t\mu_t]_j}{[Q_t]_{j,j}} \right)^2 \qquad (10)
$$

where $[\cdot]_j$ selects the j-th entry of a vector and $[\cdot]_{j,j}$ select the jth diagonal entry of a matrix. Once a point has been selected for removal, μ_t, Σ_t and Q_t are updated as

$$
\begin{aligned}
\mu_t &\leftarrow [\mu_t]_{-j} && (11)\\
\Sigma_t &\leftarrow [\Sigma_t]_{-j,-j} && (12)\\
Q_t &\leftarrow [Q_t]_{-j,-j} - \frac{[Q_t]_{-j,j}[Q_t]_{-j,j}^T}{[Q_t]_{j,j}} && (13)
\end{aligned}
$$

where $[\cdot]_{-j,j}$ selects the j-th column of the matrix with the j-th row removed and the point is removed from the active set $\mathcal{D}_t \leftarrow \mathcal{D}_t \setminus \{(\mathbf{a}_j, \mathbf{b}_j)\}$.

Algorithm 2 SOGP Update

```
input Previous posterior quantities $\mu_{t-1}$, $\Sigma_{t-1}$, $Q_{t-1}$
input Previous active set $\mathcal{D}_{t-1}$
input New input-output observation $(\mathbf{a}_t, \mathbf{b}_t)$ pair
Compute $\rho_t$, $\mu_t$ and $\Sigma_t$ as in Equations (4), (7) and (8).
if $\rho_t^2 < \epsilon$ then
    Perform update $\mu_t \leftarrow [\mu_t]_{-i}$, $\Sigma_t \leftarrow [\Sigma_t]_{-i,-i}$ where
    $i$ is the index of the newly added row to $\mu_t$.
    Set $Q_t = Q_{t-1}$, $\mathcal{D}_t = \mathcal{D}_{t-1}$.
else
    Compute $Q_t$ as in Equation (9).
    Add to active set $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{a}_t, \mathbf{b}_t)\}$.
end if
if $|\mathcal{D}_t| > N_A$ then
    Select point $j$ to remove using Equation (10).
    Perform update $\mu_t \leftarrow [\mu_t]_{-j}$, $\Sigma_t \leftarrow [\Sigma_t]_{-j,-j}$ and
    $Q_t \leftarrow [Q_t]_{-j,-j} - \frac{[Q_t]_{-j,j}[Q_t]_{-j,j}^T}{[Q_t]_{j,j}}$.
    Remove $j$ from active set $\mathcal{D}_t \leftarrow \mathcal{D}_t \setminus \{(\mathbf{a}_j, \mathbf{b}_j)\}$.
end if
output $\mu_t$, $\Sigma_t$, $Q_t$ and $\mathcal{D}_t$
```

The joint posterior at time t can be used to construct the predictive distribution for a new input $\mathbf{a}^*$

$$
p^{SOGP}(\mathbf{b}|\mathbf{a}^*, \mathcal{D}_t, \Theta) = \mathcal{N}(\mathbf{b}|\tilde{\mathbf{b}}, \tilde{\sigma}^2) \qquad (14)
$$

where $\tilde{\mathbf{b}} = \mathbf{k}_t(\mathbf{a}^*)^T Q_t \mu_t$ and $\tilde{\sigma}^2 = \sigma^2 + k(\mathbf{a}^*, \mathbf{a}^*) + \mathbf{k}_t(\mathbf{a}^*)^T(Q_t\Sigma_tQ_t - Q_t)\mathbf{k}_t(\mathbf{a}^*)$. We summarize the SOGP updates in Algorithm 2.

