Hierarchical Deep Gaussian Processes Latent Variable Model via Expectation Propagation

Nick Taubert and Martin A. Giese

Section Computational Sensomotorics, CIN/HIH, University Clinic Tübingen,
Otfried-Müller-Str. 25, 72076 Tübingen, Germany
{nick.taubert, martin.giese}@uni-tuebingen.de

Abstract. Gaussian Processes (GPs) and related unsupervised learning techniques such as Gaussian Process Latent Variable Models (GP-LVMs) have been very successful in the accurate modeling of high-dimensional data based on limited amounts of training data. Usually these techniques have the disadvantage of a high computational complexity. This makes it difficult to solve the associated learning problems for complex hierarchical models and large data sets, since the related computations, as opposed to neural networks, are not node-local. Combining sparse approximation techniques for GPs and Power Expectation Propagation, we present a framework for the computationally efficient implementation of hierarchical deep Gaussian process (latent variable) models. We provide implementations of this approach on the GPU as well as on the CPU, and we benchmark efficiency comparing different optimization algorithms. We present the first implementation of such deep hierarchical GP-LVMs and demonstrate the computational efficiency of our GPU implementation.

Keywords: Deep GP-LVM, hierarchical probabilistic model, dimension reduction, motion synthesis, Expectation Propagation

1 Introduction

Many applications, e.g. in computer graphics and robotics, require real-time generative models for complex, high-dimensional, coordinated human motion. One possible solution of this problem is the use of neural networks [10, 12]. The generalization properties of such networks are not easy to control, and often they require substantial amounts of training data to accomplish high accuracy and robustness of the generated motion. As an alternative to this approach we propose here probabilistic graphical models [1]. Such models provide an attractive theoretical framework for the construction of modular and hierarchical models, and for inference on arbitrary variables within these models. While probabilistic models have been used extensively for motion synthesis and editing in computer graphics [5, 2, 13], or robotics [30, 23], many of these techniques result in offline models that are not suitable for embedding in online control systems, or the learning from large data sets, due to their high computational complexity.

Gaussian process latent variable models (GP-LVM) provide a successful framework for the very accurate approximation high-dimensional human motion, where Gaussian processes (GP) can be interpreted as neural networks with infinitely many hidden units [21]. GP-LVMs have been successfully applied for kinematic

modeling and motion interpolation [9], inverse kinematics [16], and for the learning of low-dimensional dynamical models [29]. By inclusion of sparse approximation techniques, they can be made suitable for real-time applications [25]. However, they have a tendency to overfit and additional limitations, which make them unsuitable for a variety of applications, e.g. on large data sets. Approximative inference exploiting sparse approximations of Gaussian processes (GP) within a variational free energy (VFE) framework control for overfitting [26]. In addition, they are suitable for the formulation of multi-layer architectures in the form of Deep Gaussian processes (DGP) [14, 27]. Such architectures are equivalent to deep neural networks with infinite many hidden units per layer [7]. However, the VFE approach is not parallizable and produces unnecessarily large memory footprints during learning, which makes it specifically unsuitable for GPU implementations.

We propose here to use instead approximate inference exploiting Expectation propagation (EP) [18], resulting in an algorithm that is more suitable for GPU implementations. The use of stochastic expectation propagation (SEP) [17] it also reduces the memory overhead. In addition, the underlying learning scheme of Power EP [19] combines the advantages of VFE and EP within a single algorithm.

Exploiting such DGPs, our main contributions are: a hierarchical probabilistic graphical model in form of *hierarchical* deep GP-LVMs (hDGP-LVM); implementation of this model on the GPU, exploiting approximate inference using SEP. We demonstrate the suitability of this method for the learning of online generative models for complex full-body movements of two interacting humans with 189 degrees of freedom.

The following sections, first develop the underlying mathematical theory. We then discuss briefly the implementation details and present our results, comparing the GPU and CPU implementations, followed by a conclusion.

2 Preliminaries

2.1 GP Regression and Approximation

In the function space view GPs can be considered as nonlinear mapping, $f(\mathbf{x})$, from an input variable $\mathbf{x}$ to a one-dimensional real-valued output variable. The function value $\mathrm{f}_n := f(\mathbf{x}_n)$, at a particular input point, $\mathbf{x}_n$ of an input set $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^{\mathrm{T}} \in \mathbb{R}^{N \times Q}$, is a random variable, and a GP is an infinite collection of random variables, any finite number of which have a joint Gaussian distribution [22, 3], where N being the sample size and Q the dimensionality of the input space.

A real Gaussian process $f(\mathbf{x})$ is characterized through its mean function $m(\mathbf{x})$ and kernel function $k(\mathbf{x}, \mathbf{x}')$,

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned} \tag{1}$$

This can be interpreted as $f(\mathbf{x})$ being drawn from a Gaussian process prior with mean function $m(\mathbf{x})$ and kernel function $k(\mathbf{x}, \mathbf{x}')$,

$$f(\mathbf{x}) \sim \mathcal{GP}(f(\mathbf{x}); m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{2}$$

In most cases we assume a zero mean function, $m(\mathbf{x}) = 0$, since for our application the prior knowledge about $f(\cdot)$ can be encoded by the kernel function and

its hyperparameters. For our model we used in all layers an automatic relevance determination (ARD) kernel of the form,

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{q=1}^{Q}\frac{|x_q - x'_q|^2}{l_q^2}\right), \tag{3}$$

where σ is the variance and l_q the length-scale of the q-th input dimension of the kernel. For this particular kernel $\theta = (\sigma, \{l_q\}_{q=1}^{Q})$ are the hyperparameters.

