

# Northumbria Research Link

Citation: Liu, Zhiguang, Zhou, Liuyang, Leung, Howard and Shum, Hubert P. H. (2016) Kinect posture reconstruction based on a local mixture of Gaussian process models. IEEE Transactions on Visualization and Computer Graphics, 22 (11). pp. 2437-2450. ISSN 1077-2626

Published by: IEEE

URL: <http://doi.org/10.1109/TVCG.2015.2510000>  
<<http://doi.org/10.1109/TVCG.2015.2510000>>

This version was downloaded from Northumbria Research Link:  
<http://nrl.northumbria.ac.uk/id/eprint/25559/>

Northumbria University has developed Northumbria Research Link (NRL) to enable users to access the University's research output. Copyright © and moral rights for items on NRL are retained by the individual author(s) and/or other copyright owners. Single copies of full items can be reproduced, displayed or performed, and given to third parties in any format or medium for personal research or study, educational, or not-for-profit purposes without prior permission or charge, provided the authors, title and full bibliographic details are given, as well as a hyperlink and/or URL to the original metadata page. The content must not be changed in any way. Full items must not be sold commercially in any format or medium without formal permission of the copyright holder. The full policy is available online: <http://nrl.northumbria.ac.uk/policies.html>

This document may differ from the final, published version of the research and has been made available online in accordance with publisher policies. To read and/or cite from the published version of the research, please visit the publisher's website (a subscription may be required.)

# Kinect Posture Reconstruction Based on a Local Mixture of Gaussian Process Models

Zhiguang Liu, Liuyang Zhou, Howard Leung, and Hubert P.H. Shum

**Abstract**—Depth sensor based 3D human motion estimation hardware such as Kinect has made interactive applications more popular recently. However, it is still challenging to accurately recognize postures from a single depth camera due to the inherently noisy data derived from depth images and self-occluding action performed by the user. In this paper, we propose a new real-time probabilistic framework to enhance the accuracy of live captured postures that belong to one of the action classes in the database. We adopt the Gaussian Process model as a prior to leverage the position data obtained from Kinect and marker-based motion capture system. We also incorporate a temporal consistency term into the optimization framework to constrain the velocity variations between successive frames. To ensure that the reconstructed posture resembles the accurate parts of the observed posture, we embed a set of joint reliability measurements into the optimization framework. A major drawback of Gaussian Process is its cubic learning complexity when dealing with a large database due to the inverse of a covariance matrix. To solve the problem, we propose a new method based on a local mixture of Gaussian Processes, in which Gaussian Processes are defined in local regions of the state space. Due to the significantly decreased sample size in each local Gaussian Process, the learning time is greatly reduced. At the same time, the prediction speed is enhanced as the weighted mean prediction for a given sample is determined by the nearby local models only. Our system also allows incrementally updating a specific local Gaussian Process in real time, which enhances the likelihood of adapting to run-time postures that are different from those in the database. Experimental results demonstrate that our system can generate high quality postures even under severe self-occlusion situations, which is beneficial for real-time applications such as motion-based gaming and sport training.

**Index Terms**—Gaussian process, incremental learning, kinect, posture reconstruction

## 1 INTRODUCTION

HUMAN motion recognition is an important component in interactive applications nowadays. Traditional motion-based systems such as those for dance training are based on motion capture technology, where the user's movement is captured by an optical motion capture system [1]. While these applications can evaluate user performance with the accurately captured motions, they are not convenient since users have to wear capture suits with reflective markers. Moreover, these devices are relatively expensive and are not affordable for home use.

Depth image based motion sensing devices such as the Microsoft Kinect [2] serve as an alternative to capture human movement for interactive applications. Kinect is a controller-free device that infers 3D positions of human body joints from a single depth image with the help of a data-driven machine learning algorithm [3]. With such a device, it becomes possible to implement a natural user interface for virtual reality applications and gesture based

systems [4]. While Kinect can robustly track the 3D postures of the user, the captured data suffer from poor precision due to self-occlusions and insufficient information provided by the Kinect sensor. Therefore, Kinect based interactive applications usually require the user to face the device so that individual body parts are observable, which greatly limits the system flexibility. In addition, the user has to minimize self-occluded postures, or Kinect would misrecognize body parts. As illustrated in Fig. 1, the blue skeleton represents the tracked result by Kinect SDK [2]. We can see the tracked arms are twisted due to self-occlusions. Therefore, it is essential to develop effective posture reconstruction strategies for interactive applications.