3.2 LOCAL GAUSSIAN PROCESSES

An alternative to the SOGP approach is to use Local Gaussian Processes (LGP), which was developed specifically to deal with large, multi-modal regression problems [17, 23]. In LGP, given a test input $\mathbf{a}^*$ and a set of input-output pairs $\mathcal{D} = \{(\mathbf{a}_i, \mathbf{b}_i)\}_{i=1}^N$, the $M_{\mathbf{a}}$-nearest neighbors $\mathcal{D}_{\mathbf{a}^*} = \{(\mathbf{a}_\ell, \mathbf{b}_\ell)\}_{\ell=1}^{M_{\mathbf{a}}}$ are selected based on the distance in the input space $d_\ell = \|\mathbf{a}_\ell - \mathbf{a}^*\|$. Then, for each of the $M_{\mathbf{a}}$ neighbors, $M_{\mathbf{b}}$-nearest neighbors $\mathcal{D}_{\mathbf{b}_\ell} = \{(\mathbf{a}_j, \mathbf{b}_j)\}_{j=1}^{M_{\mathbf{b}}}$ are selected based on the distance in the output space to $\mathbf{b}_\ell$. These neighbors are then combined to form a local GP expert which makes a Gaussian prediction with mean and covariance

$$
\begin{aligned}
\mu_\ell &= B_{\mathcal{D}_{\mathbf{b}_\ell}} K^{-1}_{\mathcal{D}_{\mathbf{b}_\ell},\mathcal{D}_{\mathbf{b}_\ell}} \mathbf{k}_{\mathcal{D}_{\mathbf{b}_\ell}}(\mathbf{a}^*)\\
\sigma_\ell^2 &= k(\mathbf{a}^*, \mathbf{a}^*) - \mathbf{k}_{\mathcal{D}_{\mathbf{b}_\ell}}(\mathbf{a}^*)^T K^{-1}_{\mathcal{D}_{\mathbf{b}_\ell},\mathcal{D}_{\mathbf{b}_\ell}} \mathbf{k}_{\mathcal{D}_{\mathbf{b}_\ell}}(\mathbf{a}^*) + \sigma^2
\end{aligned}
$$

where $B_{\mathcal{D}_{\mathbf{b}_\ell}}$ is the matrix whose columns are the $M_{\mathbf{b}}$ nearest neighbors of $\mathbf{b}_\ell$, $\mathbf{k}_{\mathcal{D}_{\mathbf{b}_\ell}}(\mathbf{a}^*)$ is the vector of kernel function values for the input $\mathbf{a}^*$ and the points in $\mathcal{D}_{\mathbf{b}_\ell}$, and

[1] For simplicity of presentation, we assume that $\mathbf{b}$ is a scalar. The extension to vector valued $\mathbf{b}$ is straightforward.

$K_{\mathcal{D}_{\mathbf{b}_\ell},\mathcal{D}_{\mathbf{b}_\ell}}$ is the kernel matrix for the points in $\mathcal{D}_{\mathbf{b}_\ell}$. The final predictive distribution is then formed by combining all local experts in a mixture model

$$p^{LGP}(\mathbf{b}|\mathbf{a}^*, \mathcal{D}, \Theta) = \sum_{\ell=1}^{M_\mathbf{a}} w_\ell \mathcal{N}(\mathbf{b}|\mu_\ell, \sigma_\ell^2 I) \qquad (15)$$

with weights $w_\ell \propto 1/d_\ell$.

4 EXPERIMENTAL EVALUATION

To illustrate our approach we choose 4 very different motions, *i.e.*, walking, golf swing, swimming as well as exercises (composed of side twist and squat). The data consists of motion capture from the CMU dataset [2], where each observation is a 62 dimensional vector containing the 3D rotations of all joints. We normalize the data to be mean zero and subsample the observations to reduce the correlation between consecutive frames. We use a frequency of 12 frames/s for walking and swimming, 24 frames/s for golf swing and 30 frames/s for the exercise motion. We compute all results averaged over 3 trials and report the average root mean squared error as our measure of performance.

