Int J Comput Vis (2010) 87: 28–52
DOI 10.1007/s11263-008-0204-y

Twin Gaussian Processes for Structured Prediction

Liefeng Bo · Cristian Sminchisescu

Received: 22 July 2008 / Accepted: 24 December 2008 / Published online: 12 February 2009

Abstract We describe twin Gaussian processes (TGP), a generic structured prediction method that uses Gaussian process (GP) priors on both covariates and responses, both multivariate, and estimates outputs by minimizing the Kullback-Leibler divergence between two GP modeled as normal distributions over finite index sets of training and testing examples, emphasizing the goal that similar inputs should produce similar percepts and this should hold, on average, between their marginal distributions. TGP captures not only the interdependencies between covariates, as in a typical GP, but also those between responses, so correlations among both inputs and outputs are accounted for. TGP is exemplified, with promising results, for the reconstruction of 3d human poses from monocular and multicamera video sequences in the recently introduced HumanEva benchmark, where we achieve 5 cm error on average per 3d marker for models trained jointly, using data from multiple people and multiple activities. The method is fast and automatic: it requires no hand-crafting of the initial pose, camera calibration parameters, or the availability of a 3d body model associated with human subjects used for training or testing.

Keywords Structured prediction · Gaussian processes · 3d human pose reconstruction · Feature extraction · Video processing

L. Bo
TTI-Chicago, Chicago, IL 60637, USA
e-mail: blf0218@tti-c.org

C. Sminchisescu (✉)
University of Bonn, INS, Bonn 53115, Germany
e-mail: cristian.sminchisescu@ins.uni-bonn.de

1 Introduction

We study structured prediction problems inspired by computer vision, where we wish to predict a multivariate output from a multivariate input and a joint training set. Here, the input is the image or its descriptor (e.g. a 'bag of words' histogram model that quantizes the occurrence of a feature over the image, for instance a local edge distribution) and the output is a scene representation, an object shape or a three-dimensional human pose. Both the inputs and the outputs are high-dimensional and strongly correlated. At basic level, image features are spatially coherent (nearby pixels more often have similar color, contrast or edge orientation than not), whereas outputs are structured due to physical constraints in the world. For example three-dimensional human poses are constrained at scene level by the ground plane and the location of typical objects, at physical level by gravitation, equilibrium and joint/body limits, and at functional level by the strong body part correlations in motions like walking, running, jumping that have regular or periodic structure, hence, at least locally, low intrinsic dimensionality. Given the recent availability of training data (CMU 2003; Sigal and Black 2006), there has been increasing interest in example-intensive, discriminative approaches to 3d human pose reconstruction, either based on nearest-neighbor schemes (Shakhnarovich et al. 2003; Poppe 2007) or parametric predictors (Rosales and Sclaroff 2002; Agarwal and Triggs 2006; Sminchisescu et al. 2007), trained using images of people and their corresponding 3d ground truth pose. A shortcoming of existing methods is their inability to model interdependencies between outputs. Our work aims to improve the modeling of output correlations by using Gaussian processes and a new prediction criterion.

While Gaussian processes (Rasmussen and Williams 2006) are powerful tools for modeling non-linear dependencies, being potentially relevant for pose reconstruction, most GP models focus on the prediction of a single output. Although generalizations to multiple outputs can be derived by training independent models for each one or tying parameters across dimensions, this fails to account for output correlations in the final predictive model (Weston et al. 2002). This motivates our TGP method that encodes the relations between both inputs and outputs using GP priors. Since samples from two Gaussian processes reflect marginal similarities among them, the idea is to match distributions of inputs and outputs as well as possible. For vision, this emphasizes the objective that similar images should lead to similar 3d percepts. More robustly, generalizing to training *and* testing ensembles, we wish distributions of similar inputs and similar percepts to be close, hence the property to hold on average. Formally, this is achieved by minimizing the Kullback-Leibler divergence between the marginal GP of outputs and observations, both modeled as normal distributions over sets of finite training/testing (i.e. example/query) indices.

1.1 Related Work

Structured Prediction. Research in discriminative learning has recently focused on structured methods that exploit complex internal dependencies among outputs in order to improve prediction. Although the majority of methods address the classification case (Lafferty et al. 2001; Taskar et al. 2004; Tsochantaridis et al. 2004), structured regression has received increasing attention, due to its wide potential applicability. Weston et al. (2002) proposed Kernel Dependency Estimation, where internal dependencies among outputs are captured using kernel principal component analysis, and an input to kernel space mapping is learned for each (uncorrelated) kernel principal dimension separately, using ridge regression. Prediction works by mapping to kernel space using the learned regression model, then solving a pre-image problem to recover the result in the output coordinate system. Cortes et al. (2005) give a conceptually cleaner reformulation of kernel dependency estimation, based on a closed-form solution, where kernel principal component analysis is no longer explicitly required and prediction is made by analytically optimizing a more complex cost function. Along the lines of Cortes et al. (2005), Geurts et al. (2006, 2007) study methods to kernelize the output of regression trees and extend them to ensembles within a gradient boosting framework. Continuous value structured prediction based on both spatial (per timeframe) and temporal (across timeframes) dependencies has been pursued in the probabilistic BM^3E framework (Sminchisescu et al. 2005, 2007), which uses conditional mixture of experts (Bayesian regressors or kernel dependency estimators) for spatial prediction and conditional Markov chains for temporal fusion.

Distribution divergence measures have recently been integrated within cost functions for dimensionality reduction. Stochastic neighbor embedding (SNE) (Hinton and Roweis 2002; Memisevic 2006) explicitly computes the probability of distances between points in data space and estimates low-dimensional, latent coordinates, by matching the corresponding distributions. The Gaussian Process latent variable model (GPLVM) (Lawrence 2005) can also be interpreted as minimizing the KL-divergence between data and latent distributions with respect to the latent variables. However, the methods are designed for dimension reduction, a different problem than supervised learning, as considered here.[1] Unlike existing methods, TGP explores the use of Kullback-Leibler divergence in a supervised setting in order to make predictions in a structured output space.

An alternative is to view the Kullback-Leibler divergence as one possible dependence measure that gives the 'goodness of alignment' between two kernels. Along these lines, in Sect. 3 we comparatively analyze predictive strategies based on other two frequently used kernel dependency measures: kernel target alignment (KTA) (Cristianini et al. 2001a, 2001b) and the Hilbert-Schmidt independence criterion (HSIC) (Gretton et al. 2005a, 2005b). We also extensively compare KTA, HSIC and TGP empirically, and show that TGP is significantly more accurate than either KTA or HSIC.

In recently published work, temporally overlapping with our initial introduction of TGP (Bo and Sminchisescu 2008), Guzman and Holden (2007) also refer to their GP regression method based on a binary switching noise model as 'twinned', but this is very different from the model and goals of TGP. Guzman & Holden's method (2007) can be viewed as a particular instantiation of a two-component GP mixture model with tied kernel parameters and different noise outputs—an instance of the general mixture of GP model of Tresp (2000). Both methods are interesting in their own right, but are substantially different in scope and implementation from the structured prediction goal and cost functions used here. Name similarities are purely coincidental.

3D Human Pose Reconstruction. Given the vastness of the topic, this subsection only highlights research in 3d human pose reconstruction without aiming at a full literature review. Work in 3d human pose reconstruction has focused on both generative and discriminative methods. *Generative*

[1]Formally, these can be viewed as optimizing a reversed relative entropy with linear output kernel as opposed to TGP, which uses regular relative entropy with non-linear output kernel. 'Regular' and 'reverse' should be understood as polar options, not qualifiers—both choices can be justified, problem depending.

methods model the joint distribution of states and observations using, most frequently, an observation model and a state prior, and compute state conditionals using (an often expensive) Bayesian inversion. The key components of generative methods are the inference algorithm, the state prior and the observation model. It is essential both to build an observation model that peaks in regions corresponding to correct percepts in state space, and then find those regions efficiently—ideally the two processes should be synchronized in training (Roth et al. 2004; Sminchisescu and Welling 2007). Successful search algorithms include annealing (Neal 1998; Deutscher et al. 2000; Sidenbladh et al. 2000), hybrid Monte-Carlo and Hyperdynamic sampling (Duane et al. 1987; Choo and Fleet 2001; Sminchisescu and Triggs 2002b) or covariance and kinematic sampling (Sminchisescu and Triggs 2001, 2003). Observation models include edge and silhouettes (Deutscher et al. 2000), intensity based on learned features (Sidenbladh and Black 2001; Roth et al. 2004) or consistent silhouette likelihoods based on bidirectional attraction/overlap terms, as first proposed by Sminchisescu (2002), Sminchisescu and Telea (2002). State priors include dedicated activity models (e.g. walking, running) (Deutscher et al. 2000; Sidenbladh et al. 2000; Wang et al. 2008), physical models (Brubaker and Fleet 2008; Vondrak et al. 2008), nearest-neighbor dynamics (Sidenbladh et al. 2000) and non-linear latent variable models (Sminchisescu and Jepson 2004a; Urtasun et al. 2005; Li et al. 2006; Kanaujia et al. 2007). *Discriminative methods* focus on modeling only the conditional state distribution or, in non-probabilistic cases, its point estimates. Of importance is the design of the image descriptor, the choice of output representation and the predictor. A variety of descriptors based on hierarchical feature design and metric learning methods has been studied in Agarwal and Triggs (2006), Sminchisescu et al. (2007), Kanaujia et al. (2006). Discriminative predictors based on either nearest-neighbors (Shakhnarovich et al. 2003), sparse regression (Agarwal and Triggs 2006), mixture of neural nets (Rosales and Sclaroff 2002), or conditional Bayesian mixtures of experts with temporal smoothness constraints (Sminchisescu et al. 2007; Bo et al. 2008) have been demonstrated to produce competitive results automatically. The combination of discriminative and generative models is promising and several authors have started to study it recently: Sminchisescu et al. (2006) jointly learn coupled generative-discriminative models in alternation and integrate detection and pose estimation in a common sliding window framework, whereas Sigal et al. (2007) focus on human body surface models, estimated by combining discriminative prediction and verification, using particle filters.

To conclude, there has been significant progress in 3d human pose recovery, both conceptually—understanding the limitations of existing likelihood or search algorithms, and deriving improved procedures and conceptually new models and image features to overcome them—and practically, by building systems that can reconstruct shape and perform fine body measurements from images (Sigal et al. 2007), recover 3d human motion from multiple views (Deutscher et al. 2000; Kehl et al. 2005) or reconstruct from complex monocular video footage like dancing or movies, e.g. Run Lola Run (Sminchisescu and Triggs 2003; Kanaujia et al. 2006). The current paper consolidates this conclusion by demonstrating that a novel structured discriminative approach achieves promising results for reconstructing 3d human motion in the HumanEva benchmark (5 cm error per body joint are obtained, on average). The method, which is trained using data from all subjects and all motions, is fast and automatic: it requires no initialization of the human pose, no camera calibration and no knowledge of the body shape parameters of the human subjects during training, or their identity for testing.

Organization. Beyond the current introduction and related work Sect. 1, the paper is structured as follows: Sect. 2 presents the Twin Gaussian Process model (TGP) with both its static (Sect. 2.2) and dynamic versions (Sect. 2.3) (DTGP), and discusses methods for speeding up inference based on nearest neighbors (Sect. 2.4); Sect. 3 analyzes kernel alignment methods and contrasts their cost function and empirical prediction accuracy with the ones of TGP; Sect. 4 presents extensive comparative experiments using the HumanEva datasets, where we test several methods including K-nearest neighbors (KNN), Gaussian Process Regression (GPR), static and dynamic Twin Gaussian processes, TGP and DTGP as well as kernel alignment methods based on the Hilbert Schmidt independence criterion (HSIC), and the Kernel Target Alignment method (KTA).

2 Twin Gaussian Process Model

In structured discriminative prediction, we are given a set of inputs (e.g. for vision, image observations) $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N)$ and a set of the corresponding multivariate outputs (for vision, e.g. 3d pose vectors) $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$, with $\dim(\mathbf{x}) = D$. Here, $\mathbf{R}$ stores the image observations and $\mathbf{X}$ the output vectors columnwise, e.g. $\dim(\mathbf{X}) = D \times N$. The goal is to infer the multivariate output given an input. Since this fits, in principle, the classical regression problem, existing algorithms such as k nearest neighbor regression and Gaussian process regression can be applied, for a start. In the next sections we briefly review Gaussian Process Regression (GPR), identify its limitations for structured prediction, and describe a family of methods referred to as Twin Gaussian Processes (TGP), to address these.

2.1 Gaussian Process Regression (GPR)

A Gaussian process is a collection of random variables, any finite number of number of which have a joint Gaussian distribution. For Gaussian process regression (GPR), the random variables represent the value of the function $f(\mathbf{r})$ for inputs $\mathbf{r}$. GPR assumes $f(\mathbf{r})$ is a zero mean stationary Gaussian process with covariance function $k(\mathbf{r}_i, \mathbf{r}_j)$, encoding correlations between pairs of random variables:

$$\operatorname{cov}\big(f(\mathbf{r}_i), f(\mathbf{r}_j)\big) = K_R\big(\mathbf{r}_i, \mathbf{r}_j\big) \tag{1}$$

One covariance function particularly used is the Gaussian, e.g.: $K_R(\mathbf{r}_i, \mathbf{r}_j) = \exp(-\gamma_r \|\mathbf{r}_i - \mathbf{r}_j\|^2) + \lambda_r \delta_{ij}$, with $\gamma_r \geq 0$ the kernel width parameter, $\lambda_r \geq 0$ the noise variance and δ_{ij} the Kronecker delta function, which is 1 if $i = j$, and 0 otherwise. This kernel function prior constrains input samples that are nearby to have highly correlated outputs.

However, most often we are not interested in drawing random functions from the prior, but to incorporate the constraints from training data. According to the prior, the joint distribution of observed outputs and an unknown test output $\mathbf{x}$ corresponding to a given input $\mathbf{r}$ is:

$$\begin{bmatrix} (\mathbf{X}^{(d)})^\top \\ \mathbf{x}^{(d)} \end{bmatrix} \sim \mathcal{N}_R \left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_R & \mathbf{K}_R^r \\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r}, \mathbf{r}) \end{bmatrix} \right) \tag{2}$$

where $\mathbf{X}^{(d)}$ is the dth row of $\mathbf{X}$ and $\mathbf{x}^{(d)}$ is the dth entity of $\mathbf{x}$, $\mathbf{K}_R$ is a $N \times N$ matrix with $(\mathbf{K}_R)_{ij} = K_R(\mathbf{r}_i, \mathbf{r}_j)$ and $\mathbf{K}_R^r$ is a $N \times 1$ column vector with $(\mathbf{K}_R^r)_i = K_R(\mathbf{r}_i, \mathbf{r})$.

To obtain a meaningful posterior, we have restricted the joint distribution to contain only those functions that are consistent with the observed output. Because the joint is Gaussian, the predictive distribution obtained by conditioning on the observed output, $p(\mathbf{x}^{(d)} | (\mathbf{X}^{(d)})^\top)$ is also Gaussian with mean and variance given by:

$$m(\mathbf{x}^{(d)}) = \mathbf{X}^{(d)} \mathbf{K}_R^{-1} \mathbf{K}_R^r, \tag{3}$$

$$\sigma^2(\mathbf{x}^{(d)}) = K_R(\mathbf{r}, \mathbf{r}) - (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1} \mathbf{K}_R^r \tag{4}$$

Hence, we infer an output given a test input $\mathbf{r}$, using the mean (3) of the predictive distribution.

2.2 Twin Gaussian Processes (TGP)

Gaussian process regression (Rasmussen and Williams 2006) is a standard method to model non-linear input-output dependencies, but it only focuses on the prediction of a single output. Although generalizations to multiple outputs can be derived by training independent models for each, this fails to leverage information about correlations among output components in the predictor (Weston et al. 2002).

Another potential limitation is that GPR cannot handle the multiple solutions corresponding to plausible scene interpretations that frequently occur in hard, inverse 3d from 2d monocular vision problems like pose reconstruction. This gets further complicated in the articulated case (Lee and Chen 1985; Morris and Rehg 1998; Deutscher et al. 2000; Sidenbladh et al. 2000; Choo and Fleet 2001; Sminchisescu and Triggs 2002a, 2003; Sminchisescu and Jepson 2004b). One effective way to deal with ambiguity (multivaluedness) arising in perceptual, 3d from 2d inference problems, is to use mixture of experts or their conditional counterparts (Rosales and Sclaroff 2002; Sminchisescu et al. 2007). Such models are more complex to train and provide a multiplicity of solutions that can be either ranked using gating functions or verified using an observation model (such models have achieved state-of-the art results in HumanEva (Bo et al. 2008)).

While certain (monocular) ambiguities are intrinsic to the problem and appear to be to a large extent unavoidable using current technology,[2] in this work we aim to make 3d percepts less ambiguous through a synergy of expressive image features (we use the current state-of-the art in order to balance the delicate trade-offs of selectivity and invariance), aggressive use of prior knowledge, dynamics (if sequences rather than single images are available) and improved search. Thus, we consider a 'controlled hallucination' setting, and study search based structured prediction and density-sensitive costs that focus on the strongest mode, carrying most of the probability mass in the output distribution. Since covariance functions can encode spatial correlations explicitly, we define covariance functions over outputs in a similar way these were defined over inputs in a GP. This extends existing methods with models of output correlations.