Typically for regression models, a data set $\{y_n\}_{n=1}^{N}$ is defined by an unknown function $f(\cdot) := f$, evaluated at input location $\mathbf{x}_n$ and corrupted by some independent noise ε_n,

$$y_n = f(\mathbf{x}_n) + \varepsilon_n. \tag{4}$$

With the prior $p(\varepsilon_n) = \mathcal{N}(\varepsilon_n; 0, \beta^{-1})$ the probabilistic model can be written as follows,

$$f|\theta \sim \mathcal{GP}(f; 0, k(\cdot, \cdot)), \tag{5}$$

$$p(\mathbf{y}|f, \beta) = \prod_{n=1}^{N}\mathcal{N}(y_n; f(\mathbf{x}_n), \beta^{-1}), \tag{6}$$

where $\mathbf{y}$ specifies the vector of all one dimensional data points. To find the noise free function values $\mathbf{f}$ and the right hyperparameter set $\{\theta, \beta\}$ by hand is quite difficult. The usual Bayesian approach would be to specify a prior over the hyperparameter set to compute the joint posterior $p(\mathbf{f}, \theta, \beta|\mathbf{y})$. For Gaussian processes this inference problem is typically not analytically solveable, due to the intrinsic non-linearity of the GP. A common practice to handle this problem is to optimize the hyperparameter set instead by minimizing the negative log-marginal likelihood,

$$\begin{aligned}
\mathcal{L}(\theta, \beta) &= -\log p(\mathbf{y}|\theta, \beta), &(7)\\
&= -\log \int p(\mathbf{y}|f(\mathbf{x}_n), \beta)p(f|\theta)\,\mathrm{d}f, &(8)\\
&= -\log \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{K_{ff}} + \beta^{-1}\mathbf{I}), &(9)\\
&= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K_{ff}} + \beta^{-1}\mathbf{I}| - \frac{1}{2}\mathbf{y}^{\mathrm{T}}(\mathbf{K_{ff}} + \beta^{-1}\mathbf{I})^{-1}\mathbf{y}. &(10)
\end{aligned}$$

This allows to derive a conditioned posterior $p(\mathbf{y}_*|\mathbf{y}, \theta, \beta)$, based on Gaussian identities [1], from the joint probability of the prior function of the training set $p(\mathbf{y}|\theta, \beta)$ and the prior function of the test set $p(\mathbf{y}_*|\theta, \beta)$ for the prediction of new test outputs $\mathbf{y}_*$ with given test inputs $\mathbf{x}_*$ and a *fixed* hyperparameter set [15],

$$\tilde{m}(\mathbf{x}_*) = \mathbf{k}_{*,\mathbf{f}}(\mathbf{K_{ff}} + \beta^{-1}\mathbf{I})^{-1}\mathbf{y}, \tag{11}$$

$$\tilde{k}(\mathbf{x}_*, \mathbf{x}'_*) = k(\mathbf{x}_*, \mathbf{x}'_*) - \mathbf{k}_{*,\mathbf{f}}(\mathbf{K_{ff}} + \beta^{-1}\mathbf{I})^{-1}\mathbf{k}_{*,\mathbf{f}}^{\mathrm{T}}, \tag{12}$$

which is also a GP. The constructed matrices result from the covariance function evaluations of the training inputs $\{\mathbf{x}_n\}_{n=1}^{N}$, i.e $[\mathbf{K_{ff}}]_{n,n'} = k(\mathbf{x}_n, \mathbf{x}_{n'})$ and similarly between the test inputs $\{[\mathbf{x}_*]_i\}_{i=1}^{I}$ and training input locations $\{\mathbf{x}_n\}_{n=1}^{N}$, $[\mathbf{k}_{*,\mathbf{f}}]_{i,n} = k([\mathbf{x}_*]_i, \mathbf{x}_n)$.

4 N. Taubert, M.A. Giese

This minimization procedure has several disadvantages: First, it often gets stuck at local minima and has a computational cost, due to the inversion of $\mathbf{K_{ff}} + \beta^{-1}\mathbf{I}$, see equation (10), of $O(N^3)$ at each iteration step and a $O(N^2)$ for prediction. Second, one has to store the full observation vector $\mathbf{y}$ for learning and prediction, see equations (10) and (11). These limitations prohibit larger scale learning using this approach, because of time and memory limitations. Especially the memory aspect is critical if optimization is implemented by GPU computing. Further, due to intractability of the posterior joint probability $p(\mathbf{f}, \theta, \beta|\mathbf{y})$, the construction of hierarchical (deep) models is not possible, because of the conditional dependency between the function value sets in the different layers.

In this paper, we show that approximate inference, exploiting Expectation Propagation (EP) framework in combination with sparse approximations of the Gaussian processes, offers an elegant solution for these problems.