The occlusion problem and incompleteness of the tracked joints remain challenging despite the posture reconstruction research proposed in the past years. Generating postures from low dimensional signals is a potential solution for posture reconstruction [5], [6]. However, these methods assume the low dimensional signal to be stable and accurate, while joints tracked by Kinect are not. Hence, applying them to reconstruct Kinect postures will create unsatisfying results. Shum et al. [7] applies reliability measurement to improve the posture reconstruction process. However, the reconstruction results depend heavily on the similarity between database postures and input ones. The system therefore requires a huge posture database.

In this paper, we propose a probabilistic model based on Gaussian Process (GP) to reconstruct postures captured from Kinect, where the input motion belongs to one of the action classes in the database. Unlike previous systems that require a large motion database, GP based model can be

- Z. Liu and H. Leung are with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR. E-mail: zhiguali2-c@my.cityu.edu.hk, howard@cityu.edu.hk.
- L. Zhou is with the Wisers Research, Wisers Information Limited, Hong Kong SAR. E-mail: leozhou@wisers.com.
- H.P.H. Shum is with the Faculty of Engineering and Environment, Northumbria University, United Kingdom. E-mail: hubert.shum@northumbria.ac.uk.

Manuscript received 15 Feb. 2015; revised 4 Dec. 2015; accepted 8 Dec. 2015. Date of publication 16 Dec. 2015; date of current version 19 Sept. 2016.

Recommended for acceptance by A. Shamir.

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.

Digital Object Identifier no. 10.1109/TVCG.2015.2510000

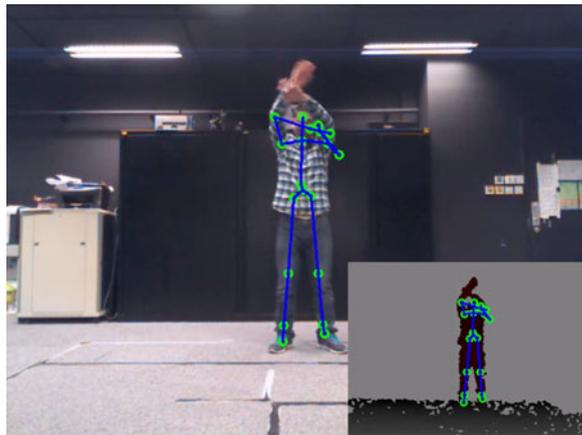


Fig. 1. Example of an inaccurately tracked posture from Kinect. The blue skeleton is the tracked result by Kinect.

robustly trained from small training sets. Moreover, the parameters of the kernel function can be optimized without relying on experimental cross validation [8]. We constrain the solution space such that the reconstructed posture is accurate while maintaining the originality of the input posture from Kinect. We adopt Gaussian Process model as a spatial prior distribution to predict the offset between Kinect and the ground truth, which aims at improving the accuracy of the postures in the case where there is sensor error from Kinect. Furthermore, since reconstructing each posture independently cannot ensure the temporal smoothness of the posture sequence, we introduce a temporal consistency term to constrain the velocity variations between successive frames. Inspired by [7], we embed the reliability of each joint into the optimization framework to ensure that the reconstructed posture resembles the accurate parts of the Kinect tracked posture. Last, we propose a new method based on local mixture of Gaussian Processes to alleviate the cubic learning complexity of a regular GP model such that our system can deal with a large variety of movement. The experimental results demonstrate that the proposed approach is effective in reconstructing a number of motions containing self-occlusions. For example, as illustrated in Fig. 5a, our method accurately reconstructs the posture of bending over with a number of joints occluded.

The major contributions of this paper are summarized as follows:

- 1) We propose a new unified framework for posture reconstruction using Kinect. The system optimizes an occluded posture live captured by Kinect, which maintains the correctness of the posture while preserving temporal smoothness between frames. The proposed system performs well with significant smaller training sets comparing with previous work in the field.
- 2) We propose three terms to constrain the solution space. The Gaussian Process based spatial prediction term utilizes motion capture data to reconstruct input postures. The temporal prediction term ensures the temporal consistency between consecutive frames. Finally, the reliability term guides the optimized postures toward the more reliable parts of the Kinect postures, preserving the property of the input postures.

- 3) We propose a new method based on local mixture of Gaussian Processes that partitions training samples into local regions to relieve the cubic learning complexity problem of Gaussian Process. With the proposed framework, the prediction speed is enhanced as only a few local models are considered for each input posture. It also enables incremental updating of local models in real time, which enhances the likelihood to adapt to the postures that are different from those in the database.

Compared with our previous work [9], we have significantly improved the spatial prediction algorithm. First, with the newly proposed method based on local mixture of GP models, our method generates postures of similar quality to that of [9] with significantly less training data. Second, we design a new algorithm to incrementally update a specific local Gaussian Process in real time, which enables the system to adapt to run-time postures that are different from those in the database. Last, because of the use of local models, our new framework only needs to consider a few local models that are close to an input posture rather than the whole database. Such an enhancement in efficiency allows us to combine all types of motion as a single database, while in [9] a separate database is built for each type of motion.