To fix ideas, we first consider an alternative view to Gaussian processes. Since $\begin{bmatrix} (\mathbf{X}^{(d)})^\top \\ \mathbf{x}^{(d)} \end{bmatrix}$ is a sample from a Gaussian distribution (2), $\mathcal{N}_X(\mathbf{0}, \mathbf{K}_{X \bigcup \mathbf{x}})$ where we can estimate the covariance matrix as:

$$\begin{aligned} \mathbf{K}_{X \bigcup \mathbf{x}} &= \begin{bmatrix} (\mathbf{X}^{(d)})^\top \\ \mathbf{x}^{(d)} \end{bmatrix} \begin{bmatrix} \mathbf{X}^{(d)} & \mathbf{x}^{(d)} \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{X}^{(d)})^\top \mathbf{X}^{(d)} & (\mathbf{X}^{(d)})^\top \mathbf{x}^{(d)} \\ \mathbf{X}^{(d)} \mathbf{x}^{(d)} & \mathbf{x}^{(d)} \mathbf{x}^{(d)} \end{bmatrix} \end{aligned} \tag{5}$$

In perception problems, we wish similar inputs to produce similar percepts. Since we know the true Gaussian distribution of the inputs $\mathcal{N}_R$, we measure the offset between the estimated Gaussian distribution of outputs, including a new target, and the homologous input distribution, using Kullback-Leibler divergence:

$$D_{KL}(\mathcal{N}_X \,\|\, \mathcal{N}_R) = -\frac{N}{2} - \frac{1}{2} \log \big|\mathbf{K}_{X \bigcup \mathbf{x}}\big|$$

$$+\frac{1}{2}\mathrm{Tr}\left\{\mathbf{K}_{X\bigcup\mathbf{x}}\begin{bmatrix}\mathbf{K}_R & \mathbf{K}_R^r\\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r},\mathbf{r})\end{bmatrix}^{-1}\right\}$$
$$+\frac{1}{2}\log\left|\begin{bmatrix}\mathbf{K}_R & \mathbf{K}_R^r\\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r},\mathbf{r})\end{bmatrix}\right| \tag{6}$$

The Kullback-Leibler divergence is always non-negative and zero if and only if the two Gaussian distributions have the same covariance. However, the output $\mathbf{x}^{(d)}$ is unknown in this measure. To match the estimated output distribution and the fully observed input one, we estimate output targets (test data) by minimizing the Kullback-Leibler divergence (6):

$$\mathbf{x}^* = \arg\min_{\mathbf{x}^{(d)}}[L(\mathbf{x}^{(d)}) \equiv D_{KL}(\mathcal{N}_X \parallel \mathcal{N}_R)] \tag{7}$$

Invoking matrix determinant and matrix inverse identities and dropping constants, we minimize (8) w.r.t. the output vector $\mathbf{x}^{(d)}$:

$$L(\mathbf{x}^{(d)}) = \mathbf{x}^{(d)}\mathbf{x}^{(d)} - 2\mathbf{x}^{(d)}\mathbf{X}^{(d)}\mathbf{K}_R^{-1}\mathbf{K}_R^r$$
$$- \left[K_R(\mathbf{r},\mathbf{r}) - (\mathbf{K}_R^r)^\top\mathbf{K}_R^{-1}\mathbf{K}_R^r\right]$$
$$\times \log\Big\{\mathbf{x}^{(d)}\mathbf{x}^{(d)} - \mathbf{x}^{(d)}\mathbf{X}^{(d)}$$
$$\times \left[(\mathbf{X}^{(d)})^\top\mathbf{X}^{(d)}\right]^{-1}(\mathbf{X}^{(d)})^\top\mathbf{x}^{(d)}\Big\} \tag{8}$$

This prediction bears certain resemblance to the mean of a GPR. If we drop the third term in (8) to obtain the cost:

$$L_{\sim GPR}(\mathbf{x}^{(d)}) = \mathbf{x}^{(d)}\mathbf{x}^{(d)} - 2\mathbf{x}^{(d)}\mathbf{X}^{(d)}\mathbf{K}_R^{-1}\mathbf{K}_R^r \tag{9}$$

we obtain a quadratic function of $\mathbf{x}^{(d)}$, with analytical solution exactly the mean of a GPR (the variance is different!).

For the multivariate case, if we consider the training outputs along each dimension as samples and assume D samples from their Gaussian distribution (2), we can estimate the covariance matrix $\mathbf{K}_{X\bigcup\mathbf{x}}$ as:

$$\mathbf{K}_{X\bigcup\mathbf{x}} = \frac{1}{D}\begin{bmatrix}\mathbf{X}^\top\\ \mathbf{x}^\top\end{bmatrix}\begin{bmatrix}\mathbf{X} & \mathbf{x}\end{bmatrix} = \frac{1}{D}\begin{bmatrix}\mathbf{X}^\top\mathbf{X} & \mathbf{X}^\top\mathbf{x}\\ \mathbf{x}^\top\mathbf{X} & \mathbf{x}^\top\mathbf{x}\end{bmatrix} \tag{10}$$

As in the 1d case discussed previously, we predict by minimizing the Kullback-Leibler divergence between the estimated output Gaussian and the one of the inputs w.r.t. output $\mathbf{x}$:

$$L(\mathbf{x}) = \mathbf{x}^\top\mathbf{x} - 2\mathbf{x}^\top\mathbf{X}\mathbf{K}_R^{-1}\mathbf{K}_R^r$$
$$- \left[K_R(\mathbf{r},\mathbf{r}) - (\mathbf{K}_R^r)^\top\mathbf{K}_R^{-1}\mathbf{K}_R^r\right]$$
$$\times \log\left[\mathbf{x}^\top\mathbf{x} - \mathbf{x}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{x}\right] \tag{11}$$

Consider the covariance matrix (10) that depends on outputs in the training set. If outputs along each dimension are i.i.d, this can be a good estimate. But if the dimensional outputs are correlated, as in the case considered here, the estimate can be poor. In general, we want the covariance matrix to be as much as possible sensitive to correlations between outputs. This justifies a model with covariance function over outputs:

$$\mathrm{cov}\big(f(\mathbf{x}_i), f(\mathbf{x}_j)\big) = K_X\big(\mathbf{x}_i, \mathbf{x}_j\big) \tag{12}$$

where, e.g. $K_X(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\gamma_x\|\mathbf{x}_i - \mathbf{x}_j\|^2) + \lambda_x\delta_{ij}$, for a Gaussian.

Using this model, we estimate the covariance matrix as

$$\mathbf{K}_{X\bigcup\mathbf{x}} = \begin{bmatrix}\mathbf{K}_X & \mathbf{K}_X^x\\ (\mathbf{K}_X^x)^\top & K_X(\mathbf{x},\mathbf{x})\end{bmatrix} \tag{13}$$

[2] *View of Monocular Ambiguities.* Reconstructing articulated 3d pose from monocular model-image point correspondences is certainly ambiguous, according to geometric (Lee and Chen 1985; Morris and Rehg 1998; Sminchisescu and Triggs 2003) and computational studies (Deutscher et al. 2000; Sidenbladh et al. 2000; Choo and Fleet 2001; Sminchisescu and Triggs 2002a, 2002b; Rosales and Sclaroff 2002; Sminchisescu et al. 2007) of the problem. For other image features and volumetric models, the problem may be more difficult to analyze geometrically, but poses that correspond to reflective placements of limbs w.r.t. the camera rays of sight produce only marginally different image projections, with comparable likelihoods, even under similarity measures based on sophisticated image features (see Deutscher et al. 2001; Sidenbladh et al. 2002; Sminchisescu and Triggs 2002a and computational studies in Deutscher et al. 2000; Sminchisescu and Triggs 2002a, 2005; Sidenbladh and Black 2001; Rosales and Sclaroff 2002; Sminchisescu et al. 2007). Subtle differences do exist between the 'close-in-depth' and 'far-in-depth' configurations, but these would give the 'true' pose some margin only for perfect subject models and image features with no data association errors (it remains unclear under what circumstances this can happen). In principle, shadows can offer supplementary cues, but the key relevant regions seem to be hard to identify in scenes with complex lighting and for unknown people wearing deformable clothing. Even so, different solutions become practically indistinguishable for objects placed further away in depth from the camera. It should also be clear that in a sufficiently general model complexity class, monocular ambiguities can persist in an image sequence, when dynamic constraints are considered (Sminchisescu and Jepson 2004b), and can also persist for models constrained using prior knowledge (Sminchisescu and Jepson 2004a; Rosales and Sclaroff 2002; Sminchisescu et al. 2007) (for illustrations of both static and dynamic ambiguities, see videos at http://sminchisescu.ins.uni-bonn.de/talks/). In the long run, a combination of low-complexity models and appropriate context management may produce solutions—effectively 'controlled hallucinations'—that are unambiguous *in their class* rather than in general. This may turn out to be more effective than a hardline bottom-up approach, where every bit of uncertainty is expelled by judiciously analyzing all 'cues'. In fact, the belief that vision can still proceed despite extensive sets of ambiguities (e.g. bas-relief, pictorial depth, structure-from-motion) is not foreign to researchers in both computer vision and psychophysics (Koenderink and van Doorn 1979; Koenderink 1998; Battu et al. 2007). Notwithstanding, the reader should be aware that we do not only design models that can represent several structured solutions, but also employ complex state-of-the-art image features (HMAX, HoG) and dynamics, see e.g. Sect. 2.3 and Sect. 4, and (Sminchisescu et al. 2007; Kanaujia et al. 2006; Bo et al. 2008). To the extent possible, no component of the system was trivialized.

Based on the estimated Gaussian, we predict by minimizing the KL divergence w.r.t. outputs:

$$\begin{aligned} L(\mathbf{x}) &= K_X(\mathbf{x},\mathbf{x}) - 2(\mathbf{K}_X^x)^\top \mathbf{K}_R^{-1}\mathbf{K}_R^r \\ &\quad - \left[K_R(\mathbf{r},\mathbf{r}) - (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1}\mathbf{K}_R^r\right] \\ &\quad \times \log\left[K_X(\mathbf{x},\mathbf{x}) - (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1}\mathbf{K}_X^x\right] \end{aligned} \tag{14}$$

When designing a cost function for structured prediction, intuitively we want: (1) training input-output pairs whose inputs are far away from the test input should make small contributions to prediction; (2) training input-output pairs whose inputs are near the test input but the corresponding outputs are far away from the test output should again make small contributions to estimates. Otherwise said, the training samples with both similar inputs and similar outputs to a test input (and the output estimate) should contribute significantly to the estimate. By carefully considering the cross term in (14): $(\mathbf{K}_X^x)^\top \mathbf{K}_R^{-1}\mathbf{K}_R^r$ (the first term is constant for a Gaussian kernel and the third does not depend on the test input) or its equivalent form, more convenient for this argument: $\sum_{i=1}^N \sum_{j=1}^N \mathbf{H}_{ij} K_X(\mathbf{x}_i,\mathbf{x}) K_R(\mathbf{r}_j,\mathbf{r})$, where $\mathbf{H} = \mathbf{K}_R^{-1}$, we see how 1) and 2) can be achieved: 1) we can limit the impact of the training pairs that have inputs dissimilar with the test input by setting γ_r large—then corresponding $K_R(\mathbf{r}_j,\mathbf{r})$ are close to zero; 2) we can downgrade the impact of training data with similar inputs but dissimilar outputs w.r.t. the test input-output (estimate) by setting γ_x large (see the third row of Fig. 2). Hence only training data that has both similar inputs and similar outputs with the query may *dynamically*[2] play a key role in the final estimate. In this respect, TGP goes beyond classical regression methods like ridge regression or Gaussian process regression, where training data with similar inputs but dissimilar outputs negatively impact the estimate by pulling in contradictory directions (see the second row in Fig. 2). It is also significantly different from their 'robust' versions (e.g. robust LWR) where a certain resistance to contradictory evidence is achieved procedurally,[3] rather than formally in TGP, at the level of a cost function that is profiled for good performance via cross-validation and hyperparameter optimization, and where estimates are provably convergent.

Training. Learning for TGP requires the inversion of kernel matrices $\mathbf{K}_R$ and $\mathbf{K}_X$ in (14) neither of which depend on the test input, nor the output. Covariances of both inputs and outputs are damped using multiplicative factors of the identity matrix (λ_r, λ_x) in order to improve the stability of inversion.

[2]The output changes during the local optimization-based inference, c.f. (7), and so does the index set of similar outputs swept by its kernel.

[3]Our experience confirms earlier findings by Shakhnarovich et al. (2003) regarding the similar performance of robust/LWR and WKNN.

Prediction. Prediction using TGP requires non-linear optimization of the cost (14). We use a second order, BFGS quasi-Newton optimizer with cubic polynomial line search for optimal step size selection. We initialize using ridge regressors independently trained for each output. This improves both the accuracy and speed of prediction compared to random initialization (Sect. 4 studies the sensitivity of prediction to initialization). For a given test input, we precompute the terms (α and β) that do not depend on output in order to accelerate loss function and gradient evaluations:

$$\begin{aligned} L(\mathbf{x}) &= K_X(\mathbf{x},\mathbf{x}) - 2\alpha^\top \mathbf{K}_X^x \\ &\quad - \beta \log\left[K_X(\mathbf{x},\mathbf{x}) - (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1}\mathbf{K}_X^x\right] \end{aligned} \tag{15}$$

$$\begin{aligned} \frac{\partial L(\mathbf{x})}{\partial x^{(d)}} &= \frac{\partial K_X(\mathbf{x},\mathbf{x})}{\partial x^{(d)}} - 2\alpha^\top \frac{\partial \mathbf{K}_X^x}{\partial x^{(d)}} \\ &\quad - \frac{\beta \log\left[\frac{\partial K_X(\mathbf{x},\mathbf{x})}{\partial x^{(d)}} - 2(\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1} \frac{\partial \mathbf{K}_X^x}{\partial x^{(d)}}\right]}{K_X(\mathbf{x},\mathbf{x}) - (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1}\mathbf{K}_X^x} \end{aligned} \tag{16}$$

where $\alpha = \mathbf{K}_R^{-1}\mathbf{K}_R^r$ and $\beta = K_R(\mathbf{r},\mathbf{r}) - (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1}\mathbf{K}_R^r$. The gradients of kernel function depend on specific choices —for the Gaussian used in the paper, we have $\frac{\partial K_X(\mathbf{x},\mathbf{x})}{\partial x^{(d)}} = 0$ and

$$\frac{\partial \mathbf{K}_X^x}{\partial x^{(d)}} = \begin{bmatrix} -2\gamma_x (x^{(d)} - x_1^{(d)}) K_X(\mathbf{x},\mathbf{x}_1) \\ -2\gamma_x (x^{(d)} - x_2^{(d)}) K_X(\mathbf{x},\mathbf{x}_2) \\ \vdots \\ -2\gamma_x (x^{(d)} - x_N^{(d)}) K_X(\mathbf{x},\mathbf{x}_N) \end{bmatrix} \tag{17}$$

Kullback-Leibler divergence being an asymmetric measure, it is natural to also consider the cost obtained by reversing it:

$$\begin{aligned} &D_{KL}(\mathcal{N}_R \,\|\, \mathcal{N}_X) \\ &\quad = -\frac{N}{2} - \frac{1}{2}\log\left|\begin{bmatrix} \mathbf{K}_R & \mathbf{K}_R^r \\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r},\mathbf{r}) \end{bmatrix}\right| \\ &\qquad + \log\left|\begin{bmatrix} \mathbf{K}_X & \mathbf{K}_X^x \\ (\mathbf{K}_X^x)^\top & K_X(\mathbf{x},\mathbf{x}) \end{bmatrix}\right| \\ &\qquad + \frac{1}{2}\mathrm{Tr}\left(\begin{bmatrix} \mathbf{K}_R & \mathbf{K}_R^r \\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r},\mathbf{r}) \end{bmatrix}\right. \\ &\qquad \left. \times \begin{bmatrix} \mathbf{K}_X & \mathbf{K}_X^x \\ (\mathbf{K}_X^x)^\top & K_X(\mathbf{x},\mathbf{x}) \end{bmatrix}^{-1}\right) \end{aligned} \tag{18}$$

Invoking matrix identities and dropping constants (detailed derivations are given in Appendix), we obtain the cost function corresponding to an 'inverse' TGP (ITGP):

$$I(\mathbf{x}) = -2(\mathbf{K}_R^r)^\top \mathbf{K}_X^{-1}\mathbf{K}_X^x + (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1}\mathbf{K}_R\mathbf{K}_X^{-1}\mathbf{K}_X^x$$

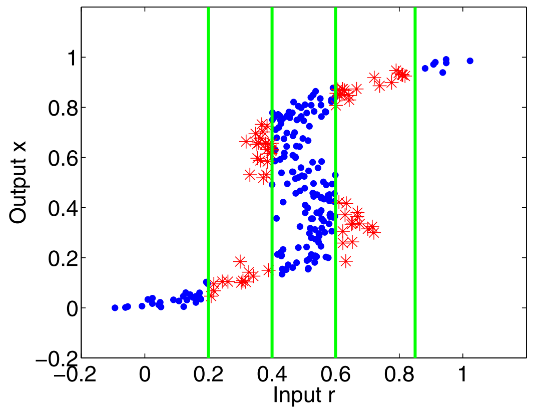

Fig. 1 Training set for a toy problem (predict a 1d output variable x given a 1d control r) consists of 250 values of x generated uniformly in (0,1), for which we evaluate $r = x + 0.3\sin(2x\pi) + \epsilon$ with ϵ drawn from a zero mean Gaussian with standard deviation 0.05. Stars correspond to examples where KNN and GPR suffer from 'boundary/discontinuous effects'. See also Fig. 2 for results that illustrate how different models (KNN, GPR, TGP, ITGP) trained on this dataset behave when tested with 250 equally spaced inputs r in (0,1)

$$\begin{aligned}
&+\left[K_X(\mathbf{x},\mathbf{x}) - (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1}\mathbf{K}_X^x\right]\\
&\times \log\left[K_X(\mathbf{x},\mathbf{x}) - (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1}\mathbf{K}_X^x\right]\\
&-\left[K_X(\mathbf{x},\mathbf{x}) - (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1}\mathbf{K}_X^x\right]\\
&\times \log\left[K_R(\mathbf{r},\mathbf{r}) - (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1}\mathbf{K}_R^r\right] \qquad (19)
\end{aligned}$$

Like TGP, we make predictions by optimizing (19) w.r.t. $\mathbf{x}$.