In all the experiments, the latent space dimensionality is set to be 3 as is common for human motion models [28]. We use PCA to initialize the latent space and K-means to obtain the data points used for the mixture components. We choose the compound kernel function $k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp(-0.5 \parallel \mathbf{x} - \mathbf{x}' \parallel^2 / \gamma^2) + l^2 \mathbf{x}^T \mathbf{x}'$ for both prediction and observation mappings. Unless otherwise stated, we use 50 particles, a training set of size of 20/30/50/450 and 2/2/5/5 mixture components for walking/golf/swimming/exercise motions respectively. For LGP, the number of local GP experts is 2/2/2/5, and the size of each local expert is 5/8/5/20. For SOGP, the size of the active set is 20/5/50/20. The parameter values were chosen to balance computational cost with the prediction accuracy and in our experiments we demonstrate the robustness of our approach to these parameters.

4.1 COMPARISON TO STATE-OF-THE-ART

We compare our approaches to two baselines: The first one is the approach of Ko and Fox [11] where a GPDM is learned offline with gradient descent [28] before performing particle filtering for state estimation. The second baseline is similar, but learns the GPDM offline using stochastic gradient descent [29]. We tested the baselines in two different settings. First, only the initial training set is available to learn the prediction and observation models. Second, all the data (including future streamed examples) are used to learn the prediction and observation models. Note that the latter represents the oracle for Ko and Fox [11].

Number of Particles: We evaluate how the accuracy changes as a function of the number of particles, N_p. As expected, the prediction error is reduced in all the methods when the number of particles increases. As shown in the first row of Fig. 1, our approaches are superior to the baselines. Importantly, we outperformed the oracle baseline as we are able to represent multi-modal distributions effectively. This is particularly important in the exercise sequence as the dynamics are clearly multimodal due to the different motions that are performed in the sequence. Furthermore, our LGP variant outperforms SOGP. We believe this is due to the fact that SOGP has a fixed capacity while LGP is able to leverage more training data when making predictions.

Influence of noise: In this experiment we evaluate the robustness of all approaches to additive noise in the observations. The second row of Fig. 1 shows that our LGP particle filter significantly outperforms the baselines, particularly in the exercise sequence. Our SOGP outperforms all baselines that have access to the same training set, and is only beaten by the oracle for walking.

Size of Training Set: We next evaluate how the accuracy depends on the size of the inital training set, T_0. The first row of Fig. 2 clearly indicates that our methods perform well even when the training set is very small. In contrast, the two baselines require bigger training sets to achieve comparable performance. This is expected as the baselines do not update the latent space to take the incoming observations into account.

4.2 QUALITATIVE EXPERIMENTS

Fig. 3 shows the latent space of both SOGP and LGP filters when employing 50 particles for each time step (depicted in blue). From the 3D latent space and predicted skeletons, we find that the manifolds of both LGP and SOGP particle filters have a good representation of the high-dimensional human motion data.

4.3 PROPERTIES OF OUR METHODS

We next discuss various aspects of our method and evaluate the influence of the parameters of SOGP and LGP filters. For LGP, due to the fact that the data sizes of walking, golf and swimming motions are small, we reduced the number (size) of local experts to be able to increase the size (number) of the local experts.

Computational Complexity: Overall the computational complexity of our method (Alg 1) is mainly determined by the complexity of constructing a prediction distribution for each components (lines 5-6 and 11-12) and model updates (line 18 and line 23). Specifically, for an individual component which is either SOGP or LGP, computing the prediction distribution is $O(N_A^2)$ or $O(M_\mathbf{a} M_\mathbf{b}^3 + T M_\mathbf{a} M_\mathbf{b})$ respectively where N_A is the size of active set, $M_\mathbf{a}$ is the number of local experts, $M_\mathbf{b}$ are the number of neigh-