The rest of the paper is organized as below. We first review the related work of posture reconstruction in Section 2. Section 3 explains the procedure of data acquisition and pre-processing processes. In Section 4, we elaborate the spatial prediction, temporal prediction, and reliability term of the objective function, and the new method based on local mixture of GP models for posture reconstruction. Experimental analysis and evaluation are conducted in Section 5. Finally, in Section 6, we conclude the paper, as well as discuss the limitations and future research directions.

## 2 RELATED WORK

With the advancement in real-time depth cameras such as Kinect, human motion recognition and posture estimation have become a popular research topic in recent years. Kinect is based on motion recognition technology proposed by [3], where they use per-pixel classification method to quickly predict 3D joint positions from a single depth image. A number of research domains have benefited from Kinect, such as human-machine interaction [10], natural user interfaces [4], and 3D reconstruction [11]. A recent review on human activity analysis with Kinect can be found in [12]. Bailey and Bodenheimer [13] investigated the perceived differences in the quality of animation generated using motion capture data and a Kinect sensor, which clearly showed that the data recorded from Kinect was of lower quality compared with motion capture data from a Vicon motion capture system. Hence, it is essential to develop an effective posture reconstruction method to enhance the posture quality of Kinect.

In this section, we first review previous work on reconstructing postures from low dimensional signals. We then discuss data-driven approaches for posture reconstruction. We finally review the regression methods applied to posture reconstruction.

## 2.1 Posture Reconstruction from Low Dimensional Signals

Full body postures can be represented by a set of low dimensional signals [5]. Some research work has been proposed to reconstruct a full posture with a subset of the signals. Kim et al. [14] reconstructed human motion from 3D motion sensors on a performer using kernel CCA-based regression. Given the input data from sparse motion sensors, they retrieve similar postures from the motion capture database and transform the low dimensional signal into the full posture space using an online local model. Chai and Hodgins [5] employed a small set of retro-reflective markers to capture performance animation in real time. In their system, the low dimensional control signals from the user's performance were supplemented by a pre-recorded human motion database. At run time, the system automatically learned some local models from the retrieved motion capture data that were close to the marker locations recorded by the camera. Their system only needs video cameras and a small set of markers, which makes it low cost and practical for home use. However, the majority of markers have to be tracked by the cameras to provide enough information for posture reconstruction.

Liu et al. [6] used a small number of motion sensors to control a full-body human character. They constructed online local dynamic models from pre-recorded motion capture database and used them to construct full-body human motion in a Maximum-a-Posteriori framework, in which the system tried to find the most similar postures from database for reconstruction. Helten et al. [15] adaptively fused inertial and depth information in a hybrid framework for posture estimation. Although these methods can be used to reconstruct postures from low dimensional signals, there is an assumption that these low dimensional signals are reliable and stable. It is therefore not applicable to noisy Kinect data.

## 2.2 Data-Driven Posture Reconstruction

Data-driven approaches usually reconstruct postures by evaluating the similarity between the input posture and a large posture database. Sigalas et al. [16] presented a data-driven model based method for 3D torso posture estimation from RGB-D image sequence. Although their method can extract the upper body posture of users without an initialization phase, they did not cope with full body posture recovery nor handle the occlusion problem. Shum et al. [7], [17] proposed a unified framework to control physically simulated characters with live captured motion from Kinect by searching for similar postures in a marker-based motion database. They constructed a latent space with a small number of retrieved similar postures, and applied optimization in the space to reconstruct the input postures. Baak et al. [18] introduced a data-driven approach for full body reconstruction from a depth camera. They proposed an efficient algorithm for extracting posture features from the depth data. However, for fast movements, the proposed system required all five extremities to be visible. Shen et al. [19] introduced an exemplar-based method to correct the postures from Kinect using marker-based motion data.

Yasin et al. [20] introduced a model based framework for full body reconstruction from 2D video data using motion

capture database as the prior knowledge. The postures were reconstructed in an optimization framework, in which similar motion capture postures were retrieved through nearest neighbor searching. However, the accuracy is not robust because the 2D features projected from 3D motion induce posture ambiguity. Wei et al. [21] solved the reconstruction problem by registering a 3D articulated model with depth information. They formulated the registration problem into a Maximum-a-Posteriori framework to register a 3D articulated human body model with monocular depth via linear system solvers. To tackle the problem of manual initialization and failure recovery, they combined 3D pose tracking with 3D pose detection.

In general, these data-driven methods requires large database as prior, and the reconstruction results depend heavily on the retrieved postures.