2.2 GP Sparse Approximations

In order to implement the proposed GP framework for large data sets it is essential to reduce the computational complexity. This can be accomplished by a the GP sparse approximation approach [4]. For this purpose, the noise-free function value set $\mathbf{f}$, is approximated by selection of a small set of $M \ll N$ pseudo-point inputs $\{[\mathbf{x_u}]_m\}_{m=1}^M$, mapped to the function values $\mathbf{u}$. It is assumed that training and test points of the Gaussian process are approximately conditionally independent, if conditioned on their pseudo-points so that $f = \{\mathbf{f}, \mathbf{u}, f_{\neq \mathbf{f},\mathbf{u}}\}$. Under this assumption, the GP prior, see equation (5), can be approximated as follows:

$$q(f|\theta) = q(\mathbf{f}|\mathbf{u}, \theta)p(\mathbf{u}|\theta)p(f_{\neq \mathbf{f},\mathbf{u}}|\mathbf{f}, \mathbf{u}, \theta), \tag{13}$$

where $p(\mathbf{u}|\theta)$ is the GP prior over $\mathbf{u}$. The conditional relationship between $\mathbf{u}$ and $\mathbf{f}$ is fundamental for this equation. With the GP priors $p(\mathbf{f}|\theta)$ and $p(\mathbf{u}|\theta)$ it is possible to derive from their joint probability the conditional dependency $p(\mathbf{f}|\mathbf{u}, \theta) = \mathcal{N}(\mathbf{f}; \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}, \mathbf{D_{ff}})$, where $\mathbf{D_{ff}} = \mathbf{K_{ff}} - \mathbf{Q_{ff}}$ and $\mathbf{Q_{ff}} = \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}}$. The constructed matrices correspond to the covariance function evaluations at the pseudo-point input locations $\{[\mathbf{x_u}]_m\}_{m=1}^M$, i.e. $[\mathbf{K_{uu}}]_{m,m'} = k_{\mathbf{f}}([\mathbf{x_u}]_m, [\mathbf{x_u}]_{m'})$ and similarly covariance function evaluations between pseudo-point input and data locations $[\mathbf{K_{fu}}]_{n,m} = k_{\mathbf{f}}(\mathbf{x}_n, [\mathbf{x_u}]_m)$. The matrices of $p(\mathbf{f}|\mathbf{u}, \theta)$ can be approximated by simpler forms using several approaches, compactly summarized [3] by:

$$q(\mathbf{f}|\mathbf{u}, \theta) = \prod_{b=1}^{B} \mathcal{N}(\mathbf{f}_b; \mathbf{K_{f_b u}}\mathbf{K_{uu}^{-1}}\mathbf{u}, \alpha\mathbf{D}_{\mathbf{f}_b,\mathbf{f}_b}). \tag{14}$$

where b indexes B disjoint blocks of data-function values with $\mathbf{f}_b = [f_1, \ldots, f_B]^{\mathrm{T}}$. The Deterministic Training Conditional (DTC) approximation uses $\alpha \to 0$; the Fully Independent Training Conditional (FITC) approximation uses $\alpha = 1$ and $B = N$; the Partially Independent Training Conditional (PITC) approximation uses $\alpha = 1$ [4, 24]. This approximation assumes that $f(\mathbf{x})$ is fully determined by the pseudo-point inputs and reduces the computational cost to $O(M^2N)$ during learning and $O(MN)$ for prediction.

The new prior $q(f|\theta)$ with approximation can be combined with the data likelihood to obtain the modified generative model:

$$q(\mathbf{y}, f|\theta) = q(f|\theta) \prod_{n=1}^{N} p(y_n|f(\mathbf{x}_n), \theta). \tag{15}$$

This modified model is suitable for the construction of hierarchical models, keeping the memory inference steps node-local because each layer is associated with its own GP approximation. However, this formulation still leaves some problems unsolved. One is the intrinsic non-linearity of the data likelihood and its related analytic intractability. The second problem is that, due to GP approximation (especially for DTC), the model tends to overfit [26]. Approximate inference exploiting the EP framework can solve these remaining problems.

3 Methods

3.1 Stochastic Expectation Propagation

Expectation Propagation (EP) is a deterministic Bayesian inference algorithm [18] which allows to approximate intractable, but factorizable joint-distributions. EP returns a tractable form of the model joint-distribution, evaluated on the observed data. In the case of GP regression, the approximation takes the form of an unnormalized process $q^*(f|\theta) \approx p(\mathbf{y}, f|\theta)$ (the superscript $*$ defines an unnormalized process). The basic concept of distributional inference approximations, like Variational Free Enery (VFE) [26], EP [18] and Power EP [19] is the decomposition of the joint-distribution into terms of interest, such that $p(f, \mathbf{y}|\theta) = p^*(\mathbf{y}|f, \theta) = p(\mathbf{y}|\theta)p(f|\mathbf{y}, \theta)$. They are fully reflected by tractable terms of the decomposed approximation $q^*(f|\theta) = Zq(f|\theta)$, where the normalization constant Z approximates the marginal likelihood $p(\mathbf{y}|\theta) \approx Z$ and the posterior is approximated by GP sparse approximation, equation (13), so that $p(f|\mathbf{y}, \theta) \approx q(f|\theta)$, i.e. the approximate inference schemes return simultaneously approximations of the posterior and marginal likelihood in form of an unnormalized Gaussian process $q^*(f|\theta)$.