Intuitions behind different cost functions for both TGP and ITGP are given next. Besides exploiting structured dependencies between outputs, TGP/ITGP give plausible solutions for 'hard multivalue prediction problems' where standard regression methods such as GPR do not work well. To illustrate, we consider a multivalued 1d input-output problem, the S-shape (Bishop and Svensen 2003). In Fig. 1, we generate a training set of 250 values of x uniformly distributed in (0,1) and evaluate $r = x + 0.3\sin(2x\pi) + \epsilon$ with ϵ drawn from a zero mean Gaussian with standard deviation 0.05. For testing, shown in Fig. 2, we generate 250 equally-spaced values with x in (0,1) and evaluate r according to the model. The goal is to predict x given r, in the test set according to the model learned based on the training set.

Although a toy problem, the S-shape models situations that occur in real world applications and datasets with multivariate input-output samples, such as human pose reconstruction, because the output: 1) is multivalued in the middle of S-shape; 2) is discontinuous on the boundary between the univalued and multivalued regions, and 3) the training set is noisy. This epitomizes hard 3d from 2d articulated pose reconstruction problems from monocular images, where ambiguities among multiple separated regions of the state space (Sminchisescu and Triggs 2003) compound with range uncertainty within each (Sminchisescu and Triggs 2001), and persist even in restricted inference settings regularized by temporal (dynamic) constraints (Sminchisescu and Jepson 2004b) or prior knowledge (Sminchisescu and Jepson 2004a). Our experiments using complex image features and real vision datasets show that predictors like TGP or conditional mixture of experts models (BME and its variants) consistently outperform GPR or KNN models—for complementary experiments see Sminchisescu et al. (2007), Bo et al. (2008).

In Fig. 2, we study the performance of 4 learning algorithms: KNN, GPR, TGP and ITGP. The 1NN method has high variance and the highest mean absolute error. All 1-NN estimates are on the curve or very close to it, but they differ from the true outputs—in high-dimensional, sparsely sampled problems with noise, this further degrades. KNN with 5 and 10 nearest neighbors, respectively, works well in the univalued ends of the (0,1) range, but performs erratically in the multivalued zone and on the boundaries between univalued and multivalued regions. Smooth GPR corresponding to small γ_r underperforms w.r.t. other models over the entire input range. In this case, the multivalue training data compounds with the slow varying, inelastic prior, to create the effect of model fitting against a large set of outliers. Other GPR models, corresponding to different γ_r values suffer from boundary effects, with otherwise sensible prediction degrading rapidly on the boundary between univalued and multivalued regions. TGP and ITGP give reasonable predictions, with consistently lower mean absolute error. Both TGP and ITGP prediction directly optimizes the output under training set constraints, hence prediction is not necessary a continuous function of input. This provides a mechanism for learning multivalued relations. Furthermore, as seen in Fig. 2, by selecting a suitable γ_x, TGP and ITGP can make predictions based on data that lives in dominant, high density regions centered at the input, hence it can ignore the impact of far away or low-density data.

Figure 3 shows how the cost functions of TGP and ITGP vary with output. In the univalued region (corresponding to $r = 0.1$, see the top two rows in Fig. 3), TGP's basin of attraction in the neighborhood of the desired solution is large, in fact significantly more so than the one of ITGP. (Certainly more local optima exist in high dimensions, but even there the methods work well, see our Sect. 4.) In the multivalued region corresponding to, say, $r = 0.5$ (see also the bottom two rows in Fig. 3), TGP has only one local optimum whereas ITGP has three. These correctly reflect the possible hypotheses, although for inputs in the middle range, although if we had to favor one solution, we should rather choose the middle branch, corresponding to a substructure

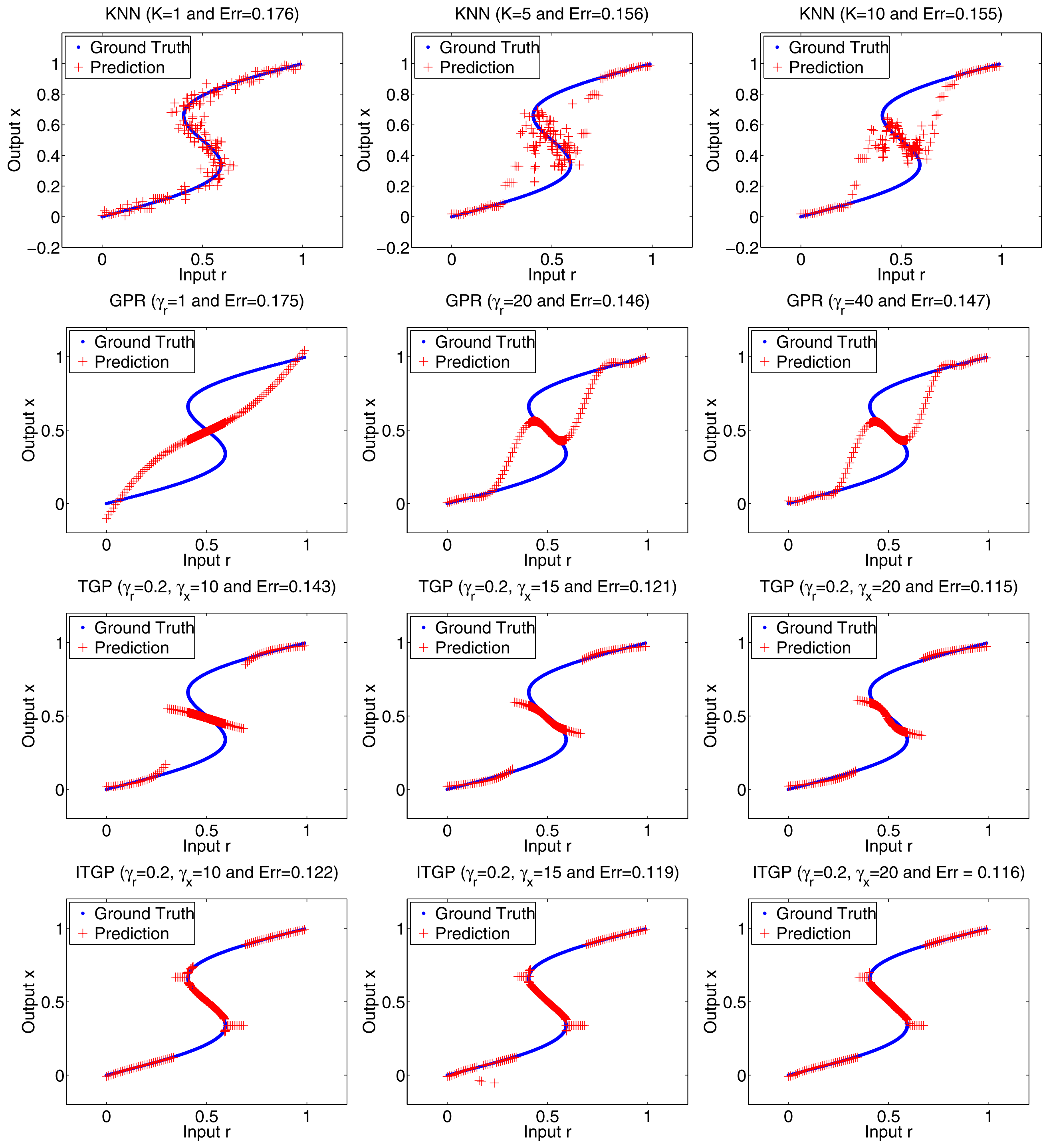

Fig. 2 Prediction of KNN, Gaussian Process Regression (GPR) and Twin Gaussian Processes (TGP, ITPG) on the test set of the toy example described in Fig. 1. Err is mean absolute error. *Top* shows the prediction given by KNN for different number of neighbors, 1, 10 and 20. *Second row* shows predictions of GPR with different kernel parameters $\gamma_r = 1$, 5 and 10. *Third row* shows predictions of TGP with different output kernel width parameters $\gamma_x = 10$, 15 and 20. *Forth row* shows prediction given by ITGP with different kernel widths $\gamma_x = 10$, 15 and 20 (width parameter γ_r is fixed at 0.2, cross-validated, for both TGP and ITGP). The minima of ITGP tend to remain unchanged on the boundary between univalued and multivalued regions, hence prediction is flat there

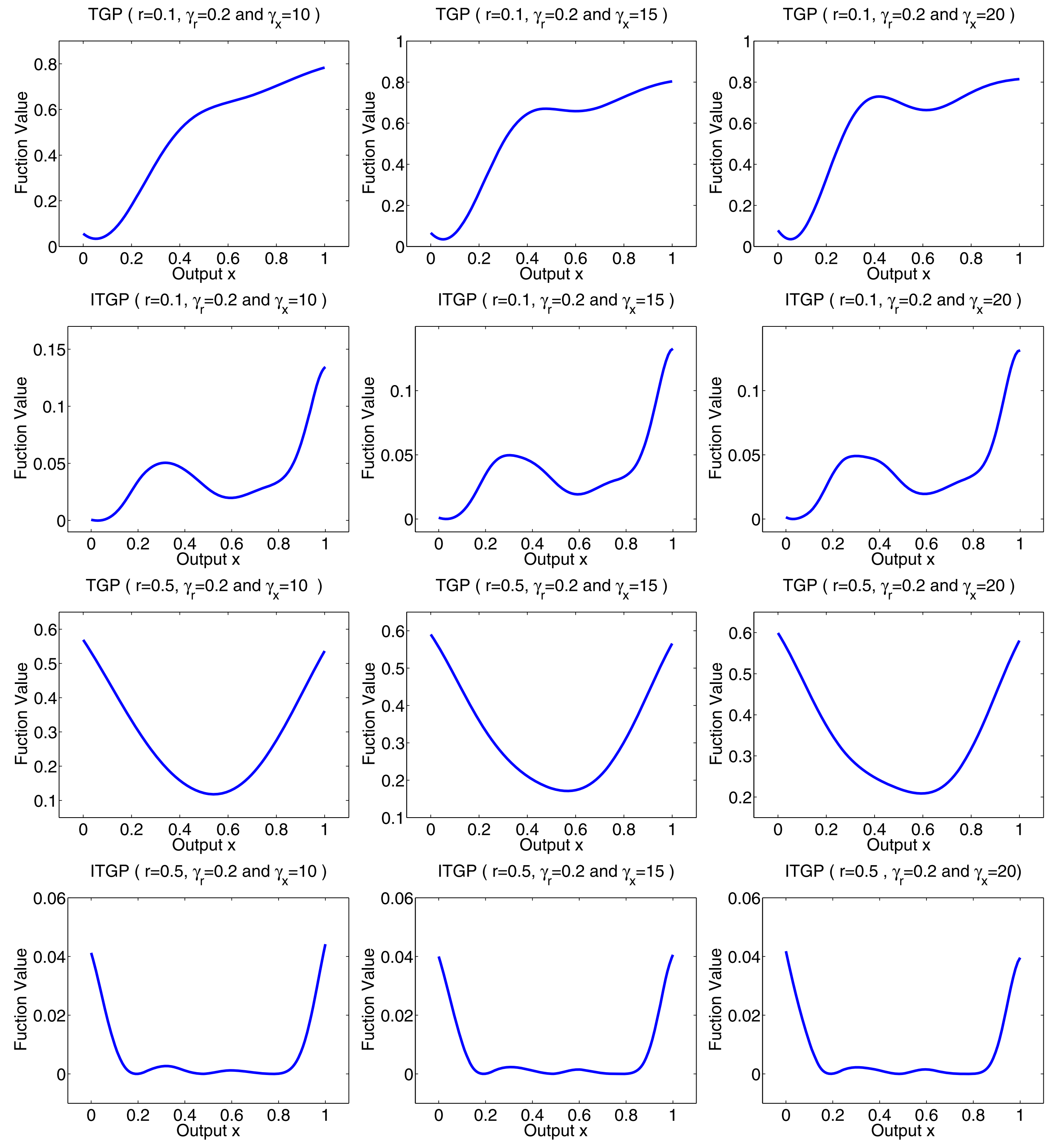

Fig. 3 Costs of TGP and ITGP (14 and 19) as function of output x, for $\lambda_r, \lambda_x = 10^{-4}$ and $\gamma_r = 0.2$. In the two *top rows*, input r is 0.1, for the bottom two, r is 0.5. From *left* to *right*, γ_x is 10, 15, 20. The local optima of TGP correspond to correct outputs for the different inputs given (to check, use Fig. 1 to raise a vertical line at the query r and read the corresponding x values where this intercepts the S shape; see the match with the minima of TGP/ITGP

with most nearby data support. This partly explains why TGP is empirically found to be more reliable than ITGP for human pose reconstruction, although both work well in the toy case.

2.3 Dynamic Twin Gaussian Processes (DTGP)

In sequence modeling or time series problems like tracking, the state vectors link temporally, so the current state not only correlates with the current image (observation), but often bears strong relation with previous states, in particular the most recent. For non-linear dynamical systems, e.g. generative time-series models solved with Kalman filtering or particle filtering, the dependencies appear explicitly in distributions $p(\mathbf{x}_t|\mathbf{r}_t)$ or $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ (Bar-Shalom and Fortman 1988; Isard and Blake 1998). For conditional time series models, the relations surface up as tertiary cliques, and recursive density propagation updates use local distributions $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{r}_t)$ that bind together the reactive effect of measurements and correlations with the previous state (Sminchisescu et al. 2007). Here we take a cost breakdown inspired by generative sequence models (but notice that our search-based structured predictor integrates prior knowledge and search in a way that is very different from generative models), and optimize the current state based not only on TGP that correlate states and observations at the same timestep, but also states at successive timesteps. Analogous to how the dependencies are modeled in autoregression (Bar-Shalom and Fortman 1988; Blake et al. 1999), we consider an auto-TGP. Without loss of generality, we work with a first-order auto-TGP, where the current state is independent of all but the most recent one. Extensions to multiple state dependencies are straightforward, although they imply a more expensive optimization over state subsequences larger than two timesteps, and are more sensitive to phase alignment of training and testing sequences.

Let $\mathbf{X}_t = (\mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_t)$ be the joint set of states, including the current timestep t, where $\mathbf{X}_t$ stores the pose vectors columnwise. For notational compactness, we assume the training set is temporally ordered. (For cases where the training set includes several sequences, the initial state $\mathbf{X}_0$ of each can be adjusted using an observation-sensitive TGP). Given $\mathbf{X}_{t-1}$ and $\mathbf{X}_t$ as training inputs and outputs, the idea in auto-TGP is to predict the current state given the last state using TGP (the observation in the original TGP becomes the state at the previous timestep in the supplementary auto-TGP term).

As for the TGP in the previous section, we specify a joint Gaussian distribution over the training inputs and the test input $\mathbf{x}_{t-1}$:

$$\begin{bmatrix} (\mathbf{X}^d_{t-1})^\top \\ \mathbf{x}^d_{t-1} \end{bmatrix} \sim \mathcal{N}_{X_{t-1}}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{X_{t-1}} & \mathbf{K}^{x_{t-1}}_{X_{t-1}} \\ (\mathbf{K}^{x_{t-1}}_{X_{t-1}})^\top & K_{X_{t-1}}(\mathbf{x}_{t-1}, \mathbf{x}_{t-1}) \end{bmatrix}\right) \tag{20}$$

Using the kernel function defined on outputs, we estimate the covariance matrix

$$\mathbf{K}_{X_t \bigcup \mathbf{x}_t} = \begin{bmatrix} \mathbf{K}_{X_t} & \mathbf{K}^{x_t}_{X_t} \\ (\mathbf{K}^{x_t}_{X_t})^\top & K_{X_t}(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix} \tag{21}$$

Again, we match the estimated covariance matrix of outputs to the one of inputs using Kullback-Leibler divergence. Invoking matrix identities and dropping constants, we minimize (22) with input $\mathbf{x}_{t-1}$ and output $\mathbf{x}_t$, as follows:

$$\begin{aligned} &A(\mathbf{x}_t, \mathbf{x}_{t-1}) \\ &\quad = K_{X_t}(\mathbf{x}_t, \mathbf{x}_t) - 2(\mathbf{K}^{\mathbf{x}_t}_{X_t})^\top \mathbf{K}^{-1}_{X_{t-1}} \mathbf{K}^{\mathbf{x}_{t-1}}_{X_{t-1}} \\ &\qquad + (\mathbf{K}^{x_{t-1}}_{X_{t-1}})^\top \mathbf{K}^{-1}_{X_{t-1}} \mathbf{K}_{X_t} \mathbf{K}^{-1}_{X_{t-1}} \mathbf{K}^{x_{t-1}}_{X_{t-1}} \\ &\qquad - \left[K_{X_{t-1}}(\mathbf{x}_{t-1}, \mathbf{x}_{t-1}) - (\mathbf{K}^{x_{t-1}}_{X_{t-1}})^\top \mathbf{K}^{-1}_{X_{t-1}} \mathbf{K}^{x_{t-1}}_{X_{t-1}}\right] \\ &\qquad \times \log\left[K_{X_t}(\mathbf{x}_t, \mathbf{x}_t) - (\mathbf{K}^{x_t}_{X_t})^\top \mathbf{K}^{-1}_{X_t} \mathbf{K}^{x_t}_{X_t}\right] \\ &\qquad + \left[K_{X_{t-1}}(\mathbf{x}_{t-1}, \mathbf{x}_{t-1}) - (\mathbf{K}^{x_{t-1}}_{X_{t-1}})^\top \mathbf{K}^{-1}_{X_{t-1}} \mathbf{K}^{x_{t-1}}_{X_{t-1}}\right] \\ &\qquad \times \log\left[K_{X_{t-1}}(\mathbf{x}_{t-1}, \mathbf{x}_{t-1}) - (\mathbf{K}^{x_{t-1}}_{X_{t-1}})^\top \mathbf{K}^{-1}_{X_{t-1}} \mathbf{K}^{x_{t-1}}_{X_{t-1}}\right] \end{aligned} \tag{22}$$

Combining observation and dynamic components, we create a dynamic TGP (DTGP), which minimize the tradeoff between an observation-sensitive TGP and an auto-TGP:

$$D(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) = \sum_{t=1}^{T} L(\mathbf{x}_t) + \lambda \sum_{t=1}^{T} A(\mathbf{x}_t, \mathbf{x}_{t-1}) \tag{23}$$

where $\lambda \geq 0$ is a tradeoff parameter.