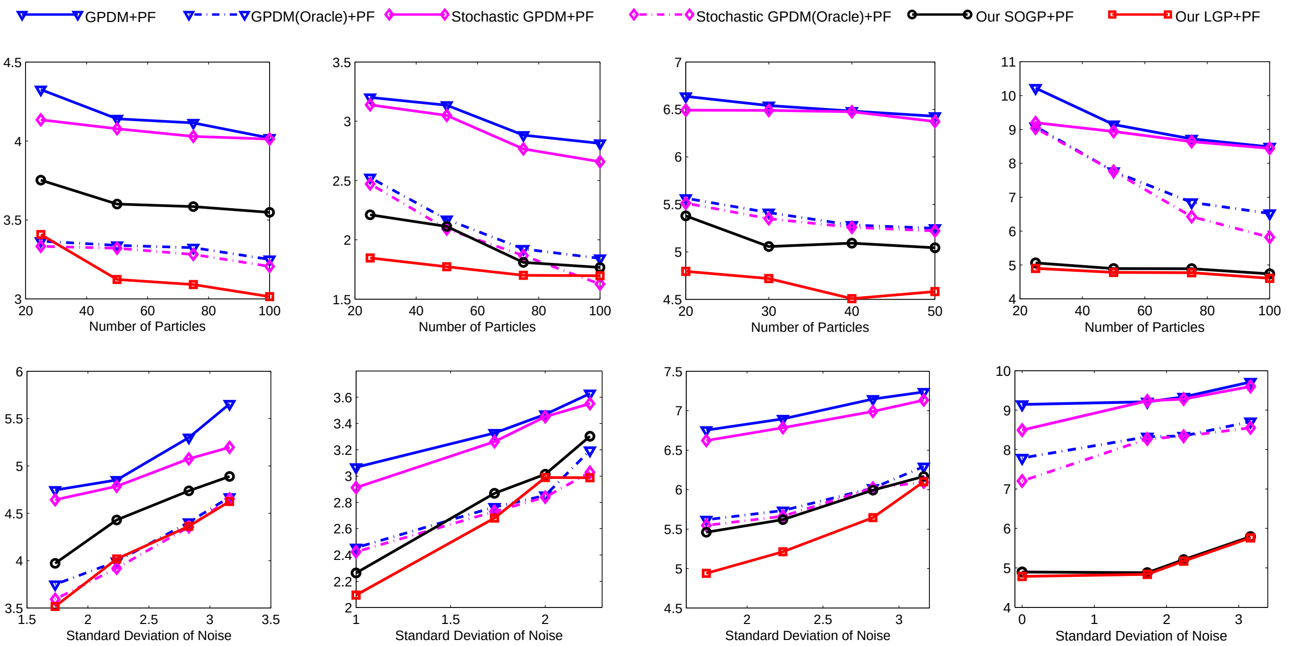


Figure 1: Root mean squared error as a function of (1st row) the number of particles in the particle filter, (2nd row) the standard deviation of noise added to the observations. The columns (from left to right) represent walking, golf, swimming and exercise motions. Note that our approach outperforms the baselines in all settings. Handling multimodal distributions is particularly important in the exercise example as it is composed of a variety of different motions.