For the implementation of EP we reformulate the unnormalized form $p^*(\mathbf{y}|f, \theta)$ in terms of a dense Gaussian process prior $p(f|\theta)$, see equation (5), and the independent likelihoods $\{p(y_n|f, \theta)\}_{n=1}^N$. EP constructs the approximate posterior as a product of *site functions* t_n [20] and employs an approximating family whose form mirrors that of the target [3],

$$p^*(f|\mathbf{y}, \theta) = p(f|\mathbf{y}, \theta)p(\mathbf{y}|\theta) = p(f|\theta) \prod_{n=1}^{N} p(y_n|f, \theta)$$
$$\approx p(f|\theta) \prod_{n=1}^{N} t_n(\mathbf{u}) = Z\, q(f|\theta) = q^*(f|\theta), \quad (16)$$

where $t_n(\mathbf{u})$ is approximated by a simple Gaussian. The site functions were iterativly refined by minimizing an unnormalized Kullback-Leibler divergence, $\overline{\mathrm{KL}}$ [3], between the real posterior and each of the distributions formed by replacing one of the likelihoods by the corresponding approximating factor [17],

$$\arg\min_{t_n(\mathbf{u})} \overline{\mathrm{KL}}\left(p(f, \mathbf{y}|\theta) \;\middle\|\; \frac{p(f, \mathbf{y}|\theta)\, t_n(\mathbf{u})}{p(y_n|\mathrm{f_n}, \theta)} \right)$$
$$= \arg\min_{t_n(\mathbf{u})} \overline{\mathrm{KL}}(p^*_{\setminus n}(f|\theta)p(y_n|\mathrm{f}_n, \theta) \parallel p^*_{\setminus n}(f|\theta)\, t_n(\mathbf{u})), \quad (17)$$

Unfortunately, such an update is still intractable as it involves the computation of the full posterior. Instead, EP replaces the leave-one-out posteriors $p^*_{\setminus n}(f|\theta) \propto p(f, \mathbf{y}|\theta)/p(y_n|\mathrm{f}_n, \theta)$ on both sides of KL by approximate leave-one-out posterior $q^*_{\setminus n}(f|\theta) \propto q^*(f|\theta)/t_n(\mathbf{u})$, called the cavity, so that:

$$\overline{\mathrm{KL}}(q^*_{\setminus n}(f|\theta)p(y_n|\mathrm{f}_n, \theta) \parallel q^*_{\setminus n}(f|\theta)t_n(\mathbf{u})) = \overline{\mathrm{KL}}(q^*_{\setminus n}(f|\theta)p(y_n|\mathrm{f}_n, \theta) \parallel q^*(f|\theta)).$$

The update for the approximating factors are coupled and must be optimized by iterative updating. EP is doing this in four steps. We apply here a generalized version, called Power EP [19], where only a fraction α of the approximate or true likelihood is removed (or included). The steps are as follows:

1. Compute the cavity distribution by removing a fraction of one approximate factor, $q^*_{\setminus n}(f|\theta) \propto q^*(f|\theta)/t^\alpha_n(\mathbf{u})$.
2. Compute a hybrid or tilted distribution, $\tilde{p}(f|\theta) = q^*_{\setminus n}(f|\theta)p^\alpha(y_n|\mathrm{f}_n, \theta)$.
3. Project the hybrid distribution onto the approximate posterior by minimizing unnormalized KL divergence,

$$q^*(f|\theta) \leftarrow \underset{q^*(f|\theta)\in\mathcal{Q}}{\arg\min}\ \overline{\mathrm{KL}}(\tilde{p}(f|\theta) \parallel q^*(f|\theta)),$$

   where $\mathcal{Q}$ is the set defined in equation (16).
4. Update the approximate factor by including the new fraction of the approximate factor, $t_n(\mathbf{u}) = [t_n^{1-\alpha}(\mathbf{u})]_{\mathrm{old}}[t_n^\alpha(\mathbf{u})]_{\mathrm{new}}$, with $[t_n^\alpha(\mathbf{u})]_{\mathrm{new}} = q^*(f|\theta)/q^*_{\setminus n}(f|\theta)$.

The fractional updates are equivalent to running the original EP procedure, but replacing the KL minimisation with an α-divergence minimisation [31, 19], where

$$\overline{\mathrm{D}}_\alpha(p^*(f|\theta) \parallel q^*(f|\theta)) = \frac{1}{\alpha(1-\alpha)} \int [\alpha p^*(f|\theta) + (1-\alpha)q^*(f|\theta) - p^*(f|\theta)^\alpha q^*(f|\theta)^{1-\alpha}]\,\mathrm{d}f. \quad (18)$$

The α-divergence becomes the inclusive KL divergence, $\overline{\mathrm{KL}}(p^*(f|\theta) \parallel q^*(f|\theta))$, when $\alpha = 1$. It becomes the exclusive KL divergence, $\overline{\mathrm{KL}}(q^*(f|\theta) \parallel p^*(f|\theta))$, when $\alpha \to 0$. Minimising a set of local exclusive KL divergences is equivalent to minimizing a single global exclusive KL divergence [19] and the Power EP solution is the minimum of a VFE [3]. Since the unnormalized approximation, $q^*(f|\theta)$, consists of the product of independent approximate factors $t_n(\mathbf{u})$, the data can be partitioned in B disjoint blocks $\mathbf{y}_b = \{y_n\}_{n\in\mathcal{B}_b}$. The choice of α and corresponding approximate factors $t_b(\mathbf{u})$ have a strong influence on the GP sparse approximation scheme, see equation (14) (and [3] for more details) and enables batch learning, which makes the GP regression task scaleable. However, local computation comes at the cost of memory overhead that grows with the number of datapoints, since local approximating factors need to be stored for every datapoint.