One option is to optimize the cost sequentially, in a sweep similar to filtering in non-linear, generative time-series models (the cheapest option, computationally): at $t = 1$, only estimate the state based on the observation sensitive TGP term of the cost (22). In subsequent steps fix $\mathbf{x}_{t-1}$ to the previously estimated value and optimize the observation sensitive and dynamic components of the cost w.r.t. $\mathbf{x}_t$. Other optimizations we have tried include the smoothed version where $\mathbf{x}_t$ at all timesteps are estimated jointly, based on an entire observation sequence, or a middle ground, where shorter state subsequences are smoothed (we tested sequences of size 2, 5, 10, 15, 20, 30, 40 frames). All these more expensive strategies improve the estimates to some degree, although in our experiments we found performance to saturate beyond 20 timesteps.

Training. We need to compute matrices $\mathbf{K}_R^{-1}$, $\mathbf{K}^{-1}_{X_{t-1}}$, $\mathbf{K}^{-1}_{X_t}$ and $\mathbf{K}^{-1}_{X_{t-1}} \mathbf{K}_{X_t} \mathbf{K}^{-1}_{X_{t-1}}$, none depending either on the test input or the test output. These account for temporal information in the training set (damping factors λ_r, $\lambda_{x_{t-1}}$ and λ_{x_t} are used, in order to improve the stability of inversion).

Prediction. Like TGP, prediction of DTGP requires non-linear optimization over the desired number of timesteps. We work with the full DTGP (no KNN optimization, see next section Sect. 2.4) and make predictions, once again, by optimizing with a BFGS quasi-Newton method, with cubic polynomial line search, and caching of output-independent matrix blocks. DTGP-KNN is potentially more complex to implement as nearest-neighbors of states being optimized, that forms the input to the pair-wise state auto-TGPs, change in a potentially non-smooth manner during optimization (we have obtained stable results using subgradient methods). Initializing the state sequence using TGP (no dynamic constraints, independent observation-based models at each time-step) often improves the speed of convergence and the accuracy of prediction significantly compared to other strategies.

2.4 Twin Gaussian Process with K Nearest Neighbors

TGP requires $N \times N$ matrix inversions with $\mathcal{O}(N^3)$ training cost and $\mathcal{O}(N^2)$ memory storage, which is impractical for problems with more than 10,000 examples. Sparse approximations can be used to reduce the training and storage cost of TGP (Smola and Schölkopf 2000; Vincent and Bengio 2002). The methods select a representative subset of regressors, dropping training complexity to $\mathcal{O}(Np^2)$, where p is the size of the subset. Since $p \ll N$ in most cases, sparse approximations achieve substantial speedups. They remain applicable here, although a potential limitation is the independence of the representative subset from the run-time query. Another aspect is scaling to very large datasets where to achieve sufficiently accurate approximations even the sparse subset can be too large.

Here, we adopt a conceptually simpler method: find the K nearest neighbors of a test input and estimate TGP on the reduced set (TGPKNN). A similar approach is widely used in local regression (Schaal et al. 1997). The differences from sparse approximations are worth noticing. Sparse methods find a reduced set globally, and this remains unchanged during testing. Instead, the working set of TGPKNN depends on the current test input. This allows us to work with a significantly smaller set that is potentially more relevant locally and likely to generalize better. The trade-off is, as expected, in training vs. testing speed.

The naive version of KNN is easy to implement (compute distances between a test and all training inputs), but computationally intensive for large datasets. Many sophisticated NN search algorithms have been proposed, generally seeking to reduce the number of distance evaluations performed. Recent work (Krauthgamer and Lee 2004; Cortes et al. 2005) shows that the cost of K nearest neighbor queries can be driven down to $\mathcal{O}(\log(N))$ if cover tree data structures are used. We follow this to drop the computational cost of TGPKNN to $\mathcal{O}(\log(N)) + \mathcal{O}(K^3)$, where K is constant (in our experiments, $K = 1000$ worked just fine). Hence, TGPKNN has potential for large training sets in the order of 10^8 examples.

3 Kernel Target Alignment and Hilbert-Schmidt Independence Criterion

An alternative interpretation is to view the Kullback-Leibler divergence as one possible kernel dependency measure that gives the goodness of alignment between two kernels. This raises the question whether other kernel dependency measures can work better than KL for prediction. Kernel Target Alignment (KTA) (Cristianini et al. 2001a, 2001b) is a widely used measure that has been applied to kernel parameter optimization, feature selection, clustering, etc. KTA writes as follows:

$$\mathrm{KTA} = \frac{\mathrm{Tr}(\mathbf{K}_R \mathbf{X}^\top \mathbf{X})}{\sqrt{\mathrm{Tr}(\mathbf{K}_R\mathbf{K}_R)\mathrm{Tr}(\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top\mathbf{X})}} \tag{24}$$

where $\mathbf{K}_R$ is the kernel matrix defined over inputs and $\mathbf{X}$ is the output. Although the original KTA does not use a nonlinear kernel on outputs, one can, in principle replace $\mathbf{X}^\top\mathbf{X}$ by $\mathbf{K}_X$:

$$\mathrm{KTA} = \frac{\mathrm{Tr}(\mathbf{K}_R \mathbf{K}_X)}{\sqrt{\mathrm{Tr}(\mathbf{K}_R\mathbf{K}_R)\mathrm{Tr}(\mathbf{K}_X\mathbf{K}_X)}} \tag{25}$$

Like in the previous section, consider the two kernel matrices defined over an augmented set consisting of a training set, a test input and its target output: $\begin{bmatrix} \mathbf{K}_R & \mathbf{K}_R^r \\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r},\mathbf{r}) \end{bmatrix}$ and $\begin{bmatrix} \mathbf{K}_X & \mathbf{K}_X^x \\ (\mathbf{K}_X^x)^\top & K_X(\mathbf{x},\mathbf{x}) \end{bmatrix}$. Substituting into (25), we obtain

$$\mathrm{KTA} = \frac{\mathrm{Tr}(\mathbf{K}_R\mathbf{K}_X) + 2(\mathbf{K}_R^r)^\top\mathbf{K}_X^x + K_R(\mathbf{r},\mathbf{r})K_X(\mathbf{x},\mathbf{x})}{\sqrt{\mathrm{Tr}(\mathbf{K}_X\mathbf{K}_X) + 2(\mathbf{K}_X^x)^\top\mathbf{K}_X^x + K_X(\mathbf{x},\mathbf{x})K_X(\mathbf{x},\mathbf{x})}} \tag{26}$$

where we dropped the constant term:

$$\sqrt{\mathrm{Tr}(\mathbf{K}_R\mathbf{K}_R) + 2(\mathbf{K}_R^r)^\top\mathbf{K}_R^r + K_R(\mathbf{r},\mathbf{r})K_R(\mathbf{r},\mathbf{r})} \tag{27}$$

Prediction is made by maximizing (26) with respect to $\mathbf{x}$.

Another potentially relevant alignment method is the Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al. 2005a, 2005b). This has the form:

$$\mathrm{HISC} = (N-1)^2\mathbf{Tr}(\mathbf{K}_X\mathbf{H_N}\mathbf{K}_R\mathbf{H_N}) \tag{28}$$

where $\mathbf{H}_N = \mathbf{I}_N - \frac{\mathbf{1}_N}{N}$, $\mathbf{I}_N$ is an $N \times N$ identity matrix and $\mathbf{1}_N$ is an $N \times N$ matrix of all 1s. We consider an augmented set (training set, test input and target output):

$$\mathrm{HSIC} = N^2\Big[\mathrm{Tr}(\overline{\mathbf{K}}_R\mathbf{K}_X) + 2(\overline{\mathbf{K}}_R^r)^\top\mathbf{K}_X^x + \overline{K}_R(\mathbf{r},\mathbf{r})K_X(\mathbf{x},\mathbf{x})\Big] \tag{29}$$

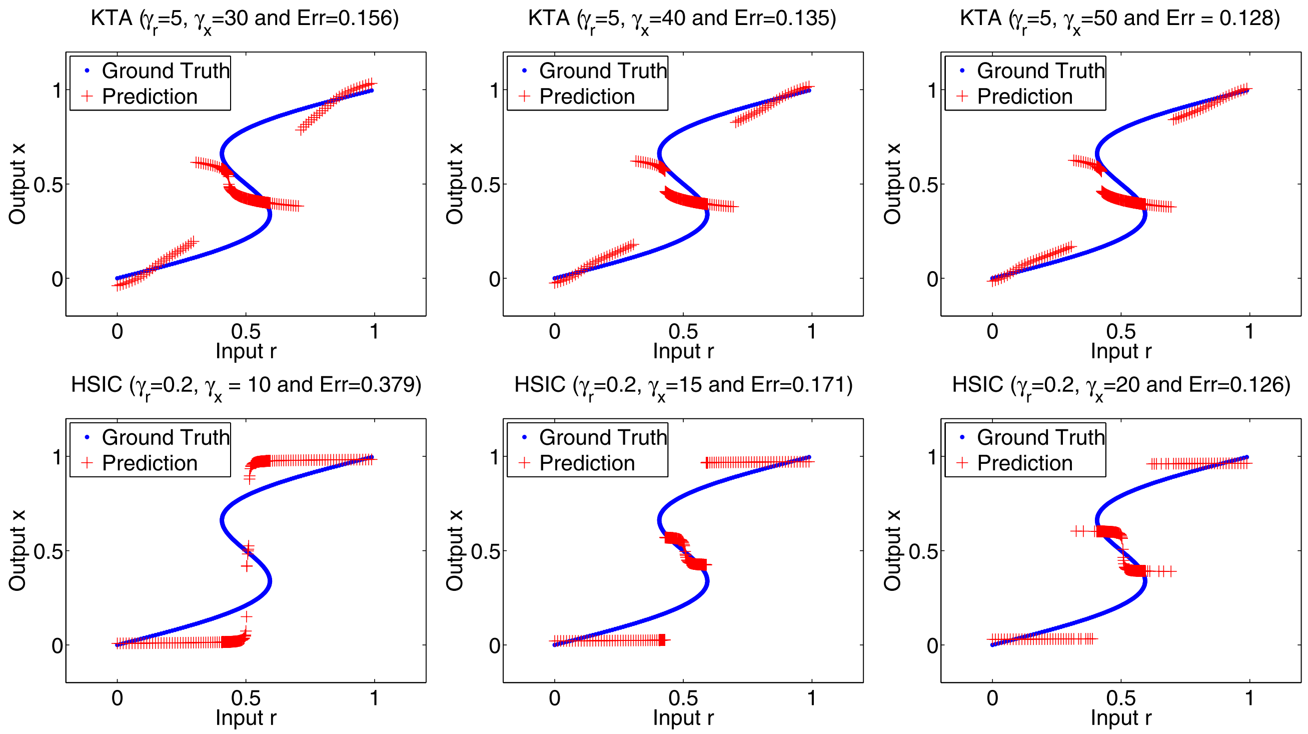

Fig. 4 Prediction of kernel target alignment and Hilbert-Schmidt independent criterion on the test set of the toy problem described in Fig. 1. Err is the mean absolute error. *Top row* shows the prediction given by KTA with width parameters $\gamma_x = 30, 40, 50$. *Bottom row* gives HSIC predictions for widths $\gamma_x = 10, 15, 20$. The parameter γ_r is cross-validated to 5 and 0.2 for KTA and HSIC, respectively

where

$$\begin{bmatrix} \overline{\mathbf{K}}_R & \overline{\mathbf{K}}_R^r \\ (\overline{\mathbf{K}}_R^r)^\top & \overline{K}_R(\mathbf{r},\mathbf{r}) \end{bmatrix} = \mathbf{H}_{N+1} \begin{bmatrix} \mathbf{K}_R & \mathbf{K}_R^r \\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r},\mathbf{r}) \end{bmatrix} \mathbf{H}_{N+1} \tag{30}$$

Figure 4 shows predictive results of KTA and HSIC on the toy problem described in Fig. 1. Like TGP, KTA and HSIC can deal with discontinued, multivalued regimes (the prediction can change abruptly on the boundary between univalued and multivalued regions) although KTA and HSIC appear to be less accurate than TGP. Our experiments for human motion reconstruction in Sect. 4 suggest that TGP is significantly more accurate than KTA and HSIC. TGP is closely related to Gaussian processes which are successful predictors for standard cases, whereas algorithms like KTA or HSIC were primarily designed for parameter selection or independence testing, not for prediction. For instance, the cost function of HSIC is not necessarily maximized when the two kernel matrices are identical, which makes prediction by HSIC somewhat inaccurate even in regions where there are consistently single solutions.

4 Experiments

We evaluate the performance of TGP and its variants on the HumanEva-I dataset (Sigal and Black 2006), which consists of 4 subjects doing 6 predefined actions: Walking, Jogging, Throw-catch, Gestures, Boxing and Combo (walking followed by jogging and then balancing on each one of the two feet). Video data and ground truth motion of the body are captured using a marker-based motion capture system and synchronized in software. Combo motions of all subjects and all motions of one subject are withheld. Table 1 summarizes the structure of the training set (frames with invalid MoCap were removed). For details on HumanEva, see the report by the authors (Sigal and Black 2006). For some of the experiments we use image features extracted from the first camera, together with additional training samples obtained from the second and the third camera, using virtual rotations of their pose sets into one common monocular frame (Poppe 2007). In this way we triple the 'monocular' training set (rather than triple the descriptor size). Models obtained in this way are referred in our tables as 'C1Add'. We also run (truly) monocular experiments where only data from one camera is used, e.g. 'C1'. In other experiments, image features extracted of images from 3 cameras are combined in a (longer $3\times$) descriptor, 'C1+C2+C3'.

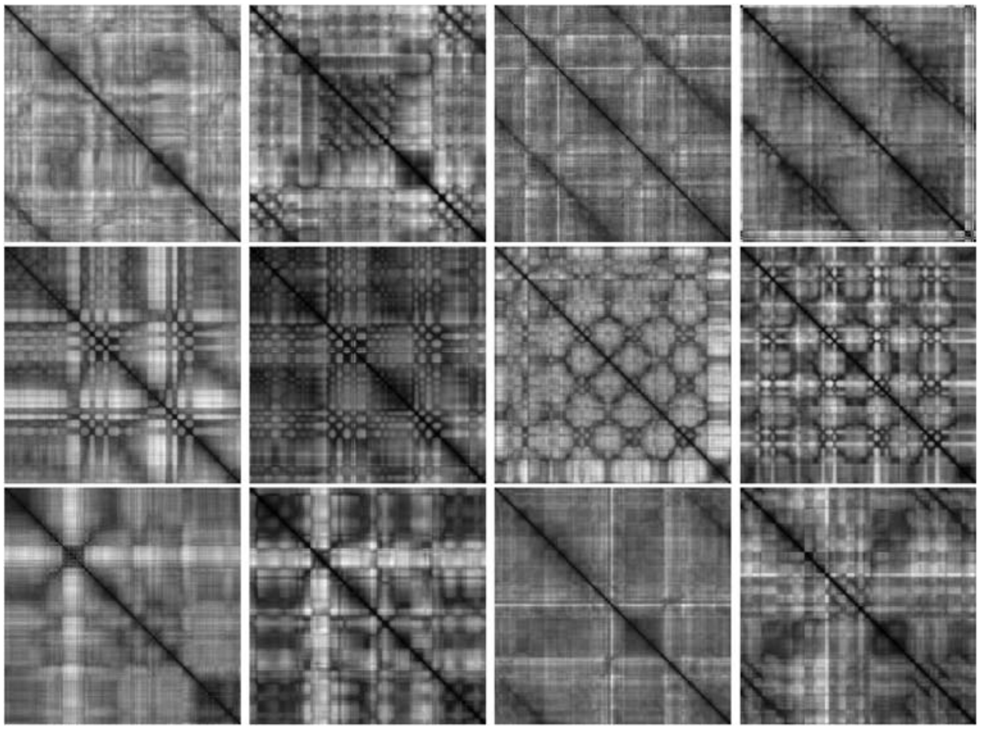
Fig. 5 Affinity matrices for different motions executed by Subject 2 in HumanEva. These correspond to features extracted from images of the first camera, on temporally ordered test sequences, and are computed based on the Euclidean distance between image features (darker means more similar). The *first* and *third columns* show affinities for HMAX, the *second* and *fourth* for HoG. From *top* to *bottom*: Walking, Jogging and Gesture sequences

Table 1 Size of training set for HMAX and HoG features on HumanEva-1. HMAX features are computed as in Kanaujia et al. (2006) and we use the HoG set of Poppe (2007) for comparisons. There are small variances in the number of training samples for HMAX and HoG, caused by different frames of invalid Mocap removed by Poppe (2007), Kanaujia et al. (2006)

Features	Action	Subject 1	Subject 2	Subject 3	Total
HMAX	Walking	1197	870	931	2998
	Jogging	597	789	834	2220
	Throw/Catch	217	1011	0	1228
	Gestures	795	893	1096	2784
	Box	783	652	1015	2450
	Total	3589	4215	3876	11680
HoG	Walking	1176	876	895	2947
	Jogging	439	795	831	2065
	Throw/Catch	217	806	0	1023
	Gestures	801	681	214	1696
	Box	502	464	933	1889
	Total	3135	3622	2873	9630

Image Features. The backgrounds are known and fairly uniform, hence silhouettes can be computed. We use non-parametric background models and adaptive thresholding procedures for better accuracy (Elgammal and Lee 2004). We consider two types of state-of-the art image features: HMAX and histogram of oriented gradients (HoG). HoG (Lowe 2004; Dalal and Triggs 2005) is a block representation that concatenates SIFT descriptors (histograms of gradient orientation) extracted on a regular grid placed in a putative detection window, in our case, the bounding box of a silhouette. HMAX (Deutscher et al. 2000) is a hierarchical, multilayer model inspired by the feedforward processing pipeline of the visual cortex. It alternates layers of template matching (simple cell) and max pooling (complex cell) operations in order to build representations that are increasingly invariant to scale and translation. Simple layers use convolution with local filters (template matching against a set of prototypes), in order to compute higher order (hyper)features, whereas complex layers pool their afferent units over limited ranges, using a MAX operation, in order to increase invariance. Rather than learning the bottom layer, the model uses a bank of Gabor filter simple cells, computed at multiple positions, orientations and scales. Higher layers use simple cell prototypes, obtained by randomly sampling descriptors in the equivalent layer of a training set (k-means clustering can also be used), hence the construction of the hierarchical model has to be done stage-wise, bottom-up, as layers become available.