## 2.3 Regression Based Posture Reconstruction

Structured regression models for posture estimation such as [22] and [23] can model the correlations between multivariate output and input. Bo and Sminchisescu [22] presented the Twin GP model that employs GP priors to model input and output relations. The output postures were estimated by minimizing the Kullback-Leibler divergence. Bo and Sminchisescu [23] optimized an output-associative functional that incorporates outputs and inputs using primal/dual formulations and adapts the model to kernel ridge regression and support vector regression. Shakhnarovich et al. [24] estimated upper body posture, interpolating k-nearest-neighbor postures matched by parameter sensitive hashing. Ramakrishna et al. [25] presented an inference machine to estimate articulated human pose. Their method allows learning a rich spatial model and incorporating high-capacity supervised predictors, which results in substantially improved pose estimation performance. Recently, it has been shown that deep learning methods such as [26] and [27] generate high precision pose estimates compared to state-of-art methods.

Gaussian Process models are flexible probabilistic non-parametric models. [9] presented a new probabilistic framework based on Gaussian Process to enhance the accuracy of the postures live captured by Kinect. Their method can generate high quality postures even under severe self-occlusion situations. GP models are usually applied to small data sets of a few hundred samples due to its  $O(N^3)$  training complexity, where  $N$  is the size of training data. In contrast, our incremental sparsification method can efficiently handle large data sets.

Previous attempts to solve the cubic learning complexity problem of GP involve sparse Gaussian Process (SGP) [28], [29], [30] and mixture of experts (ME) [31], [32], [33], [34]. SGP approximates the covariance matrix with a small subset of training data [29] or a set of inducing variables [30]. While SGP can greatly reduce the computational complexity, it utilizes a global voting scheme in which all training samples contribute to the prediction of a new test input. In contrast, ME applies gating network to partition the input space into different subspaces, where each GP expert is trained independently [31]. Compared with a regular GP model, the computational complexity of ME is reduced due to the significantly decreased sample size in each

subspace [35], [36]. However, for simple expert, the gating network has to be more complicated to model the function, which results in a higher risk of getting stuck in local minima or a slower learning process [37].

Urtasun and Darrell [33] proposed a sparse regression scheme for efficient inference of high dimensional and multi-modal mappings. Their method was based on a local mixture of Gaussian Processes defined on both appearance and posture. Different from [33] in which local GP models were defined separately in the input and output spaces, our local GP models are defined in the common posture space. Nguyen-Tuong et al. [36] proposed a method with local GP models to speed up standard Gaussian process regression. They grouped the training data into local regions by the distance measurement. The prediction of a query point was determined by the weighted prediction of nearby local models. Our local GP models are similar to that of [36] in the sense that we use the Gaussian kernel to measure the similarity between a test joint and the centers of local models. However, our framework also embeds the temporal constraint into optimization, which ensures the smoothness across consecutive frames. In addition, compared with both work, we implement the incremental updating of local models in real time, which enhances the likelihood to adapt to run-time postures that are different from those in the database. Sigal et al. [38] proposed the shared Kernel Information Embedding that can learn mappings from image features to 3D postures. In spite of efficient solutions, this method typically requires large training sets to represent the variability of appearance of different people and viewpoints.

In this paper, we use Gaussian Process to model the prior distribution of postures from Kinect with marker-based motion data. Our method draws inspiration from [35], [36], which apply local GP models to speed up the training and prediction. In addition to training local GP models, ME needs to learn a gating network to select local models. We use a kernel function to measure the similarity rather than training a gating network. The parameters of the kernel can be calculated during the learning of local GP models. Our approach is effective even with small training data as GP based model can robustly learn from small training sets.

### 3 DATA ACQUISITION AND PREPROCESSING

For brevity, in this paper we will use *MOCAP* to represent human motion data captured by an optical motion capture system. The postures obtained from Kinect are noisy and incomplete while *MOCAP* is accurate and stable. Hence, we can use *MOCAP* captured in an offline training stage to reconstruct postures captured by Kinect in real time.

#### 3.1 Data Acquisition

We build a motion database captured from an optical motion capture system of Motion Analysis Corporation [39] with seven cameras. Our database consists of different types of motions such as golf swinging and Tai Chi. The skeleton of the *MOCAP* system is a superset of that of the Kinect system, so we manually select 20 joints from the skeleton of the *MOCAP* system to match those of Kinect. Each posture in the database denotes a set of 3D positions of the body parts.