Stochastic Expectation Propagation (SEP) reduces the memory complexity of Power EP by a factor of N. This is accomplished by parameterizing a global factor, $t(\mathbf{u})$, that captures the average effect of a likelihood on the posterior $t(\mathbf{u})^N := \prod_{n=1}^N t_n(\mathbf{u})$ and combines the benefits of local approximation

(tractability of updates, distributability, and parallelisability) with global approximation (reduced memory demands) [17]. The set of the approximate posterior in equation (16) can be rewritten as,

$$p^*(f|\mathbf{y},\theta) = p(f|\mathbf{y},\theta)p(\mathbf{y}|\theta) = p(f|\theta)\prod_{n=1}^{N} p(y_n|f,\theta)$$
$$\approx p(f|\theta)\prod_{n=1}^{N} t_n(\mathbf{u}) = p(f|\theta)t(\mathbf{u})^N = q^*_{\mathrm{SEP}}(f|\theta). \quad (19)$$

One method to implement SEP would be to compute the approximate factorizations by the iterative procedure of Power EP and optimize the hyperparameters in an outer loop with Power EP [17]. But, it was shown in [11] that the factor tying approximation turns the optimization problem into a minimization problem, i.e. the approximate Power EP energy can be optimized with standard optimization algorithms (e.g. ADAM, L-BFGS-B) to find the approximate posterior and hyperparameters at the same time for each iteration step. This makes construction of hierarchical deep GP-LVMs feasible.

3.2 DGP-LVM

Standard GP models with sparse approximation have strong limitations in terms of real world data sets with large numbers of training examples. This typically requires a substantial increase of the pseudo-inputs for a good approximation, resulting in a quadratic increase of computational complexity with the number of data points. This renders larger-scale learning not practicable. Constructing a multi-layer GP reduces the computational cost to $O(NLM^2)$, where L is the number of layers. Further, DGPs employ a hierarchical structure of GP mappings and therefore are arguably more flexible, have a greater capacity to generalize, and are potentially able to provide better predictive performance [6].

DGPs have relationships to different kinds of GP models, including GP-LVMs. The basic probabilistic model can be written as,

$$f^{(l)}|\theta^{(l)} \sim \mathcal{GP}(f^{(l)}; 0, k^{(l)}(\cdot,\cdot)),\ l = 1,\ldots,L$$
$$p(\mathbf{h}^{(l)}|f^{(l)},\mathbf{h}^{(l-1)},\beta^{(l)}) = \prod_{n=1}^{N} \mathcal{N}(h_n^{(l)}; f^{(l)}(h_n^{(l-1)}), [\beta^{(l)}]^{-1}),\ h_n^{(1)} = \mathbf{x}_n,\ h_n^{(L)} = y_n$$

where $h_n^{(l)}$ is the hidden variable associated to the l-th layer, which forms the output vector $\mathbf{h}^{(l)} := \{h_n^{(l)}\}_{n=1}^N$, i.e. $\mathbf{h}^{(L)} = \mathbf{y}$, and $f^{(l)}$ is the function of the l-th layer. In order to infer the posterior distribution over the latent functions mappings, hidden variables and to obtain an estimate for the marginal likelihood for hyperparameter tuning the joint probability can be written as follows,

$$p(\mathbf{h}^{(1:L)}, f^{(1:L)}|\theta^{(1:L)}) = \prod_{n=1}^{N}\prod_{l=1}^{L} p(f^{(l)}|\theta^{(l)})p(h_n^{(l)}|f^{(l)},\theta^{(l)}), \quad (20)$$

where $\mathbf{h}^{(1:L)}$ is a short notation for $\mathbf{h}^{(1)},\ldots,\mathbf{h}^{(L)}$, likewise for $f^{(1:L)}$ and $\theta^{(1:L)}$. Again, the joint probability is analytically intractable, due to the non-liniarity

of the data likelihoods and also the unknown intermediate outputs of each layer. So we make use of the SEP set for a tied factor constraint, see equation (19), in combination with GP sparse approximation to approximate the data likelihood of each layer,

$$q^*_{\mathrm{SEP}}(f^{(1:L)}|\theta^{(1:L)}) = Z\prod_{l=1}^{L} q^{(l)}(f^{(l)}|\theta^{(l)}) = \prod_{l=1}^{L} p(f^{(l)}|\theta^{(l)})t^{(l)}(\mathbf{u}^{(l)})^N, \tag{21}$$

to obtain a scalable, convergent approximate inference method. The model above can be extended for unknown and random inputs $\mathbf{X}$, defining a deep Gaussian process latent variable model (DGP-LVM).

In a GP-LVM regression model, we suppose that each output point $\mathbf{y}_n$ of a high dimensional data set $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times D}$ is represented by a corresponding, uncertain instance $\mathbf{x}_n$ of a low dimensional latent input variable, where D is the dimension of the data space. Each dimension of $\mathbf{y}_n = [y_{n,1}, \ldots, y_{n,D}]^{\mathrm{T}}$ is defined by an unknown function $f_d(\cdot) := f_d$, evaluated at input location $\mathbf{x}_n$. The usual generative model can be summarised as follows,

$$p(\mathbf{X}) = \prod_{n=1}^{N} p(h_n^{(1)}) \tag{22}$$

$$f_d|\theta \sim \mathcal{GP}(f_d; 0, k(\cdot,\cdot)), \quad \text{for } d = 1, \ldots, D \tag{23}$$

$$p(\mathbf{Y}|\mathbf{X}, f_1, \ldots, f_D, \beta) = \prod_{n=1}^{N}\prod_{d=1}^{D} \mathcal{N}(y_{n,d}; f_d(\mathbf{x}_n), \beta^{-1}), \tag{24}$$