Affinity matrices for HMAX (Serre et al. 2007) and HoG (Lowe 2004; Dalal and Triggs 2005; Sminchisescu et al. 2006) are shown in Fig. 5. The strongly similar subdiagonal bands confirm some of the ambiguities associated to the selected image features, e.g. at half-cycles for walking parallel to the image plane. For an alternative view see also Fig. 6.

Pose Representation. Human pose ($\mathbf{x}$) is represented as 60d vectors of three-dimensional body joint positions (20 joints or markers each having X, Y and Z co-ordinates) using 'torsoDistal' as root joint. All poses are preprocessed

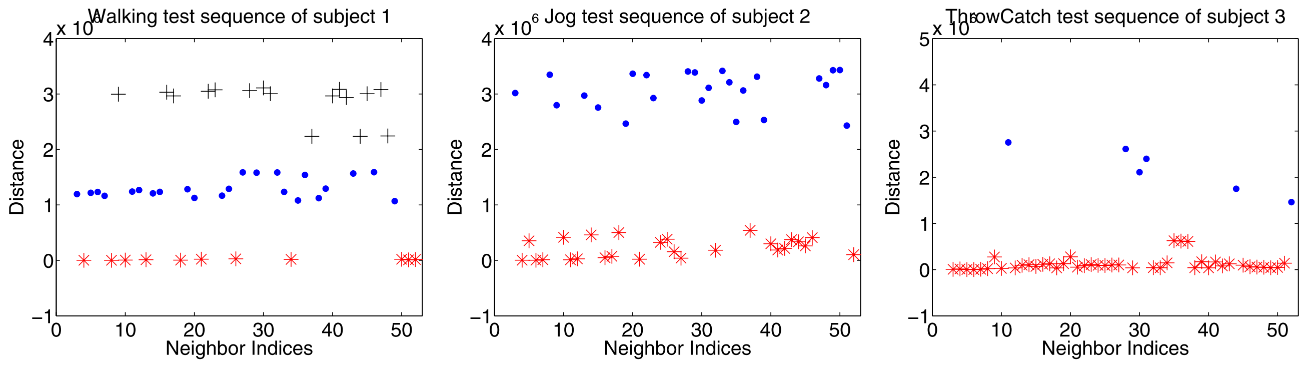

Fig. 6 Ambiguities in HumaEva-I. We select a test image according to the motion shown in the title and find its 50 nearest neighbors based on HoG. We record 3d poses associated with the 50 images and plot the Euclidean distance between the pose of the nearest image and the other 49 poses (notice that no ground truth is used, as we don't have it). Notice that poses cluster at multiple well separated levels

by subtracting the root joint location from all the other joint positions in every frame. This representation is not particularly good for learning, as perceptually identical poses of people with different body proportions tend to be encoded somewhat differently (joint angles and skeletons could have been more appropriate, but they are not currently available for HumanEva), but our normalization by root subtraction appears to largely palliate this. In fact, it turns out that TGP is not sensitive to different body proportions or the existence of accurate kinematic or volumetric human body models: methods trained on samples captured from all subjects do not perform significantly worse than those trained on individual subjects (see our Tables 4 and 5). This degree of robustness is harder to achieve with a generative model where alignment (observation modeling) constraints pay an important role, and reliable inference is often contingent both on a 3d human model well adapted to the body proportions of each human subject and on good camera calibration, set aside model initialization. These stringent requirements are no longer required for TGP.

Error Metric. We report the error between the estimated $\bar{\mathbf{x}}$ and the ground truth pose $\mathbf{x}$ according to the measure suggested by Sigal and Black (Sigal and Black 2006), and used by the online evaluation system:

$$D(\mathbf{x}, \bar{\mathbf{x}}) = \frac{1}{M}\sum_{i=1}^{M} \|m_i(\bar{\mathbf{x}}) - m_i(\mathbf{x})\| \tag{31}$$

where $m_i(\mathbf{x}) \in \mathbb{R}^3$ is a function that extracts the three dimensional coordinates of the ith joint position, M is the number of the joint positions for each pose and $\|\cdot\|$ is the Euclidean distance. For a motion sequence of T frames, we report the average joint position error as:

$$\mathrm{Err}_{seq} = \frac{1}{T}\sum_{i=1}^{T} D(\mathbf{x}_i, \bar{\mathbf{x}}_i) \tag{32}$$

4.1 Initialization of TGP

For pose estimation, TGP optimizes a non-convex cost function (14), and a suitable initialization is desirable in order to avoid shallow local optima. We study the impact of three different initialization methods: K nearest neighbors (KNN), ridge regression (RR) with models trained for each output dimension independently and ground truth (GrTruth), which we don't have in general, but represents a gold standard to check against model bias. In this experiment we use the validation set as test set, but for methodological consistency we divide each train and validation sequence into approximately equal chunks. The new 'training and validation set' consists of the second half of all motions sequences from all subjects and the new 'test set' consists of their first half.

We set $\lambda_r = \lambda_x = 10^{-4}$ (inversion of kernel matrices is involved in TGP and mild damping with a multiplicative factor of a diagonal matrix improves stability), and to cross-validate we grid search for γ_r and γ_x over suitable ranges. The results are shown in Table 2. As expected, ground truth initialization works best among all three methods. RR outperforms KNN in most cases with estimates based on RR close to the ones obtained from ground truth. For example, for HoG (C1+C2+C3), RR initialization gives the same average error as the ground truth for Subjects 2 and 3, and only 0.4 mm higher average error for Subject 1.

4.2 Evaluation of Dynamic TGP

This section compares the static and dynamic versions of TGP. We initialize TGP using independent output RR and optimize the kernel parameters γ_r and γ_x using cross validation. We initialize DTGP with the estimated pose of the observation-sensitive TGP, obtained independently, at each timestep—this usually gives good starting points and accelerates convergence. The observation-dependent component of DTGP is set as in the static model (TGP) and the kernel parameter of the dynamic, auto-TGP, γ_A is set to γ_x. We

Table 2 Evaluation of different initializations using HMAX and HoG on HumanEva-I. Models are trained on data from all subjects. The lowest error is shown in bold. In the table, '/' show that the values are not available (no training samples); 'Average' gives the averaged error for the different motions of the same subject; 'C1' means image feature are computed only from the first camera; 'C1+C2+C3' means image features from three cameras are combined in a single descriptor; 'C1Add' means that image features from all three cameras are used as if they were captured by a single camera by transforming the 3d coordinates of poses in those cameras using their displacements relative to the first camera, thus tripling the standard monocular training set size (Poppe 2007). KNN, RR and GrTruth indicate that TGPKNN is initialized with K nearest neighbors, ridge regression and ground truth, respectively

Features	Motions	Subject 1			Subject 2			Subject 3		
		KNN	RR	GrTruth	KNN	RR	GrTruth	KNN	RR	GrTruth
HMAX	Walking	43.9	43.7	41.8	25.4	25.4	25.4	70.9	68.9	64.5
(C1)	Jogging	60.2	57.6	54.7	53.7	53.2	52.7	37.5	37.4	36.4
	Gestures	11.9	11.9	11.9	100.8	102.1	87.3	27.7	27.7	27.8
	Box	52.9	52.9	52.7	104.5	82.5	78.6	65.2	68.9	61.4
	ThrowCatch	136.8	137.8	129.5	89.9	83.9	78.7	/	/	/
	Average	61.1	60.9	**58.1**	74.9	69.4	**64.5**	50.3	50.7	**47.5**
HMAX	Walking	33.2	33.2	33.2	22.6	22.6	22.6	67.4	61.1	57.5
(C1+C2+C3)	Jogging	47.1	47.1	47.1	47.1	46.9	46.7	32.7	31.9	31.8
	Gestures	10.0	10.0	10.0	96.5	86.4	82.2	25.9	25.7	24.1
	Box	49.6	49.8	46.8	63.2	57.8	57.5	53.5	56.8	52.7
	ThrowCatch	127.3	126.0	117.9	71.9	71.3	69.9	/	/	/
	Average	53.4	53.2	**51.0**	60.3	57.0	**55.8**	44.9	43.9	**41.5**
HoG	Walking	45.4	43.4	42.9	28.3	28.3	28.3	62.3	62.5	62.3
(C1)	Jogging	55.1	55.1	55.1	43.2	42.5	42.5	37.4	37.4	37.4
	Gestures	11.8	11.8	11.8	68.7	68.3	68.3	74.6	74.5	74.5
	Box	42.5	42.5	42.5	64.0	64.0	62.8	69.3	70.4	67.4
	ThrowCatch	137.8	137.8	136.9	81.5	78.7	70.6	/	/	/
	Average	58.5	58.1	**57.8**	57.1	56.4	**54.5**	60.9	61.2	**60.4**
HoG	Walking	29.1	29.1	29.1	19.2	19.2	19.2	46.8	47.0	46.8
(C1+C2+C3)	Jogging	43.7	43.6	43.7	32.6	32.6	32.6	28.9	28.9	28.9
	Gestures	8.9	8.9	8.9	58.2	58.2	58.2	66.9	66.8	66.9
	Box	35.6	35.6	35.6	49.9	49.9	49.9	57.9	47.8	47.7
	ThrowCatch	145.2	116.7	114.8	60.9	60.9	60.9	/	/	/
	Average	52.5	46.8	**46.4**	**44.2**	**44.2**	**44.2**	50.1	**47.6**	**47.6**
HoG	Walking	31.4	30.8	30.6	25.4	25.4	25.4	54.5	51.4	49.9
(C1Add)	Jogging	50.2	49.5	45.1	32.3	32.3	32.3	31.6	31.6	31.6
	Gestures	11.9	11.9	11.9	75.7	75.8	75.7	119.2	118.2	78.7
	Box	44.7	44.7	44.6	64.7	64.3	64.3	72.3	67.6	59.8
	ThrowCatch	134.2	137.1	125.9	76.5	72.5	71.0	/	/	/
	Average	54.5	54.8	**51.6**	54.9	54.0	**53.7**	69.4	67.2	**55.4**

choose the remaining two free parameters: step length T and tradeoff λ using CV (here $T = 20$).

We show the average joint position error of TGP and DTGP as function of frame in Fig. 7. In general, DTGP improves over TGP by 5–20%. The error of DTGP for the Jogging motion of Subject 1 (top), Walking motion of Subject 2 (middle) and Box motion of Subject 3 (bottom) are 77.4 mm, 22.6 mm and 60.5 mm, respectively whereas those of TGP are 81.3 mm, 27.9 mm and 65.8 mm. The use of temporal constraints makes DTGP more robust to outliers or poorly segmented silhouettes compared to TGP. E.g., the standard deviation of DTGP for the Box motion of Subject 3 is 26.4 mm whereas the one of TGP is 52.5 mm. In practice, the superior performance over TGP does not maintain its margin for models trained with data from multiple subjects on the same motion, or for even more complex models where different motions from multiple subjects are combined. In such cases, we notice an attenuation in the dynamic

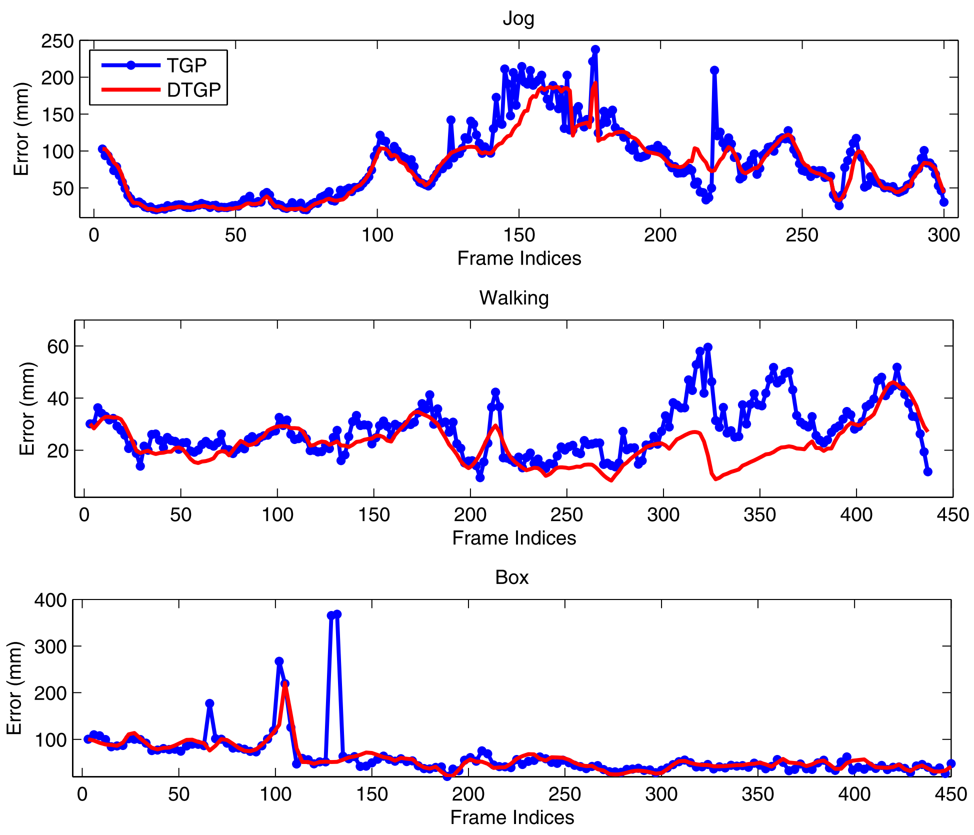

Fig. 7 TGP with and without dynamics. HMAX features extracted on images from the first camera are used for evaluation. Models are trained on a single motion of a single subject and tested on the same motion. Since the size of training set is relatively small, we run full TGP and DTGP models without KNN preprocessing. *Top:* Jogging motion of Subject 1. *Middle*: Walking motion of subject 2. *Bottom*: Box motion of Subject 3. DTGP not only gives lower average error, but also significantly lower maximum error

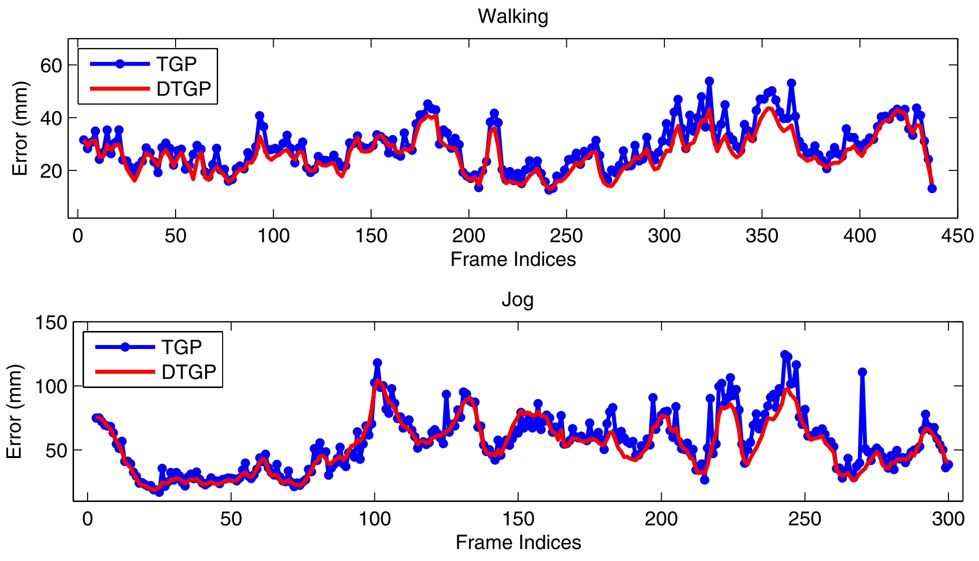

Fig. 8 TGP and DTGP (no KNN optimization) with and without dynamics, for complex training scenarios (both use HMAX). *Top* show results for models trained on *the same motion type* with data from multiple subjects. *Bottom* show models trained jointly using *all motions from all subjects*. DTGP are still marginally better and many outlier predictions appear to be smoothed out, but the dynamic boost is no longer as significant as for the subject specific models illustrated in Fig. 7, which is what we usually fear of most motion models. This likely happens because different subjects execute activities at different speeds and with different styles, and combining multiple motions further increases the chance to 'pick the wrong branch' in the dynamic model

boost offered by DTGP (Fig. 8), which is what we fear in the first place of most dynamical models. This likely happens because different subjects execute activities at different speeds and with different styles, and combining multiple motions puts additional pressure on the fixed trade-off parameter that weights the observation and dynamic-sensitive terms in DTGP (extensions to more complex non-constant trade-offs are currently studied).