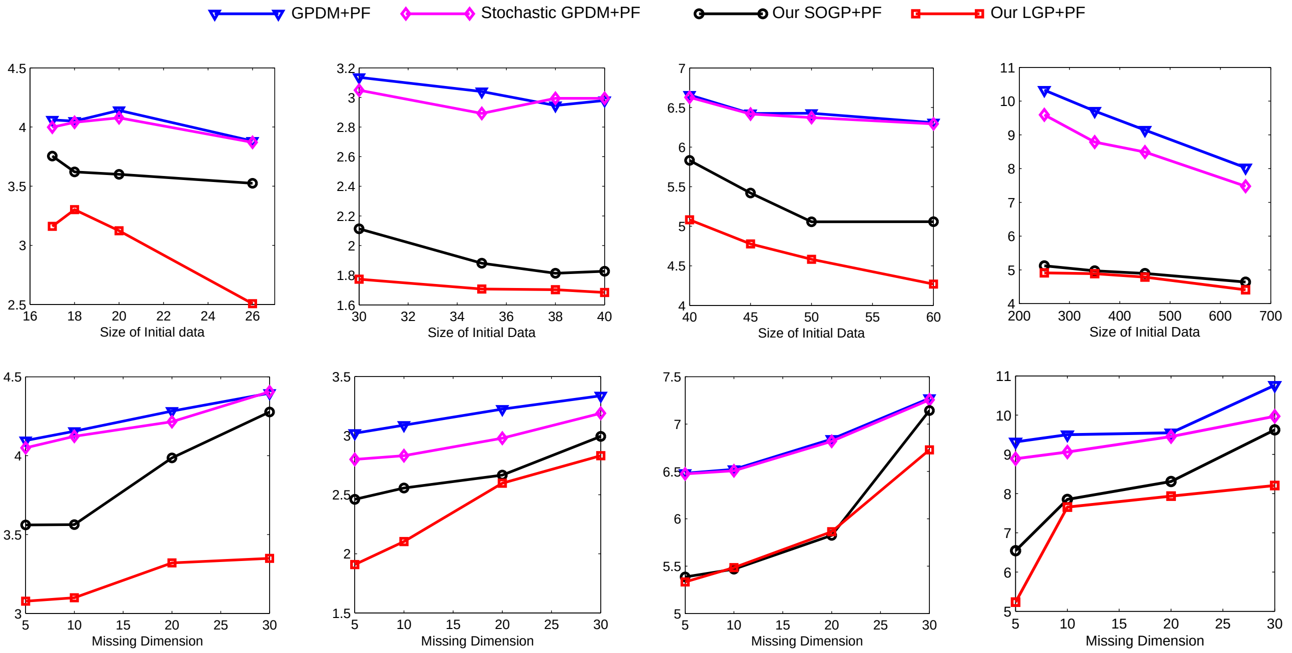


Figure 2: Root Mean Squared Error as a function of (1st row) the number of initial training points, (2nd row) the number of the missing dimensions. The columns (from left to right) are respectively for walking, golf, swimming and exercise motions.

bors in the output space and $TM_{\mathbf{a}}M_{\mathbf{b}}$ comes from the KNN search. The model updates for the mixture components (lines 18 and 23) have a computational complexity of $O(N_A^2)$ and $O(1)$ for SOGP and LGP respectively.