where $p(\mathbf{X})$ is an isotropic Gaussian and $h_{1,n} = \mathbf{x}_n$. For a DGP-LVM the joint density in equation (20) has to be extended with the prior over $\mathbf{X}$ and dimensions D,

$$p(\mathbf{H}^{(1:L)}, f^{(1:L)}|\theta^{(1:L)}) = \prod_{n=1}^{N} p(h_n^{(1)}) \prod_{l=1}^{L}\prod_{d=1}^{D^{(l)}} p(f_d^{(l)}|\theta^{(l)}) p(h_n^{(l)}|f_d^{(l)}, \theta^{(l)}), \tag{25}$$

where $\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)}, \ldots, \mathbf{h}_N^{(l)}]^{\mathrm{T}} \in \mathbb{R}^{N\times D^{(l)}}$ for $l = 1, \ldots, L$. The approximate joint density takes the following form,

$$q^*_{\mathrm{SEP}}(f^{(1:L)}|\theta^{(1:L)}) = p(\mathbf{H}^{(1)})\gamma(\mathbf{H}^{(1)})^N \prod_{l=1}^{L}\prod_{d=1}^{D^{(l)}} p(f_d^{(l)}|\theta^{(l)}) t_d^{(l)}(\mathbf{u}^{(l)})^N, \tag{26}$$

where $\gamma(\mathbf{H}_1)^N$ is the global site function of $\mathbf{X}$. The prior over $\mathbf{X}$ can be replaced by top-down influences and enables, together with the approximate Power EP energy and the resulting minimization problem, hierarchical structures of multiple DGP-LVMs.

4 Hierarchical interaction model

We tested the proposed novel statistical framework by learning of the high dimensional kinematic data of two interacting humans (actors). The interactive

motion of a pair of actors was modeled using DGP-LVMs for successive dimension reduction, see Figure 1. The model is composed of three DGP-LVMs which were learned jointly. Learning includes both the bottom-up and top-down contributions at each hierarchy level. The right part of the figure represents the kinematic data of each actor $a \in \{1; 2\}$ by a DGP-LVM. The left part of the graph corresponds to a top level DGP-LVM that represents the interaction of both actors in a lower-dimensional latent space.

Modeling of the human actors. The bottom layers of our model represent the observed kinematic data of each actor $\mathbf{Y}_a$. The dimensionality of each of these data sets is reduced by a DGP-LVM with the same sample size ($N_a = 360$) and dimensionality ($D_a = 159$). Each hidden layer reduces the dimensionality by a factor of two (i.e. $D_a^{(3)} = 80$, $D_a^{(2)} = 40$ and $D_a^{(1)} = 20$), resulting in a dimensionality $Q_a = 10$ on the top layer $(\mathbf{X}_1, \mathbf{X}_2)$. The hidden layers are approximated sparsly by a pseudo-input set $[\mathbf{X_u}]_a^{(l)}$ of size $M_a^{(l)} = 30$.

Modeling of the interaction between the human actors. The top levels of our model represents the interaction of both actors, also with a DGP-LVM with two hidden layers. The mapping function of $\mathbf{h}_3^{(2)}$ maps onto the concatenated set $[\mathbf{X}_1, \mathbf{X}_2]$ with dimension $D_{[\mathbf{X}_1,\mathbf{X}_2]} = 20$. Similarly to the bottom DGP-LVMs each hidden layer reduced the dimensionality by a factor of two ($D_3^{(2)} = 10$, $D_3^{(1)} = 5$). Again, each mapping function is approximated by a pseudo-input set $[\mathbf{X_u}]_3^{(l)}$ of size $M_3^{(l)} = 100$. The latent inputs $\mathbf{X}_3$ have a dimensionality of $Q_3 = 2$. The unnormalized, approximated process of the whole model can be written as follows,

$$
\begin{aligned}
q^*_{\mathrm{SEP}}(f_1^{(1)}, f_1^{(2)}, f_1^{(3)}, f_2^{(1)}, f_2^{(2)}, f_2^{(3)}, f_3^{(1)}, f_3^{(2)} | \theta_1^{(1)}, \theta_1^{(2)}, \theta_1^{(3)}, \theta_2^{(1)}, \theta_2^{(2)}, \theta_2^{(3)}, \theta_3^{(1)}, \theta_3^{(2)}) \\
= p(\mathbf{X}_3)\gamma_3(\mathbf{X}_3)^N \prod_{l=1}^{L_3} \prod_{d=1}^{D_3^{(l)}} p(f_{3,d}^{(l)} | \theta_3^{(l)})\, t_{3,d}^{(l)}(\mathbf{u}^{(l)})^N \\
\times p(\mathbf{X}_2 | f_3^{(2)})\gamma_2(\mathbf{X}_2 | f_3^{(2)})^N \prod_{l=1}^{L_2} \prod_{d=1}^{D_2^{(l)}} p(f_{2,d}^{(l)} | \theta_2^{(l)})\, t_{2,d}^{(l)}(\mathbf{u}^{(l)})^N \\
\times p(\mathbf{X}_1 | f_3^{(2)})\gamma_1(\mathbf{X}_2 | f_3^{(2)})^N \prod_{l=1}^{L_1} \prod_{d=1}^{D_1^{(l)}} p(f_{1,d}^{(l)} | \theta_1^{(l)})\, t_{1,d}^{(l)}(\mathbf{u}^{(l)})^N. \quad (27)
\end{aligned}
$$