Table 3 Average errors for ITGPKNN, HSICKNN and KTAKNN on the validation set of HumanEva-I, with lowest error in bold. In the table, '/' marks values that are not available (no training samples); 'Average' gives average error for different motions of the same subject; 'C1' indicates that models are trained only with HoG features extracted form only the first camera. ITGPKNN, HSICKNN and KTAKNN use 800 nearest neighbors, and are initialized using the output of ridge regression (RR) with univariate predictors trained independently for each dimension

Features	Motions	Subject 1			Subject 2			Subject 3		
		ITGPKNN	KTAKNN	HSICKNN	ITGPKNN	KTAKNN	HSICKNN	ITGPKNN	KTAKNN	HSICKNN
HMAX	Walking	67.6	89.5	90.7	47.7	71.8	82.0	85.7	102.1	124.2
(C1)	Jogging	84.4	95.0	101.0	83.3	95.7	94.2	71.4	84.9	134.6
	Gestures	13.0	24.4	26.7	117.9	118.9	136.4	30.3	43.8	39.2
	Box	64.6	79.1	132.1	99.3	106.1	153.5	85.5	101.7	112.0
	ThrowCatch	143.6	152.6	156.8	95.4	102.1	93.6	/	/	/
	Average	**74.6**	88.1	101.5	**88.7**	98.9	111.9	**68.2**	83.1	102.5
HoG	Walking	71.7	92.0	95.1	46.5	76.8	85.5	80.2	95.9	116.7
(C1)	Jogging	72.6	82.7	97.5	72.4	89.3	93.2	73.2	86.4	123.4
	Gestures	12.3	23.7	25.8	95.2	95.0	126.9	36.2	45.5	39.3
	Box	55.8	75.9	121.1	96.8	108.7	121.7	81.8	102.2	109.3
	ThrowCatch	149.7	163.8	166.4	91.8	98.8	92.6	/	/	/
	Average	**72.4**	87.6	101.2	**80.5**	93.7	104.0	**67.9**	82.5	97.2

4.3 ITGP, KTA and HSIC

This section studies the performance of ITGP, KTA and HSIC, described in Sect. 3. These work similarly with TGP (the output is non-linearly optimized for prediction, initialized using RR, etc.), except that different cost functions used. The regularizers $\lambda_r = \lambda_x = 10^{-4}$ and kernel parameters γ_r and γ_x are optimized using cross validation. The average error for ITGP, KTA and HSIC is given in Table 3. In general, ITGP is superior to KTA and HSIC. Compared with Table 2, TGP significantly outperforms ITGP, KTA and HSIC, a finding consistent with our earlier analysis. Notice that it is computational feasible to run KTA and HSIC without KNN, but our experience suggests that combining KTA and HSIC with KNN gives lower errors compared to models trained on the full dataset.

4.4 Evaluation of TGP

We compare TGP with weighted K nearest neighbor (WKNN) and Gaussian Process Regression (GPR), using HumanEva's online evaluation system on the test set (the ground truth poses of test videos are withheld by the database creators). WKNN locates the K closest training samples between a test input and all training inputs and predicts as the weighted sum of their corresponding outputs:

$$\mathbf{x}^* = \frac{\sum_{j\in\mathcal{N}_k(\mathbf{r})} W(\mathbf{r},\mathbf{r}_j)\mathbf{x}_j}{\sum_{j\in\mathcal{N}_k(\mathbf{r})} W(\mathbf{r},\mathbf{r}_j)} \tag{33}$$

where $\mathcal{N}_k(\mathbf{r})$ is the index set of the K nearest neighbors for the test input $\mathbf{r}$ and $W(\mathbf{r},\mathbf{r}_j)$ is the similarity measure (here Euclidean distance) between $\mathbf{r}$ and $\mathbf{r}_j$. A different, yet related approach for estimation is locally weighted regression (LWR). LWR is an extension of K nearest neighbors where the target is approximated locally by a particular function class $f(\mathbf{r};\beta)$. The parameters β are learned by minimizing a weighted cost function defined over the neighbors of the test input:

$$\beta_r^* = \operatorname{argmin}_{\beta_r} \sum_{j\in\mathcal{N}_k(\mathbf{r})} W(\mathbf{r},\mathbf{r}_j)\|\mathbf{x}_j - f(\mathbf{r}_j;\beta_r)\|^2 \tag{34}$$

A common selection for $f(\mathbf{r};\beta)$ is a low-order polynomial, constant or linear. When $f(\mathbf{r};\beta)$ is constant, LWR is equivalent to weighted KNN. We evaluated LWR with linear $f(\mathbf{r};\beta)$ on the validation set and our experimental results indicated this model has similar performance with WKNN (in our tests, it was actually slightly lower), which is also consistent with the observations of Shakhnarovich et al. (2003). Therefore in the sequel we show comparisons with the faster WKNN.

Gaussian Process Regression (GPR), described in Sect. 2.1 is also tested. We determine optimal hyperparameters γ_r and λ_r by maximizing the marginal likelihood, the standard GPR hyperparameter selection scheme. For the tripled training set (C1Add), we use Gaussian process regression with K nearest neighbors, which works similarly with TGPKNN, by locating the K nearest neighbors of a test input and performing GPR.

Testing times for WKNN, GPR and TGPKNN are given in Fig. 9, with close timings of WKNN and GPR. This is not surprising because similarly to WKNN, the most time-consuming operation in GPR is the evaluation of the dis-

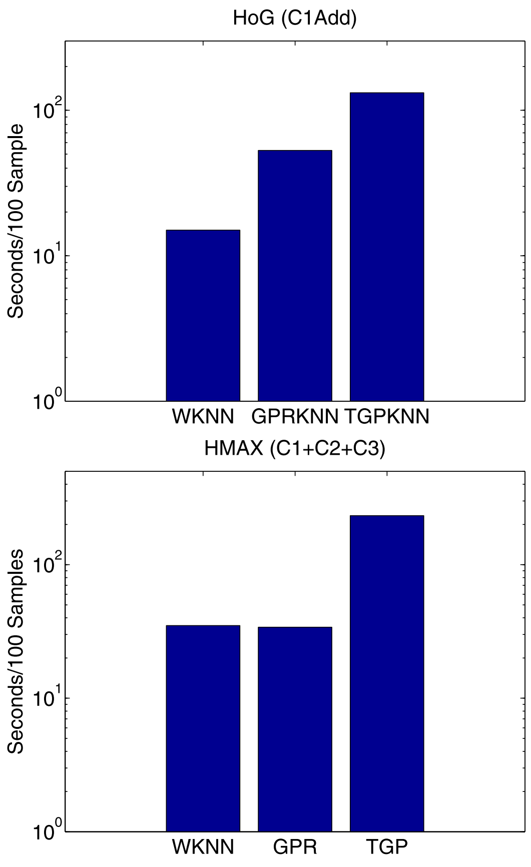

Fig. 9 Comparison of test time (in seconds) for WKNN, GPR, GPKNN and TGPKNN. All models are trained on the full dataset. The *upper plot* shows results on HoG, tripled for camera 1 (C1Add), whereas the *bottom plot* shows results for a model that used HMAX, trained on features from all cameras, concatenated in a single input descriptor. Algorithms are implemented in Matlab 7.1 and run on a PC with a 3.6 GHz P4 processor and 8 Gb of RAM. WKNN, GPKNN and TGPKNN use 25, 800 and 800 nearest neighbors, respectively, obtained by cross-validation. For GPR, training time is about 400 seconds, whereas no training is needed for WKNN, GPRKNN and TGPKNN. Testing time of TGPKNN is slightly higher than WKNN, GPR and GPRKNN (as expected), but not significantly and scales favorably with the size of the dataset. As the dataset grows, KNN calculations dominate the cost of local optimization required for structured prediction

tance between the test input and all the ones in the training set (or those selected by KNN). TGPKNN is more time-consuming than both WKNN and GPR as besides finding the K nearest neighbors of the test input, it also needs to optimize the cost (14). For the current dataset, the testing time of TGPKNN remains similar to competitors even if the training set is increased, which indicates that optimizing the cost function does not dominate the computational cost of TGPKNN (the optimization problem in TGPKNN does not depend on the size of the full training set, only on the dimensionality of the output). Hence, if the number of nearest neighbors is fixed, the test time of TGPKNN will eventually approach WKNN, since KNN calculations dominated as the number of training samples increases.

We report average joint position errors for WKNN, GPR, GPRKNN and TGPKNN in Tables 4 and 5. We run two sets of comparisons: one uses models trained on the full dataset and tested on all the motions of all the subjects (Table 4). For the other tests, we use models trained on all motions of one subject and tested on all motions of the same subject (Table 5). It is remarkable that there is no significant difference between results of models trained on the full dataset and ones trained per subject. In practice, this is an essential feature of a robust system as building an accurate model for each new subject by hand is infeasible and obtaining it automatically is at the moment questionable with any of the existing methods, particularly given the variety of clothing and human body proportions in real scenes.

As expected, models that combine features from multiple views work better than those that use only information from single views. In general, all models work slightly better for HoG than HMAX in this dataset, especially for Subjects 2 and 3, where about 10 mm lower average error is obtained using HoG. TGPKNN outperforms other methods we tested and gives lower average errors. For the average joint position error over each subject (corresponding to 'Average' line in Tables 4 and 5), TGPKNN consistently gives 10–15 mm improvement relative to WKNN, GPR (GPRKNN). TGPKNN works best on multiple views with HoG features, HoG (C1+C2+C3), where it obtains 44.4 mm and 46.3 mm errors for the models trained on the full dataset and one subject, respectively. GPR and WKNN give similar performance in most cases. An exception is multiview-HoG where GPR is slightly better than WKNN.

Taking a closer look at each activity, we observe that there is quite a bit of a difference in the errors among different motions and cameras. In general, Walking and Jogging motions are reconstructed with low relative error. Analysis of the video data and viewpoints shows that these two motions are executed at lower speed than other motions. For Gestures, the error of Subject 1 and Subject 3 is much lower than for Subject 2. The video data shows that the body and the two legs of Subject 2 move a lot. Subject 3 is facing the first camera, and it appears more difficult to estimate the depth of the arms. This explains why errors in the 2nd and 3rd cameras tend to be lower than that of the 1st camera (the former two are in flank of Subject 3). Box and ThrowCatch (Subject 3) motions are recovered with relatively large error. The video data shows that these motions are faster than others and all the parts of the body are moving. Being more complex, it is also quite likely that the three human subjects perform the motions very differently. In Combo sets made of walking, jog and jumping, the reconstruction of walking and jogging is accurate whereas jumps give larger error, as they do not appear in the training set and tend to vary significantly among subjects.

Insight on how errors evolve, function of time, for different models WKNN, GPR and TGPKNN is given in

Table 4 Evaluation of average joint position error (and variance) of different models that use input descriptors based on HMAX and HoG on the test set of HumanEva-I (error reported in mm). All models are trained jointly on data from the three human subjects. Average joint position error is computed by Humaneva's online evaluation system, and we discard frames with invalid mocap case when the evaluation system returns −1 (note that averages become lower if values for invalid frames are taken to be 0). The lowest error is given in bold. In the table, '/' indicates that the values are not available (for whatever reason no test results are returned); 'Average' gives the average error for the different motions of the same subject; 'C1' indicates that image features are computed only using data from the first camera; 'C1+C2+C3' means that image features from three cameras are combined in a single descriptor; 'C1Add' means that data from the first camera is augmented with data from the second and third by transforming to the coordinate system of the first. We used GPR with KNN for the tripled training samples (standard GPR wouldn't work because there are too many training samples). WKNN, GPRKNN and TGPKNN use 25, 800 and 800 neighbors, respectively, all cross-validated

Features	Motions	Subject 1			Subject 2			Subject 3		
		WKNN	GPR	TGPKNN	WKNN	GPR	TGPKNN	WKNN	GPR	TGPKNN
HMAX	Walking	45.9(22.6)	65.3(22.3)	37.5(17.0)	48.2(25.5)	60.2(19.2)	40.2(17.8)	63.8(23.6)	74.4(27.9)	51.1(27.1)
(C1)	Jogging	55.9(21.0)	69.6(19.8)	48.7(17.7)	56.4(22.6)	66.6(19.0)	46.5(15.3)	68.3(27.2)	82.2(22.0)	58.9(21.1)
	Gestures	25.7(5.7)	30.8(6.7)	22.3(4.1)	93.3(30.9)	103.9(20.5)	84.5(16.1)	75.9(26.7)	68.5(9.9)	54.3(11.2)
	Box	73.6(14.3)	90.8(15.6)	79.6(23.4)	130.7(60.6)	133.4(54.6)	122.6(59.5)	112.1(48.1)	135.7(46.0)	116.9(64.9)
	ThrowCatch	/	/	/	74.5(35.3)	70.4(26.9)	67.5(32.1)	124.5(36.9)	78.3(21.7)	112.7(34.6)
	Combo	/	/	/	87.6(61.6)	90.1(45.6)	74.1(53.3)	131.7(88.9)	125.2(63.3)	116.1(77.4)
	Average	50.3	64.1	**47.0**	81.7	87.4	**72.6**	96.0	94.1	**85.0**
HMAX	Walking	41.5(15.5)	53.0(17.0)	31.3(7.2)	47.2(30.4)	44.8(15.5)	32.2(15.3)	46.2(21.7)	56.1(19.5)	35.3(16.3)
(C1+C2+C3)	Jogging	52.1(18.7)	49.8(12.2)	37.1(9.1)	44.3(15.2)	47.8(12.5)	34.6(7.4)	52.3(22.3)	60.6(18.6)	42.7(15.7)
	Gestures	24.3(7.4)	28.0(5.8)	21.6(5.2)	84.0(22.3)	86.1(17.0)	68.7(15.7)	49.6(6.1)	55.3(6.9)	46.5(4.7)
	Box	86.8(46.3)	74.2(22.6)	81.7(36.3)	112.1(56.6)	108.5(53.0)	89.7(42.9)	93.4(37.4)	131.6(44.1)	92.8(51.6)
	ThrowCatch	/	/	/	69.2(33.5)	70.4(26.9)	53.2(22.9)	114.1(35.9)	66.4(24.9)	90.2(33.6)
	Combo	/	/	/	82.3(53.4)	73.5(44.8)	62.3(50.0)	91.9(49.3)	110.7(53.3)	81.4(49.1)
	Average	51.2	51.3	**42.9**	73.1	71.9	**56.8**	74.6	78.5	**64.8**
HoG	Walking	47.5(21.1)	62.1(25.9)	38.2(21.4)	46.7(35.2)	51.1(30.0)	32.8(23.1)	66.7(32.0)	58.2(25.2)	40.2(23.2)
(C1)	Jogging	54.5(22.4)	64.4(21.1)	42.0(12.9)	43.3(14.4)	51.9(22.7)	34.7(16.6)	56.1(25.4)	66.6(28.0)	46.4(28.9)
	Gestures	23.4(13.5)	26.8(11.2)	20.4(7.6)	75.1(28.1)	85.5(21.8)	71.7(25.7)	75.3(11.1)	75.3(11.5)	72.7(21.2)
	Box	79.7(27.7)	78.3(21.7)	63.1(14.2)	105.8(46.9)	100.3(46.2)	98.6(64.1)	100.4(52.3)	117.7(43.7)	106.7(62.6)
	ThrowCatch	/	/	/	71.4(34.2)	78.6(22.6)	52.6(25.8)	111.7(32.9)	68.1(21.8)	107.5(38.2)
	Combo	/	/	/	78.6(48.6)	83.8(50.5)	66.5(61.2)	114.0(77.3)	122.6(80.2)	95.3(75.6)
	Average	51.3	57.9	**40.9**	70.1	75.2	**59.5**	87.4	84.8	**78.1**
HoG	Walking	37.5(12.0)	45.1(21.0)	26.6(7.0)	40.1(23.9)	33.5(15.1)	25.2(9.7)	55.3(25.1)	42.3(20.7)	31.0(13.3)
(C1+C2+C3)	Jogging	45.2(13.7)	43.1(14.4)	32.2(9.1)	37.7(12.2)	31.4(9.2)	26.9(6.0)	45.4(18.3)	42.6(14.3)	32.4(10.5)
	Gestures	23.7(7.2)	27.7(7.4)	19.2(3.5)	72.8(26.3)	67.6(17.8)	50.2(11.2)	56.1(6.4)	48.6(5.2)	50.9(4.6)
	Box	88.7(36.2)	66.7(19.1)	57.7(17.3)	91.8(41.2)	81.3(42.9)	72.5(38.0)	92.3(47.9)	90.9(38.1)	75.5(45.1)
	ThrowCatch	/	/	/	57.6(23.8)	45.9(20.7)	40.5(16.5)	92.8(31.8)	68.6(23.3)	74.1(31.8)
	Combo	/	/	/	71.9(52.2)	58.3(41.8)	51.9(50.3)	83.9(53.0)	76.2(42.8)	64.6(44.3)
	Average	48.8	45.7	**33.9**	62.0	53.0	**44.5**	71.0	61.5	**54.8**
HoG	Walking	41.2(16.8)	49.6(21.1)	32.0(17.7)	39.6(26.8)	40.7(16.7)	26.9(8.4)	55.3(21.7)	50.9(21.4)	38.4(14.6)
(C1Add)	Jogging	46.4(18.6)	52.1(12.6)	36.0(23.4)	38.0(10.0)	43.1(12.8)	31.2(6.8)	47.4(23.5)	50.7(17.7)	35.5(12.2)
	Gestures	26.4(13.5)	23.4(10.6)	20.1(7.6)	75.1(28.1)	92.2(24.0)	69.6(23.9)	75.3(11.1)	85.3(12.3)	70.4(26.1)
	Box	79.7(27.7)	81.8(24.0)	61.8(12.6)	103.4(45.1)	94.4(37.3)	92.3(51.0)	100.4(52.3)	146.8(54.3)	104.1(61.8)
	ThrowCatch	/	/	/	69.5(31.9)	62.9(24.9)	52.2(22.7)	111.7(33.4)	123.0(42.4)	107.6(42.4)
	Combo	/	/	/	69.8(49.3)	75.4(54.7)	60.3(63.2)	106.1(79.7)	113.7(87.9)	94.2(91.0)
	Average	48.4	51.7	**37.5**	65.9	68.1	**55.4**	82.7	95.1	**75.0**