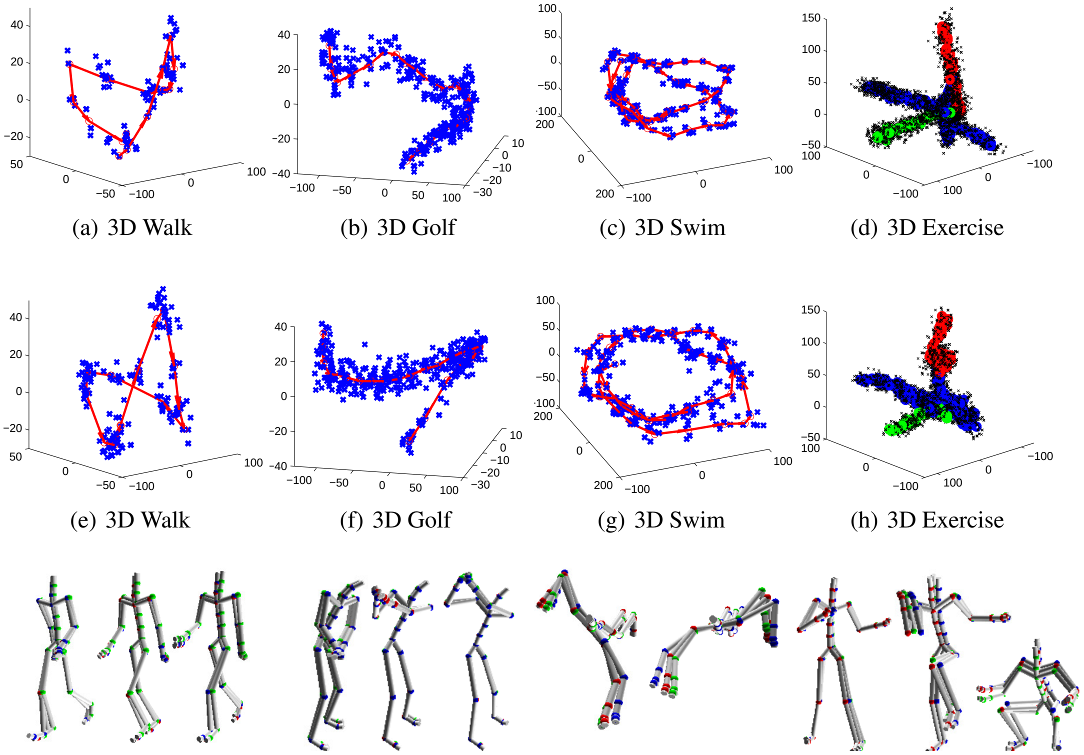

(a) 3D Walk
(b) 3D Golf
(c) 3D Swim
(d) 3D Exercise
(e) 3D Walk
(f) 3D Golf
(g) 3D Swim
(h) 3D Exercise
(i) Walk (t=23, 26, 29)
(j) Golf (t=33, 41, 54)
(k) Swim (t=71,79)
(l) Exercise (t=508,555,721)

Figure 3: 3D latent spaces learned while tracking: The first row depicts the results of our SOGP variant, while the second row shows our LGP variant. In the walk/golf/swimming plots the red curve represents the predicted mean of the latent state sequence and the blue crosses are the particles at each step. In the plots for exercise (last column), the red, blue and green curves are the predicted mean of the latent state for three motions in the exercise sequence, and the black crosses are the particles at each step. The third row depicts the predicted skeletons, where ground truth is shown in green, our SOGP variant in blue and our LGP variant in red.

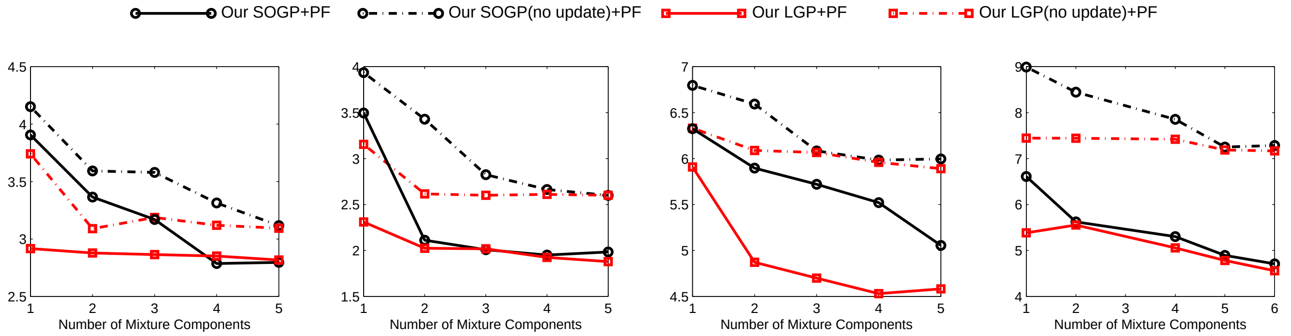
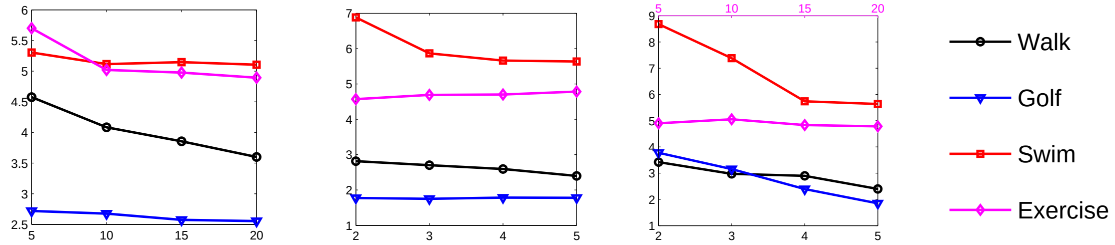


Figure 4: Root Mean Squared Error as a function of the number of mixture components. The columns (from left to right) are respectively for walking, golf, swimming and exercise motions.