5 Results

We tested our method by modeling of the body movements of two interacting subjects, each with 189 DOFs, performing a 'high five' movement with different emotional expressions. The kinematic data (BVH format) was converted into exponential maps [8], and we computed also velocities within this representation. Our GP model was trained on a small set of 360 data points (6 trajectories, each

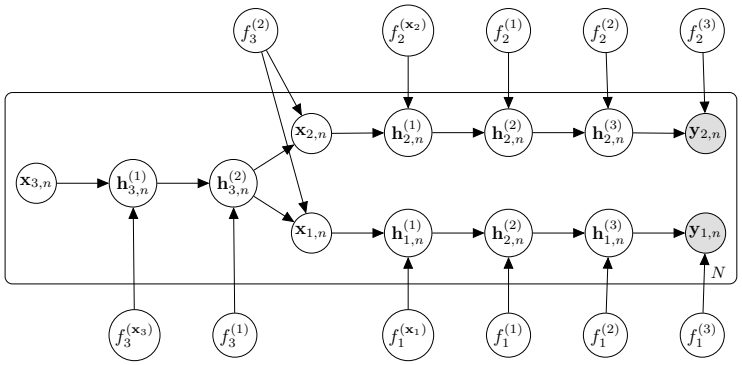

Fig. 1. *Graphical model of the hierarchical DGP-LVM for modeling the interactive motion of two actors. The model consists of three DGP-LVMs, where each actor* $\in \{1, 2\}$ *is represented by a DGP-LVM on the right. The interaction of both actors is represented by the third DGP-LVM on the left side of the graph.*

normalized to 60 frames (2 angry, 2 sad, 2 neutral). We varied the α-values (0.1, 0.25, 0.5, 0.75, 0.9) and tested different optimizers: ADAM and L-BFGS-B. The second algorithm was implemented with different line search algorithms: Armijo Backtracking (AB), More Thuente (MT), Wolfe Backtracking (WB).

Learning was implemented on the CPU with an AMD Ryzon Threadripper 1950X 16 core processor 3.4 GHz with 64 GB RAM, and on GPU with a NVIDIA Quadro P600 graphics card with 24 GB RAM. Both implementations were realized in C++ using the ArrayFire framework [28] which allows to implement just one code for CPU and GPU solutions.

For maximal 2000 iteration steps in each learning condition this took in average 3.63h on CPU and 1,21h on GPU. ADAM, due to stochastic gradient decent, was always the fastest optimizer in each condition (Table 1 right). The GPU implementation with ADAM was almost seven times faster than the one on the CPU (GPU: 0.48h vs. CPU: 3.39h in average). For the L-BFGS-B implementation on the GPU the line search, which includes several internal loops, turned out to be the bottleneck in terms of computation time.

To determine the reconstruction performance of our model we computed the *normalized mean squared error* (NMSE) for the different optimization methods. On the one hand, overall ADAM performed best on average (Table 1 left). We found an increase of NMSE with higher α-values (i.e. the approximation being closer to a Power EP than to VFE). On the other hand the algorithm converged faster for larger values of α. An α-value of 0.5 seems to be the best trade-of between reconstruction performance and optimization time. Our probabilistic generative motion model is fully real-time capable for computer-animation applications with the GPU implementation, since the computation time is only 0.0035ms per frame. An example movie of the reconstructed data and the corresponding ground truth can be found at `https://hih-git.neurologie.uni-tuebingen.de/ntaubert/highfive_icann`.

6 Conclusion

Combining methods from Bayesian unsupervised learning and inference, we devised a novel real-time-capable method for the realization of Deep Gaussian Pro-

		NMSE			
		ADAM	L-BFGS-B		
			AB	MT	WB
α	0.1	0.0210	0.0725	0.0485	0.0645
	0.25	0.0213	0.0615	0.0419	0.0766
	0.5	0.0309	0.0733	0.0898	0.0643
	0.75	0.0900	0.1357	0.2419	0.2238
	0.9	0.3175	0.3068	0.2408	0.2789
	Average	**0.0961**	0.1300	0.1326	0.1416

		Optimization Time			
		ADAM		L-BFGS-B	
		GPU	CPU	GPU	CPU
α	0.1	0.54h	3.75h	3.34h	6.89h
	0.25	0.53h	3.73h	2.23h	4.38h
	0.5	0.51h	3.66h	1.83h	3.79
	0.75	0.48h	3.47h	1.65h	3.04
	0.9	0.34h	2.32h	0.65h	1.25
	Average	**0.48h**	**3.39h**	1.94h	3.87

Table 1: *Normalized Mean Square Error (NMSE) and optimization time for different learning conditions.* Reconstructions from a model optimized with ADAM results in the smallest reconstruction error (left) and lowest optimization time (right). The optimization time of the line search algorithms are averaged for L-BFGS-B, due to similarity of inner loops.

cess Latent Variable models (DGP-LVMs). Our method combines sparse GP approximations with Stochastic Expectation Propagation, and we provide a GPU implementation of the developed algorithms. To our knowledge, implementations of hierarchical DGP-LVMs have never been developed before. We found that optimization using ADAM results in the highest prediction accuracy of the model and optimization speed. In spite of the sophisticated underlying underlying probabilistic model, we demonstrated that the algorithm is real-time-capable when implemented on the GPU for applications in computer animation.