Table 5 Evaluation of the different models using HoG on the test set of HumanEva-I (error reported in mm). Models for each subject are trained separately. Average error is computed by Humaneva's online evaluation system, with the lowest error in bold. '/' entries mean that values are not available (no test results returned), 'Average' gives averages for different motions of the same subject; 'C1' gives results for image feature extracted from images captured by the first camera; 'C1+C2+C3' means that image features from three cameras are combined in a single descriptor. WKNN and TGPKNN use 25 and 800 neighbors, respectively, both cross-validated

Features	Motions	Subject 1			Subject 2			Subject 3		
		WKNN	GPR	TGPKNN	WKNN	GPR	TGPKNN	WKNN	GPR	TGPKNN
HoG	Walking	49.2(21.9)	56.5(21.1)	38.1(21.4)	47.4(36.0)	53.4(27.8)	34.4(17.9)	66.8(31.6)	58.1(26.7)	43.0(23.7)
(C1)	Jogging	57.8(24.1)	64.9(25.5)	48.1(38.0)	43.6(14.8)	52.4(20.0)	34.9(7.8)	56.2(25.5)	63.4(21.6)	45.0(20.2)
	Gestures	26.4(13.5)	28.6(9.6)	20.0(6.6)	74.5(27.4)	85.0(20.7)	71.3(27.0)	75.3(11.1)	73.7(10.8)	64.6(14.3)
	Box	85.5(34.7)	75.6(21.0)	62.2(12.4)	105.6(46.7)	94.4(39.5)	92.7(51.7)	100.7(52.3)	118.2(41.0)	101.9(55.6)
	ThrowCatch	/	/	/	71.7(34.6)	56.8(24.3)	54.0(26.7)	114.2(35.4)	86.8(25.6)	106.1(34.3)
	Combo	/	/	/	79.2(48.9)	85.1(51.1)	68.4(59.1)	123.4(99.8)	118.0(73.3)	97.2(86.3)
	Average	54.7	56.4	**42.1**	70.3	71.2	**59.3**	89.4	86.4	**76.3**
HoG	Walking	53.7(31.8)	58.2(23.0)	40.7(30.0)	52.6(45.5)	53.1(31.3)	32.2(7.6)	67.5(32.1)	56.4(30.6)	45.3(35.0)
(C2)	Jogging	69.5(34.2)	67.9(32.7)	47.3(26.1)	41.6(14.8)	43.6(13.7)	32.2(7.6)	52.9(26.5)	52.6(20.2)	41.2(16.3)
	Gestures	24.6(8.1)	25.9(6.3)	21.6(4.0)	80.6(31.2)	80.1(18.7)	65.3(15.2)	55.3(7.6)	57.1(16.9)	43.8(32.6)
	Box	93.3(33.2)	87.1(22.1)	85.5(34.7)	96.4(43.9)	98.4(48.5)	82.1(34.1)	106.3(47.8)	131.6(51.7)	106.1(82.8)
	ThrowCatch	/	/	/	66.5(33.8)	59.1(26.2)	51.1(24.6)	87.0(37.2)	104.2(38.8)	89.7(39.1)
	Combo	/	/	/	82.9(56.8)	79.6(53.6)	66.4(54.2)	91.6(53.7)	96.5(54.4)	78.5(52.4)
	Average	60.3	59.8	**48.8**	70.1	69.0	**54.9**	76.8	83.1	**67.4**
HoG	Walking	66.6(49.4)	62.4(28.5)	42.0(33.1)	50.9(36.8)	54.1(26.2)	37.1(20.2)	62.0(30.9)	54.4(26.1)	40.0(20.5)
(C3)	Jogging	69.7(36.3)	65.7(26.0)	37.1(20.2)	42.5(16.5)	53.1(31.3)	34.3(10,8)	53.3(26.9)	53.9(21.0)	40.9(15.7)
	Gestures	25.0(7.8)	39.2(6.0)	23.6(4.6)	81.9(32.8)	88.2(27.0)	72.9(39.7)	57.9(9.2)	55.4(9.1)	45.9(7.6)
	Box	99.0(45.4)	83.8(18.7)	87.1(48.8)	97.1(50.9)	97.1(50.9)	87.1(48.8)	105.8(47.3)	112.1(42.1)	89.0(51.3)
	ThrowCatch	/	/	/	67.3(34.6)	62.7(25.6)	52.5(22.1)	96.1(43.7)	98.3(37.3)	94.8(40.8)
	Combo	/	/	/	86.4(59.1)	79.8(46.4)	63.9(42.8)	92.8(53.9)	93.2(52.3)	81.5(61.2)
	Average	65.1	62.8	**47.5**	72.1	72.0	**58.0**	78.0	77.9	**65.4**
HoG	Walking	38.0(12.0)	45.2(14.9)	28.3(8.2)	41.1(26.3)	38.9(19.9)	28.0(12.8)	56.5(25.6)	44.6(19.7)	31.5(14.2)
(C1+C2+C3)	Jogging	49.1(17.2)	47.2(18.2)	37.6(19.8)	37.7(12.4)	34.8(11.2)	28.6(6.6)	45.5(18.4)	43.8(14.6)	33.4(11.6)
	Gestures	23.7(7.2)	24.9(5.5)	19.1(3.5)	76.9(26.3)	61.6(15.4)	49.2(10.5)	56.1(6.4)	52.3(5.9)	48.9(4.5)
	Box	89.0(36.7)	66.9(16.9)	56.9(15.8)	91.3(41.3)	77.0(38.8)	77.2(40.9)	93.3(48.4)	91.6(35.8)	77.2(40.9)
	ThrowCatch	/	/	/	58.1(24.8)	58.1(24.7)	40.7(17.3)	93.3(31.7)	86.4(31.0)	85.4(33.2)
	Combo	/	/	/	73.0(52.5)	63.3(41.8)	54.7(49.1)	84.3(52.1)	79.7(44.0)	65.6(43.6)
	Average	50.0	46.1	**35.5**	63.0	55.6	**46.4**	71.5	66.5	**57.0**

Figs. 10–12. All models are trained on the full dataset and tested on all motions. We observe that TGPKNN not only gives lower average joint position error but also smaller maximum errors in most frames. For example, the maximum joint position error of TGPKNN is 107.8 mm on walking of Subject 2 while that of WKNN and GPR are 147.1 mm and 246.7 mm, respectively.

5 Conclusions

We have presented a generally applicable structured prediction method, Twin Gaussian Processes (TGP) to model correlations among both inputs and outputs in multivariate, continuous valued supervised learning problems. By accounting for correlations using input-output kernels and a KL-divergence criterion, prediction becomes less sensitive to data living in low-density regions or far away from a query. Our experiments show that TGP models trained jointly, using motion data from multiple human subjects achieve 5 cm reconstruction error per 3d body joint, on average, for video sequences in the HumanEva benchmark. TGP outperforms competitors based on weighted K nearest neighbors, Gaussian process regression, or predictors based on Kernel Target Alignment or the Hilbert Schmidt Independence Criterion. The results are obtained automatically:

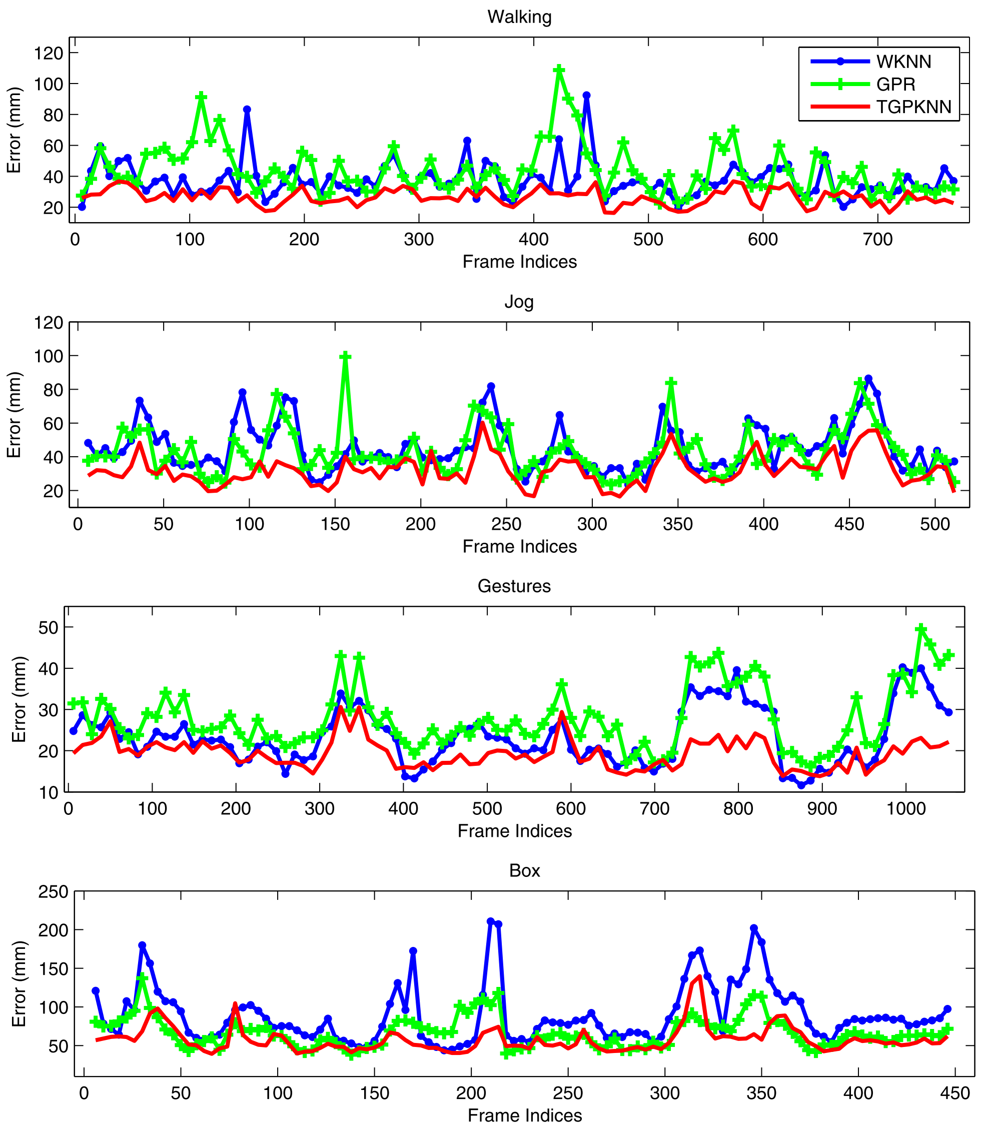

Fig. 10 Average joint position test error for three different models, and four different motions, all from Subject 2, as function of the image frame. Models are trained on the full dataset. HoG from the three cameras are combined in a single descriptor. WKNN and TGPKNN use 25 and 800 nearest neighbors, both cross-validated

no information is required about the camera calibration, the body proportions of the human subjects or their identity for training and testing, nor do we need to manually initialize the pose of the subject in the first frame of each sequence.

Future Work. We plan to study alternative kernels and image features, cost functions and hyperparameter optimization criteria. Searching for multiple optima rather than a single one could also improve the quality of solutions. We used KNN preprocessing in order to speed up some of the calculations, but optimizing the cost directly using sparse formulations is also feasible. We also plan to apply the method to other structured prediction problems and extend it to the case of discrete multivariate outputs.

Acknowledgements This work was supported, in part, by the EC and the NSF, under awards MCEXT-025481 and IIS-0535140. We thank Atul Kanaujia for discussions and generous support with feature extraction, and Roland Poppe for helpful feedback and for gracefully making available materials from his work (Poppe 2007) for comparison.

Appendix: Derivation of the ITGP Cost

Let $\mathbf{B}_{ij}$ be the ij-th block of $\mathbf{B}$ and $\mathbf{C} = \mathbf{B}_{22} - \mathbf{B}_{21}\mathbf{B}_{11}^{-1}\mathbf{B}_{12}$. The determinant identity of the block matrix can be expressed as:

$$\left|\begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}\right| = |\mathbf{B}_{11}|\,|\mathbf{C}| \tag{35}$$

The inverse identity of block matrix can be expressed as:

$$\begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}^{-1}$$

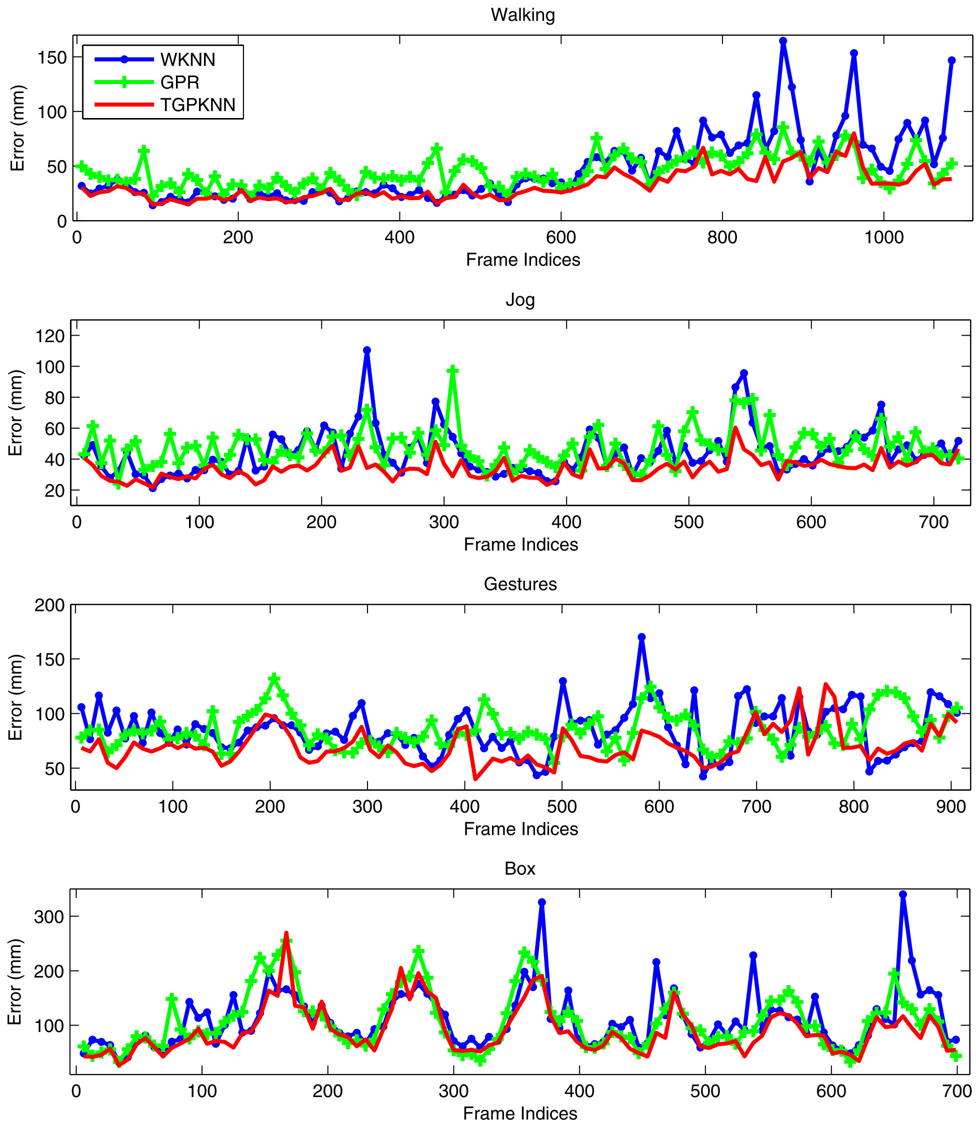

Fig. 11 Average joint position test error for three different models, and four different motions, all from Subject 1, function of timestep. Models are trained on the full dataset. HoG features from the three cameras are combined in a single descriptor

$$= \begin{bmatrix} \mathbf{B}_{11}^{-1} + \mathbf{B}_{11}^{-1}\mathbf{B}_{12}\mathbf{C}^{-1}\mathbf{B}_{21}\mathbf{B}_{11}^{-1} & -\mathbf{B}_{11}^{-1}\mathbf{B}_{12}\mathbf{C}^{-1} \\ -\mathbf{C}^{-1}\mathbf{B}_{21}\mathbf{B}_{11}^{-1} & \mathbf{C}^{-1} \end{bmatrix} \tag{36}$$

Applying (35) to the second and fourth terms of (6), we have

$$\log \left| \begin{bmatrix} \mathbf{K}_X & \mathbf{K}_X^x \\ (\mathbf{K}_X^x)^\top & K_X(\mathbf{x},\mathbf{x}) \end{bmatrix} \right| = \log|\mathbf{K}_X| - \log\left[K_X(\mathbf{x},\mathbf{x}) + (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1}\mathbf{K}_X^x\right] \tag{37}$$

$$\log \left| \begin{bmatrix} \mathbf{K}_R & \mathbf{K}_R^r \\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r},\mathbf{r}) \end{bmatrix} \right| = \log|\mathbf{K}_R| + \log\left[K_R(\mathbf{r},\mathbf{r}) - (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1}\mathbf{K}_R^r\right] \tag{38}$$