Number of Mixture Components: Fig. 4 shows performance as a function of the number of mixture components, R_M and R_O, for both SOGP and LGP. For LGP+PF in walk/golf/swim/exercise, the number of local GPs are 1/1/2/5 and the size of each local GP are 3/3/5/20. In all cases, we set $R_M = R_O$. Note that performance typically increases with the number of mixture components, for SOGP, but less so for LGP. Furthermore, our approaches outperform the baselines in which the model is not updated during filtering indicating that the online model updating is very important in practice. Also note that while LGP generally outperforms SOGP, the difference quickly declines



(a) Size of Active Data (b) Number of Local GPs (c) Size of Local GPs

Figure 5: Root mean squared error as a function of the size of the active set in SOGP, the number of the local GP experts and the size of each local GP expert in LGP. In the subplot 5(c), the top x-axis is for exercise and the bottom one for the other motions.

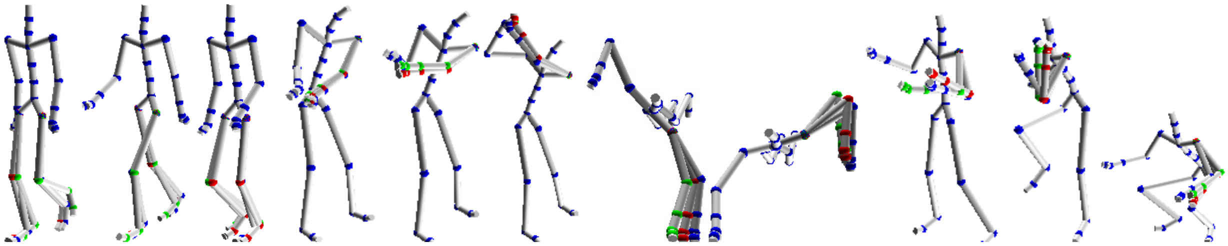


Figure 6: Predicted skeleton for missing parts (Walk: two legs; Golf, Swim and Exercise: left arm). The ground truth is shown in green, our SOGP particle filter in blue and LGP particle filter in red. We show the predicted performance at t=24, 27, 32 for walk, t=34, 38, 50 for golf, t=71,79 for swim, t=470, 592, 711 for exercise.

as the number of mixture components increases. This suggests that, when the fixed memory requirements of SOGP is desirable, a larger number of mixture components will achieve performance comparable to LGP.

Active Set size in SOGP: To explore the effect of the size of the active set, N_A, on performance we set the number of mixture components, R_M and R_O, to be 2/1/5/5 for walk/golf/swim/exercise, and use the same settings as before for the other parameters. Results are shown in Fig. 5(a). As expected performance improves when the size of the active set increases.

Number and Size of Local Experts in LGP: Figs. 5(b) and 5(c) show the performance of our approach as a function of the number of local GP experts, $M_{\mathbf{a}}$, as well as their size, $M_{\mathbf{b}}$. For this experiment we set the number of mixture components, R_M and R_O, to be 1/2/1/5 and used the same settings as before for the other parameters except when evaluating the size of each local GP expert where we set the number of local GP experts to 5/2/5/5. As shown in the figure, even with the small number (size) of local GP experts, we still achieve good performance.

4.4 HANDLING MISSING DATA

In this setting, we evaluate the capabilities of our approaches to handle missing data. We assume that the initial set has no missing values, but a fixed set of joint angles are missing for all incoming frames. Our approach is able to cope with missing data with only two small modifications. First, particles are weighted only based on the observed dimensions. Furthermore, when updating the prediction and observation models, we employ mean imputation for the missing observation dimensions. Fig. 6 shows reconstructions of the missing dimensions for all our motions, which consists of the two legs for walking, the left arm for golf swing, swimming and exercise motions. We can see that our approach is able to reconstruct the missing parts well.

Finally, to evaluate the tracking performance as a function of the number of missing dimensions, we randomly generate the indices for the missing dimensions and use the same missing dimensions for all incoming frames. The second row of Fig. 2 shows that, compared to the baselines, our methods perform well even when the number of missing dimensions is 20 (1/3 of the skeleton) for all the motions. In addition, our LGP particle filter outperforms our SOGP variant.