Future work will extend such architectures by applying a dynamical system in form of a state space model to learn also time evolution. Further we plan to apply our algorithm on bigger data sets, exploiting batch learning on the GPU.

Acknowledgments. This research was funded through The research leading to these results has received funding from HFSP RGP0036/2016; NVIDIA Corp.; BMBF FKZ 01GQ1704, KONSENS-NHE BW Stiftung NEU007/1; DFG GZ: KA 1258/15-1; ERC 2019-SyG-RELEVANCE-856495; SSTeP-KiZ BMG: ZMWI1-2520DAT700.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2007)
2. Brand, M., Hertzmann, A.: Style machines. In: Proc. SIGGRAPH'00. pp. 183–192 (2000)
3. Bui, T.D.: Efficient Deterministic Approximate Bayesian Inference for Gaussian Process models (September) (2017)
4. Quiñonero Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate Gaussian process regression. J. Mach. Learn. Res. 6, 1939–1959 (2005)
5. Chai, J., Hodgins, J.K.: Performance animation from low-dimensional control signals. ACM Trans. Graph. 24(3), 686–696 (2005)
6. Dai, Z., Damianou, A.C., Hensman, J., Lawrence, N.D.: Gaussian Process Models with Parallelization and {GPU} acceleration. CoRR abs/1410.4 (2014)
7. Damianou, A., Lawrence, N.: Deep Gaussian Processes. Proceedings of Machine Learning Research, vol. 31, pp. 207–215. PMLR, Scottsdale, Arizona, USA (2013)
8. Grassia, F.S.: Practical parameterization of rotations using the exponential map. J. Graph. Tools 3(3), 29–48 (Mar 1998)
9. Grochow, K., Martin, S.L., Hertzmann, A., Popovic, Z.: Style-based inverse kinematics. ACM Trans. Graph. 23(3), 522–531 (2004)

10. Harvey, F.G., Pal, C.: Recurrent Transition Networks for Character Locomotion (2019)
11. Hernandez-Lobato, D., Hernandez-Lobato, J.M.: Scalable Gaussian Process Classification via Expectation Propagation. In: Gretton, A., Robert, C.C. (eds.) Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 51, pp. 168–176. PMLR, Cadiz, Spain (2016)
12. Holden, D., Komura, T., Saito, J.: Phase-functioned neural networks for character control. ACM Trans. Graph. 36(4) (Jul 2017)
13. Ikemoto, L., Arikan, O., Forsyth, D.A.: Generalizing motion edits with Gaussian processes. ACM Trans. Graph. 28(1) (2009)
14. Kaiser, M., Otte, C., Runkler, T., Ek, C.H.: Bayesian Alignments of Warped Multi-Output Gaussian Processes. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 6995–7004. Curran Associates, Inc. (2018)
15. Lawrence, N.D.: Probabilistic non-linear principal component analysis with gaussian process latent variable models. Journal of Machine Learning Research 6, 1783–1816 (2005)
16. Levine, S., Wang, J.M., Haraux, A., Popović, Z., Koltun, V.: Continuous character control with low-dimensional embeddings. ACM Trans. Graph. 31(4), 1–10 (jul 2012)
17. Li, Y., Hernandez-Lobato, J.M., Turner, R.E.: Stochastic Expectation Propagation (2015)
18. Minka, T.P.: Expectation Propagation for Approximate Bayesian Inference. In: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence. pp. 362–369. UAI'01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
19. Minka, T.: Power EP. Tech. rep. (2004)
20. Naish-guzman, A., Holden, S.: The Generalized FITC Approximation. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems. vol. 20. Curran Associates, Inc. (2008)
21. Neal, R.: Bayesian Learning for Neural Networks. Ph.D. thesis, Dept. of Computer Science, University of Toronto (1994)
22. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. J. Am. Stat. Assoc. 103, 429–429 (2008)
23. Schmerling, E., Leung, K., Vollprecht, W., Pavone, M.: Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 3399–3406 (may 2018)
24. Schwaighofer, A., Tresp, V.: Transductive and inductive methods for approximate gaussian process regression. In: Advances in Neural Information Processing Systems 15. p. 953–960. MIT Press (2002)
25. Taubert, N., Löffler, M., Ludolph, N., Christensen, A., Endres, D., Giese, M.A.: A virtual reality setup for controllable, stylized real-time interactions between humans and avatars with sparse Gaussian process dynamical models. In: Proc. SAP'13. pp. 41–44 (2013)
26. Titsias, M.: Variational learning of inducing variables in sparse Gaussian processes. Journal of Machine Learning Research - Proceedings Track 5, 567–574 (2009)
27. van der Wilk, M., Dutordoir, V., John, S.T., Artemev, A., Adam, V., Hensman, J.: A Framework for Interdomain and Multioutput Gaussian Processes (2020)
28. Yalamanchili, P., Arshad, U., Mohammed, Z., Garigipati, P., Entschev, P., Kloppenborg, B., Malcolm, J., Melonakos, J.: ArrayFire - A high performance software library for parallel computing with an easy-to-use API (2015)
29. Ye, Y., Liu, C.K.: Synthesis of responsive motion using a dynamic model. Computer Graphics Forum 29(2), 555–562 (2010)
30. Zhao, X., Robu, V., Flynn, D., Dinmohammadi, F., Fisher, M., Webster, M.: Probabilistic Model Checking of Robots Deployed in Extreme Environments (2018)
31. Zhu, H., Rohwer, R.: Information Geometric Measurements of Generalisation (1995)