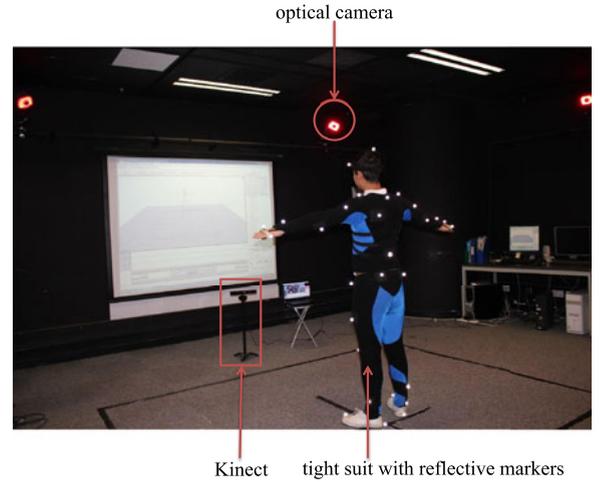


Fig. 2. Human motion capture with Kinect and an optical motion capture system.

In this paper, we model the relationship between Kinect data and *MOCAP* with Gaussian Process. Specifically, we capture motions with Kinect and optical motion capture system at the same time to identify their correspondence. The setup of this capturing procedure is shown in Fig. 2. The posture of Kinect at time  $t$  is denoted as  $X_t = (x_t^1, x_t^2, \dots, x_t^n)$ ,  $x_t^i \in \mathbb{R}^3$ , where  $x_t^i$  represents the 3D joint position of joint  $i$  over time  $t$ . There are 20 joints based on the skeleton definition of Kinect, i.e.,  $n = 20$ . The corresponding *MOCAP* of  $X_t$  is denoted as  $M_t = (m_t^1, m_t^2, \dots, m_t^n)$ ,  $m_t^i \in \mathbb{R}^3$ .

To enhance the robustness of the spatial prediction model (Section 4.1) and to make the system invariant to different subjects, we follow [7] to conduct the normalization and retargeting processes, as they are simple yet effective. The posture normalization procedure is done by removing the rotation along the vertical axis and the global 3-D translation. The retargeting procedure ensures the system to be invariant to the skeleton size of the user.

#### 3.2 Posture Budgeting

In this section, we introduce a data pruning scheme called posture budgeting to discard redundant samples.

We employ the probabilistic GPs to determine the samples that are informative to the model. We remove a specific training sample if such a sample can be precisely predicated by its neighbors in terms of the mean and variance. Specifically, we iteratively check each training sample  $(x_t^i, y_t^i)$  of joint  $i$  and determine its redundancy by calculating the relative entropy of the prediction of the training sample  $(x_t^i, y_t^i)$  with respect to the rest of the database. Following [33], we compute the Kullback-Leibler (KL) divergence by:

$$D_{KL}(p(y_t^i|X^i, Y^i, x_t^i) || p(y_t^i|X^i - x_t^i, Y^i - y_t^i, x_t^i)), \quad (1)$$

where  $(X^i, Y^i)$  is a set of training samples,  $(x_t^i, y_t^i)$  is one of the samples in  $(X^i, Y^i)$ , and  $y_t^i$  is the difference between *MOCAP* data and Kinect data. Since both  $p(y_t^i|X^i, Y^i, x_t^i)$  and  $p(y_t^i|X^i - x_t^i, Y^i - y_t^i, x_t^i)$  are Gaussian Processes, we can solve the KL divergence in close form as:

$$\int p(y_t^i|X^i, Y^i, x_t^i) \log \frac{p(y_t^i|X^i, Y^i, x_t^i)}{p(y_t^i|X^i - x_t^i, Y^i - y_t^i, x_t^i)} dx. \quad (2)$$

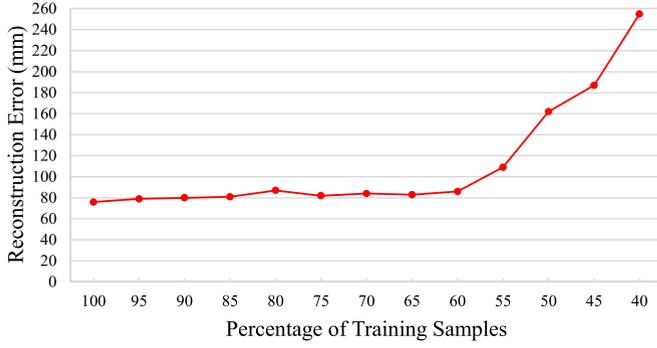


Fig. 3. Posture budgeting: We can shrink up to 40 percent of the training data while the mean error almost remains constant.

Further details of solving  $p(y_t^i|X^i, Y^i, x_t^i)$  can be found in Sections 4.1 and 4.2.

Fig. 3 shows the impact of posture budgeting. The unfiltered Walking motion database includes 1,120 training samples, which is reduced to 681. Nearly 40 percent of the training data can be pruned while maintaining a similar error level. For the details of the reconstruction error definition, please refer to Section 5.3.