Applying (36) to the third term of (6), we have

$$\begin{aligned} &\mathrm{Tr}\left(\begin{bmatrix} \mathbf{K}_X & \mathbf{K}_X^x \\ (\mathbf{K}_X^x)^\top & K_X(\mathbf{x},\mathbf{x}) \end{bmatrix}\begin{bmatrix} \mathbf{K}_R & \mathbf{K}_R^r \\ (\mathbf{K}_R^r)^\top & K_R(\mathbf{r},\mathbf{r}) \end{bmatrix}^{-1}\right) \\ &\quad = \mathrm{Tr}\left(\mathbf{K}_X\mathbf{K}_R^{-1}\right) \\ &\qquad + \left[K_X(\mathbf{x},\mathbf{x}) - 2(\mathbf{K}_X^x)^\top \mathbf{K}_R^{-1}\mathbf{K}_R^r\right] \\ &\qquad \times \left[K_R(\mathbf{r},\mathbf{r}) - (\mathbf{K}_R^r)^\top\mathbf{K}_R^{-1}\mathbf{K}_R^r\right]^{-1} \\ &\qquad + \left[(\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1}\mathbf{K}_X\mathbf{K}_R^{-1}\mathbf{K}_R^r\right] \\ &\qquad \times \left[K_R(\mathbf{r},\mathbf{r}) - (\mathbf{K}_R^r)^\top\mathbf{K}_R^{-1}\mathbf{K}_R^r\right]^{-1} \end{aligned} \tag{39}$$

Adding up (37), (38) and (39), dropping constants not including $\mathbf{x}$ and $\mathbf{r}$, then multiplying by $K_R(\mathbf{r},\mathbf{r}) -$

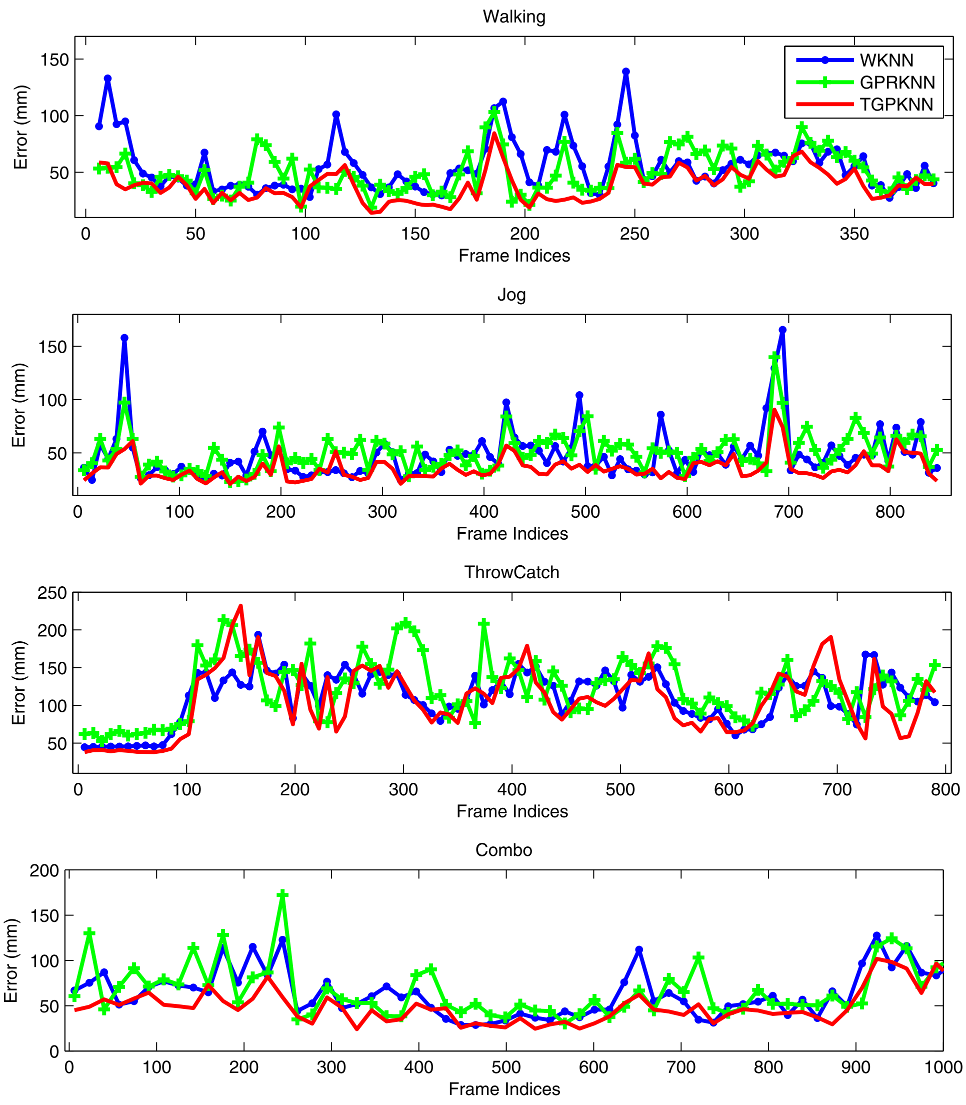

Fig. 12 Average joint position test error for three different models, and four different motions, all from Subject 3, as function of timestep. Models are trained on the full dataset. HoG from the three cameras are combined in a single descriptor. WKNN and TGPKNN use 25 and 800 nearest neighbors, both cross-validated

$(\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1} \mathbf{K}_R^r$, we have

$$\begin{aligned} &A(\mathbf{x}, \mathbf{r}) \\ &\quad = K_X(\mathbf{x}, \mathbf{x}) - 2(\mathbf{K}_X^x)^\top \mathbf{K}_R^{-1} \mathbf{K}_R^r \\ &\qquad + (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1} \mathbf{K}_X \mathbf{K}_R^{-1} \mathbf{K}_R^r \\ &\qquad + \left[K_R(\mathbf{r}, \mathbf{r}) - (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1} \mathbf{K}_R^r \right] \\ &\qquad \times \log \left[K_R(\mathbf{r}, \mathbf{r}) - (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1} \mathbf{K}_R^r \right] \\ &\qquad - \left[K_R(\mathbf{r}, \mathbf{r}) - (\mathbf{K}_R^r)^\top \mathbf{K}_R^{-1} \mathbf{K}_R^r \right] \\ &\qquad \times \log \left[K_X(\mathbf{x}, \mathbf{x}) - (\mathbf{K}_X^x)^\top \mathbf{K}_X^{-1} \mathbf{K}_X^x \right] \end{aligned} \tag{40}$$

Removing terms that are independent of **x** (3rd and 4th), we obtain (14). Removing terms that do not include **r** (1st term), we get (19). Replacing corresponding terms with the ones in the dynamic TGP, we obtain (22).

References

Agarwal, A., & Triggs, B. (2006). Recovering 3d human pose from monocular images. *IEEE transactions on pattern analysis and machine intelligence*.

Bar-Shalom, Y., & Fortman, T. (1988). *Tracking and data association*. San Diego: Academic Press.

Battu, B., Krappers, A., & Koenderink, J. (2007). Ambiguity in pictorial depth. *Perception 36*.

Bishop, C., & Svensen, M. (2003). Bayesian mixtures of experts. In *Uncertainty in artificial intelligence*, 2003.

Blake, A., North, B., & Isard, M. (1999). Learning multi-class dynamics. *Advances in Neural Information Processing Systems*, *11*, 389–395.

Bo, L., & Sminchisescu, C. (2008). Twin Gaussian processes for structured prediction. *Snowbird Learning*, April.

Bo, L., Sminchisescu, C., Kanaujia, A., & Metaxas, D. (2008). Fast algorithms for large scale conditional 3D prediction. In *IEEE conference on computer vision and pattern recognition*, 2008.

Brubaker, M., & Fleet, D. (2008). The kneed walker for human pose tracking. In *IEEE international conference on computer vision and pattern recognition*, 2008.

Choo, K., & Fleet, D. (2001). People tracking using hybrid Monte Carlo filtering. In *IEEE international conference on computer vision*, 2001.

CMU (2003). Human Motion DataBase. Online at http://mocap.cs.cmu.edu/search.html.

Cortes, C., Mohri, M., & Weston, J. (2005). A general regression technique for learning transductions. In *International conference on machine learning* (pp. 153–160) 2005.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. S. (2001a). On kernel-target alignment. In *Advances in neural information processing systems* (pp. 367–373) 2001.

Cristianini, N., Shawe-Taylor, J., & Kandola, J. S. (2001b). Spectral kernel methods for clustering. In *Advances in neural information processing systems* (pp. 649–655) 2001.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE international conference on computer vision and pattern recognition*, 2005.

Deutscher, J., Blake, A., & Reid, I. (2000). Articulated body motion capture by annealed particle filtering. In *IEEE international conference on computer vision and pattern recognition*, 2000.

Deutscher, J., Davidson, A., & Reid, I. (2001). Articulated partitioning of high dimensional search spaces associated with articulated body motion capture. In *IEEE international conference on computer vision and pattern recognition*, 2001.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, *195*(2), 216–222.

Elgammal, A., & Lee, C. (2004). Inferring 3d body pose from silhouettes using activity manifold learning. In *IEEE international conference on computer vision and pattern recognition*, 2004.

Geurts, P., Wehenkel, L., & d'Alché Buc, F. (2006). Kernelizing the output of tree-based methods. In *International conference on machine learning* (pp. 345–352) 2006.

Geurts, P., Wehenkel, L., & d'Alché Buc, F. (2007). Gradient boosting for kernelized output spaces. In *International conference on machine learning*, New York, NY, USA (pp. 289–296) 2007.

Gretton, A., Bousquet, O., Smola, A. J., & Schölkopf, B. (2005a). Measuring statistical dependence with Hilbert-Schmidt norms. In S. Jain & W.-S. Lee (Eds.), *Proceedings algorithmic learning theory*, 2005.

Gretton, A., Herbrich, R., Smola, A. J., Bousquet, O., & Schölkopf, B. (2005b). Kernel methods for measuring independence. *Journal of Machine Learning Research*, *6*, 2075–2129.

Guzman, A. N., & Holden, S. (2007). Twinned Gaussian processes. In *Advances in neural information processing systems*, December 2007.

Hinton, G. E., & Roweis, S. T. (2002). Stochastic neighbor embedding. In *Advances in neural information processing systems* (pp. 833–840) 2002.

Isard, M., & Blake, A. (1998). CONDENSATION—conditional density propagation for visual tracking. *International Journal of Computer Vision*.

Kanaujia, A., Sminchisescu, C., & Metaxas, D. (2006). Semi-supervised hierarchical models for 3D human pose reconstruction. In *IEEE international conference on computer vision and pattern recognition*, 2006.

Kanaujia, A., Sminchisescu, C., & Metaxas, D. (2007). Spectral latent variable models for perceptual inference. In *IEEE International Conference on Computer Vision*, Vol. 1, 2007.

Kehl, R., Bray, M., & Gool, L. V. (2005). Full body tracking from multiple views using stochastic sampling. In *IEEE international conference on computer vision and pattern recognition*, 2005.

Koenderink, J. (1998). Pictorial relief. *Philosophical Transactions Royal Society London A 356*.

Koenderink, J., & van Doorn, A. (1979). The internal representation of solid shape with respect to vision. *Biological Cybernetics 32*(3).

Krauthgamer, R., & Lee, J. R. (2004). Navigating nets: simple algorithms for proximity search. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 798–807) 2004.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *International conference on machine learning*, 2001.

Lawrence, N. (2005). Probabilistic non-linear component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, *6*, 1783–1816.

Lee, H. J., & Chen, Z. (1985). Determination of 3D human body postures from a single view. *Computer Vision, Graphics and Image Processing*, *30*, 148–168.

Li, R., Yang, M., Sclaroff, S., & Tian, T. (2006). Monocular tracking of 3D human motion with a coordinated mixture of factor analyzers. In *European conference on computer vision*, 2006.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2).

Memisevic, R. (2006). Kernel information embeddings. In *International conference on machine learning*, 2006.

Morris, D., & Rehg, J. (1998). Singularity analysis for articulated object tracking. In *IEEE international conference on computer vision and pattern recognition* (pp. 289–296) 1998.

Neal, R. (1998). *Annealed importance sampling* (Technical Report 9805). Department of Statistics, University of Toronto.

Poppe, R. (2007). Evaluating example-based human pose estimation: Experiments on HumanEva sets. In *HumanEva Workshop CVPR*, 2007.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning. Adaptive computation and machine learning*. Cambridge: MIT Press.

Rosales, R., & Sclaroff, S. (2002). Learning body pose via specialized maps. In *Advances in neural information processing systems*, 2002.

Roth, S., Sigal, L., & Black, M. (2004). Gibbs likelihoods for Bayesian tracking. In *IEEE international conference on computer vision and pattern recognition*, 2004.

Schaal, S., Atkeson, C., & Moore, A. (1997). Locally weighted learning. *Artificial Intelligence Review*, *11*, 11–73.

Serre, T., Wolf, L., Bileschi, S., & Riesenhuber, M. (2007). Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(3), 411–426.

Shakhnarovich, G., Viola, P., & Darrell, T. (2003). Fast pose estimation with parameter sensitive hashing. In *IEEE international conference on computer vision*, 2003.

Sidenbladh, H., & Black, M. (2001). Learning image statistics for Bayesian tracking. In *IEEE international conference on computer vision*, 2001.

Sidenbladh, H., Black, M., & Fleet, D. (2000). Stochastic tracking of 3D human figures using 2D image motion. In *European conference on computer vision*, 2000.

Sidenbladh, H., Black, M., & Sigal, L. (2002). Implicit probabilistic models of human motion for synthesis and tracking. In *European conference on computer vision*, 2002.

Sigal, L., & Black, M. (2006). *HumanEva: synchronized video and motion capture dataset for evaluation of articulated human motion* (Technical Report CS-06-08). Brown University.

Sigal, L., Balan, A., & Black, M. J. (2007). Combined discriminative and generative articulated pose and non-rigid shape estimation. In *Advances in neural information processing systems*, 2007.

Sminchisescu, C. (2002). Consistency and coupling in human model likelihoods. In *IEEE international conference on automatic face and gesture recognition* (pp. 27–32). Washington, DC, 2002.

Sminchisescu, C., & Jepson, A. (2004a). Generative modeling for continuous non-linearly embedded visual inference. In *International conference on machine learning* (pp. 759–766). Banff, 2004.

Sminchisescu, C., & Jepson, A. (2004b). Variational mixture smoothing for non-linear dynamical systems. In *IEEE international conference on computer vision and pattern recognition* (Vol. 2, pp. 608–615). Washington, DC, 2004.

Sminchisescu, C., & Telea, A. (2002). Human pose estimation from silhouettes. A consistent approach using distance level sets. In *WSCG international conference for computer graphics, visualization and computer vision*, Czech Republic, 2002.

Sminchisescu, C., & Triggs, B. (2001). Covariance-scaled sampling for monocular 3D body tracking. In *IEEE international conference on computer vision and pattern recognition* (Vol. 1, pp. 447–454). Hawaii, 2001.

Sminchisescu, C., & Triggs, B. (2002a). Building roadmaps of local minima of visual models. In *European conference on computer vision* (Vol. 1, pp. 566–582). Copenhagen, 2002.

Sminchisescu, C., & Triggs, B. (2002b). Hyperdynamics importance sampling. In *European conference on computer vision* (Vol. 1, pp. 769–783). Copenhagen, 2002.

Sminchisescu, C., & Triggs, B. (2003). Kinematic jump processes for monocular 3D human tracking. In *IEEE international conference on computer vision and pattern recognition* (Vol. 1, pp. 69–76). Madison, 2003.

Sminchisescu, C., & Triggs, B. (2005). Mapping minima and transitions in visual models. *International Journal of Computer Vision 61*(1).

Sminchisescu, C., & Welling, M. (2007). Generalized darting Monte-Carlo. In *Artificial Intelligence and Statistics*, Vol. 1, 2007.

Sminchisescu, C., Kanaujia, A., Li, Z., & Metaxas, D. (2005). Conditional visual tracking in kernel space. In *Advances in neural information processing systems*, 2005.

Sminchisescu, C., Kanaujia, A., & Metaxas, D. (2006). Learning joint top-down and bottom-up processes for 3D visual inference. In *IEEE international conference on computer vision and pattern recognition*, 2006.

Sminchisescu, C., Kanaujia, A., & Metaxas, D. (2007). BM^3E: discriminative density propagation for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Smola, A. J., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *International conference on machine learning* (pp. 911–918) 2000.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin Markov networks. In *Advances in neural information processing systems*, 2004.

Tresp, V. (2000). Mixtures of Gaussian processes. In *Advances in neural information processing systems*, 2000.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *International conference on machine learning*, 2004.

Urtasun, R., Fleet, D., Hertzmann, A., & Fua, P. (2005). Priors for people tracking in small training sets. In *IEEE international conference on computer vision*, 2005.

Vincent, P., & Bengio, Y. (2002). Kernel matching pursuit. *Machine Learning*, *48*, 165–187.

Vondrak, M., Sigal, L., & Jenkins, O. C. (2008). Physical simulation for probabilistic motion tracking. In *IEEE international conference on computer vision and pattern recognition*, 2008.

Wang, J., Fleet, D. J., & Hertzmann, A. (2008). Gaussian process dynamical models. In *IEEE transactions on pattern analysis and machine intelligence*, 2008.

Weston, J., Chapelle, O., Elisseeff, A., Scholkopf, B., & Vapnik, V. (2002). Kernel dependency estimation. In *Advances in neural information processing systems*, 2002.