5 CONCLUSION

In this paper we have presented a novel non-parametric approach to Bayesian filtering, where the observation and prediction models are constructed using a mixture model with GP components learned in an online fashion. We have demonstrated that our approach can capture the multimodality accurately and efficiently by online updates. We have explored two fast GP variants for updating which keep memory and computation bounded for individual mixture components. We have demonstrated the effectiveness of our approach when tracking different human motions and explored the impact of various parameters on performance. The Local GP particle filter proved superior to our SOGP variant, however these differences can be mitigated by using more mixture components when using SOGP. In the future, we plan to investigate the usefulness of our approach in other settings such as shape deformation estimation and financial time series.

References

[1] K. Chalupka, C. K. I. Williams, and I. Murray. A Framework for Evaluating Approximation Methods for Gaussian Process Regression. *JMLR*, 2013.

[2] CMU Mocap Database. CMU Mocap Database. http://mocap.cs.cmu.edu/.

[3] L. Csato and M. Opper. Sparse online gaussian processes. In *Neural Computation*, 2002.

[4] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck. Analytic moment-based gaussian process filtering. In *ICML*, 2009.

[5] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 2000.

[6] T. Hastie and C. Loader. Local Regression: Automatic Kernel Carpentry. *Statistical Science*, 1993.

[7] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 1991.

[8] O. C. Jenkins and M. Matarić. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *ICML*, 2004.

[9] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *American Control Conference*, 1995.

[10] J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. In *IROS*, 2008.

[11] J. Ko and D. Fox. Learning gp-bayesfilters via gaussian process latent variable models. In *RSS*, 2009.

[12] N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *JMLR*, 6:1783–1816, 2005.

[13] N. Lawrence, M. Seeger, and R. Herbrich. Fast Sparse Gaussian Process Methods: The Informative Vector Machine. In *NIPS*, 2003.

[14] C-S. Lee and A. Elgammal. Coupled Visual and Kinematics Manifold Models for Human Motion Analysis. *IJCV*, 2010.

[15] S. Levine, J. Wang, A. Haraux, Z. Popovic, and V. Koltun. Continuous character control with low-dimensional embeddings. In *SIGGRAPH*, 2012.

[16] K. Moon and V. Pavlovic. Impact of dynamics on subspace embedding and tracking of sequences. In *CVPR*, pages 198–205, 2006.

[17] D. Nguyen-Tuong, J. Peters, and M. Seeger. Local gaussian process regression for real time online model learning and control. In *NIPS*, 2008.

[18] J. Quiñonero-Candela and C. E. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *JMLR*, 2005.

[19] S. Roweis and L. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290 (5500):2323–2326, 2000.

[20] S. Schaal and C. G. Atkeson. Constructive Incremental Learning From Only Local Information. *Neural Computation*, 1998.

[21] J. Tenenbaum, V. de Silva, and J. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 2000.

[22] R. Turner, M. P. Deisenroth, and C. E. Rasmussen. State-space inference and learning with gaussian processes. In *AISTATS*, 2010.

[23] R. Urtasun and T. Darrell. Sparse probabilistic regression for activity-independent human pose inference. In *CVPR*, 2008.

[24] R. Urtasun, D. Fleet, A. Hertzman, and P. Fua. Priors for people tracking from small training sets. In *ICCV*, 2005.

[25] R. Urtasun, D. Fleet, and P. Fua. 3d people tracking with gaussian process dynamical models. In *CVPR*, 2006.

[26] R. Urtasun, D Fleet, A. Geiger, J. Popovic, T. Darrell, and N. Lawrence. Topologically-constrained latent variable models. In *ICML*, 2008.

[27] S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría. Kernel recursive least-squares tracker for time-varying regression. IEEE Transactions on Neural Networks and Learning Systems, 2012.

[28] J. Wang, D. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *PAMI*, 30 (2):283–298, 2008. ISSN 0162-8828.

[29] A. Yao, J. Gall, L. V. Gool, and R. Urtasun. Learning probabilistic non-linear latent variable models for tracking complex activities. In *NIPS*, 2011.