## 4 POSTURE RECONSTRUCTION

To ensure that the reconstructed posture is accurate and resembles the input data from Kinect, we formulate the posture reconstruction as an optimization problem by minimizing an energy function. Such an energy function consists of three energy terms to constrain the solution space, which are the spatial prediction term, the temporal prediction term, and the reliability term. In the following, we will elaborate the definition and purpose of each term.

### 4.1 Spatial Prediction

Assuming that the *MOCAP* posture  $M_t$  is the corrected posture of the Kinect posture  $X_t$ , we design a spatial prediction term to evaluate how well the reconstructed posture fits with the *MOCAP* data, which implicitly favors solutions that are more similar to the correct posture.

Due to self-occlusions and sensor error, there exists a residual offset between  $X_t$  and  $M_t$ , which is calculated by  $Y_t = M_t - X_t$ , where  $Y_t = (y_t^1, y_t^2, \dots, y_t^j, y_t^i \in \mathbb{R}^3$ . During run time, the objective is to predict the residual offset  $Y_t$  so that we can obtain the reconstructed posture  $M_t$  by appending  $Y_t$  to  $X_t$ .

In this paper, we adopt the non-parametric GP as the predictor. More formally, let  $X^i = [x_1^i, \dots, x_T^i]^T$  be the input data of an arbitrary joint  $i$ , where  $T$  is the total number of frames. Let  $Y^i = [y_1^i, \dots, y_T^i]^T$  denote the output values such that  $y_t^i$  is the corresponding output of the input  $x_t^i$ . Here, we model the sensor error as the difference between the Mocap data and the Kinect data using Gaussian process, which transforms the input  $X^i$  of the  $i$ th joint into the output  $Y^i$  by:

$$y_t^i = f(x_t^i) + \epsilon, \quad (3)$$

where  $\epsilon \sim \mathcal{N}(0, \beta^{-1})$  is a noise variable, which is independent for each data point. The joint distribution of the output  $Y^i$  conditioned on input  $X^i$  is given by:

$$p(Y^i|X^i) = \int p(Y^i|f^i, X^i)p(f^i|X^i)df = \mathcal{N}(Y^i|0, K), \quad (4)$$

where  $K$  is the covariance matrix, in which the element  $cgqk(x_a^i, x_b^i)$  is defined as:

$$k(x_a^i, x_b^i) = \theta_0 \exp\left(-\frac{1}{2}(x_a^i - x_b^i)^T W (x_a^i - x_b^i)\right) + \theta_1 + \beta^{-1}\delta_{ab}, \quad (5)$$

where  $a$  and  $b$  are indices of training samples of joint  $i$ ,  $\delta_{ab}$  is Kronecker's delta function,  $W$  is kernel width,  $\theta_0$  is signal noise,  $\theta_1$  is a constant bias. At the training stage, with the obtained training data from Kinect and *MOCAP*, we can learn the hyper-parameters of  $\Phi = \{\theta_0, \theta_1, W, \beta\}$  by maximizing the log marginal likelihood:

$$\log p(Y^i|X^i, \Phi) = -\frac{1}{2}Y^{iT}K^{-1}Y^i - \frac{1}{2}\log|K| + \mathcal{C}, \quad (6)$$

where  $K$  is the covariance matrix defined in (5) and  $\mathcal{C}$  is a constant. Obviously, the computational cost of learning GP is dominated by the cubic complexity of computing the inverse of covariance matrix  $K^{-1}$ .

Human body joints are highly coordinated and it is important to take into account the relationship between them. Here, given an arbitrary joint, we use its neighboring joints for prediction. Specifically, given a joint  $i$  at time  $t$ ,  $x_t^i$ , its neighboring joints  $N(x_t^i)$  are defined as the set of joints that are directly connected with the same bone segment as joint  $i$ . Therefore, the input feature  $x_t^i$  for obtaining  $y_t^i$  of joint  $i$  is the union set of  $\tilde{x}_t^i$  and  $N(\tilde{x}_t^i)$ , where  $N(\tilde{x}_t^i)$  is the normalized position of the neighboring joints for joint  $i$ . The input data of joint  $i$  is thus defined as  $X^i = [(\tilde{x}_{t_1}^i, N(\tilde{x}_{t_1}^i)), \dots, (\tilde{x}_{t_n}^i, N(\tilde{x}_{t_n}^i))]^T$ , where  $t_1, \dots, t_n$  are time slices. The output data  $Y^i$  correspond to  $X^i$  of the prediction model. To simplify notation, we use  $x_*^i = (\tilde{x}_*^i, N(\tilde{x}_*^i))$  to denote new input of the  $i$ th joint at time  $t_*$  and use  $y_*^i$  to represent the corresponding output of  $x_*^i$  in the remaining parts of the paper.

With the learned model, we formulate the above prediction for  $y_*^i$  as a conditional probability distribution, yielding the spatial prediction energy term of the  $i$ th joint as defined below:

$$E_S^i = \ln p(y_*^i|X^i, Y^i, x_*^i) \sim \mathcal{N}(\mu(x_*^i), \sigma(x_*^i)), \quad (7)$$

where

$$\mu(x_*^i) = k(x_*^i, X^i)K^{-1}Y^i = k(x_*^i, X^i)\alpha \quad (8)$$

$$\sigma(x_*^i) = k(x_*^i, x_*^i) - k(x_*^i, X^i)K^{-1}k(x_*^i, X^i)^T \quad (9)$$

are the predicted mean and variance respectively,  $K$  is the covariance matrix defined in (5),  $\alpha = K^{-1}Y^i$  is the so-called prediction vector that can be pre-calculated from training samples, and the predicted mean is determined by the vector  $k(x_*^i, X^i)$ .

The term  $E_S$  ensures that the reconstructed postures are similar to the correct postures as much as possible. Predicting the offset of each joint reduces the searching space compared with inferring individual joints directly. The use of the weighted local GP models allows synthesizing

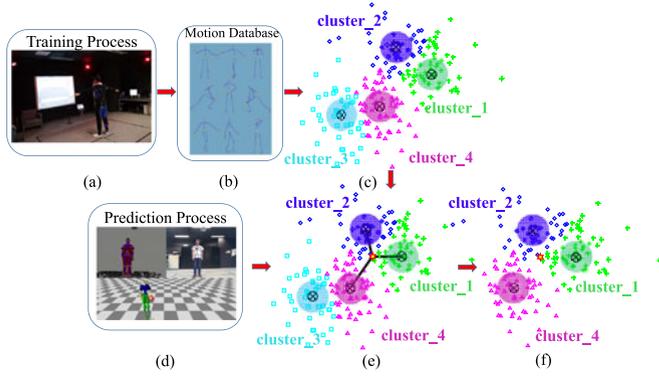


Fig. 4. Overview of the local mixture of GP models. (a-b) We capture postures by the MOCAP system and Kinect at the same time to generate the training samples. (c) At the training stage, we partition the samples into  $Q = 4$  local regions by K-means and learn  $Q$  local GPs with  $S = 10$  training samples independently. (d) During prediction, we extract feature of the  $i$ th = 7 (left hand) joint,  $x_*^i = (\tilde{x}_*^i, N(\tilde{x}_*^i))$ . (e) For a given test sample,  $x_*^i$ , shown in red star, we find the nearby  $L = 3$  local models by similarity measurement defined in (10). (f) We compute the local predicted mean by the  $l$ th local GP model and then generate the weighted mean prediction  $\mu(x_*^i)$  using  $L$  nearby local models given by (13).

variations in postures based on the motion database. There are several publicly available implementations of Gaussian Process. In this paper, we used the library developed by Lawrence [40].

## 4.2 Incremental Learning of Local Gaussian Processes

The major problem of using full GP is its cubic learning complexity of the inverse covariance matrix  $K^{-1}$  in (6). Here, we propose a new method based on a local mixture of Gaussian Processes that has the following advantages: 1) The local GP models are created by partitioning the posture space into  $Q$  local regions using clustering algorithm, and training  $Q$  local GP models independently. This relieves the cubic computational cost for learning the full GP model. 2) Since we use the weighted average prediction of nearby local models in which only a small number of training samples are involved, the prediction process is fast. 3) With the use of local models, it becomes possible to incrementally update a specific local GP with the complexity of  $O(S^2)$ , where  $S$  is the size of local GP. With the newly added predicted samples, the system accuracy can be enhanced for run-time postures that are different from those in the database.

Our algorithm consists of three major parts: 1) learning the hyper-parameters of local GP models; 2) performing the weighted prediction of local GP models; 3) incremental updating of corresponding local GP models, that is, adding a new sample into the closest local model and updating the inverse covariance matrix  $K^{-1}$  in (8). Fig. 4 shows an example of applying the local mixture of GP models. To simplify illustration, we project the 3D joint positions  $\mu$  onto the  $XZ$  plane.

### 4.2.1 Learning of Local Gaussian Processes

We cluster the training samples into  $Q$  regions and learn the hyper-parameters at each local region. In our system,  $Q$  is empirically set as 30. Similar to Section 4.1, the hyper-parameters of  $\Phi$  for each local model can be estimated by maximizing the log marginal likelihood in each local region as defined in (6).

Here, we explain how to partition the samples into different local regions. Given a new input  $x_*^i$  of joint  $i$ , assigning  $x_*^i$  to the  $q$ th local region is straightforward by measuring the similarity between  $x_*^i$  and the center of cluster  $C_q$ . Here, we use the Gaussian kernel to measure the similarity, which is in the same form of (5):

$$\text{similarity}(q) = \exp\left(-\frac{1}{2}(x_*^i - C_q)^T W(x_*^i - C_q)\right), \quad (10)$$

where  $C_q$  is the center of the  $q$ th cluster ( $q \in Q$ ) and  $W$  is the kernel width.

To speed up the run-time computation, we learn these hyper-parameters as an offline process named GP-offline. Its computational complexity is the summation of the complexity of clustering,  $O(QdN)$ , and the complexity of learning  $Q$  local GP models,  $O(QS^3)$ , where  $Q$  is the number of local models,  $d$  is the dimension of input,  $N$  is the total training data, and  $S$  is the size of each local model. The first part of Algorithm 1 summarizes the training of local GP models, where the  $kmeans(X^i, Q)$  function partitions the  $X^i$  into  $Q$  clusters, and returns an index set  $\zeta_q$  of samples for the  $q$ th local model.

### Algorithm 1. Local mixture of GP models and prediction

- 1 **Offline: Learning of hyper-parameters**
- 2  $Q$ : total number of local GP models
- 3  $C_Q$ : the center of each local GP model
- 4  $\zeta_q$ : the index set of samples for the  $q$ th local model
- 5  $(C_Q, \zeta_Q) = kmeans(X^i, Q)$
- 6 **for**  $q = 1$  to  $Q$  **do**
- 7    $\{\bar{\phi}^q\} \leftarrow \max(\ln p(Y_{\zeta_q}^i | X_{\zeta_q}^i, \bar{\phi}^q))$
- 8 **end**
- 9 **Online: Prediction of a new input of joint  $i$**
- 10 **Input:** new input,  $x_*^i$ , of joint  $i$
- 11  $L$ : the number of nearby local models
- 12  $S$ : the size of training samples for each local GP model
- 13 **for**  $l = 1$  to  $L$  **do**
- 14   Compute the similarity to the center of  $l$ th cluster:
- 15    $\text{similarity}(l) = \exp(-\frac{1}{2}(x_*^i - C_l)^T W(x_*^i - C_l))$
- 16   Compute the local predicted mean by the  $l$ th local model and  $\zeta_l$  is the index set of  $l$ th local training samples:
- 17    $\mu_l(x_*^i) = k(x_*^i, X_{\zeta_l}^i)K_{\zeta_l, \zeta_l}^{-1}Y_{\zeta_l}^i$
- 18    $\sigma_l(x_*^i) = k(x_*^i, x_*^i) - k(x_*^i, X_{\zeta_l}^i)^T K_{\zeta_l, \zeta_l}^{-1}k(x_*^i, X_{\zeta_l}^i)$
- 19 **end**
- 20 Compute the weighted prediction of new input  $x_*^i$  of joint  $i$  by the  $L$  local models:
- 21  $y_*^i = \sum_{l \in L} (\text{similarity}(l) / \sum_{\eta \in L} \text{similarity}(\eta)) \mathcal{N}(\mu_l(x_*^i), \sigma_l(x_*^i))$

### 4.2.2 Prediction of Local Gaussian Processes

Note that the mean of the prediction in (8) can be written as a function of  $Y^i$ :

$$\mu(x_*^i) = \sum_{t=1}^T w_t^i y_t^i, \quad (11)$$

where  $w_t^i$  is the  $t$ th element of  $k(x_*^i, X^i)K^{-1}$ ,  $T$  is the total number of frames. In this view, the mean of the prediction distribution is determined by the weighted combination of the  $N$  training outputs. We therefore propose to interpret the full GP as a global voting process, in which all weighted training outputs contribute to the decision of the test sample  $x_*^i$  of joint  $i$ . We observed that local neighborhoods behave

similarly such that nearby samples are likely to have similar output. With this insight, the full GP could be locally approximated by a small number of GPs near a given feature  $x_*^i$  of joint  $i$ , which could significantly reduce the computational cost and speed up the prediction.

Here, we explain how to determine the vote of local models. Similar to the model in [41], the contribution of each local model is determined by the distance to each local model. Given a new input  $x_*^i = (\tilde{x}_*^i, N(\tilde{x}_*^i))$  of joint  $i$ , the weight of each local GP model can be determined by the normalized distance to  $l$ th local model. Specifically, we compute the averaged prediction of  $L$  nearby local models by  $\mu(x_*^i) = E\{\mu_l|x_*^i\} = \sum_{l \in L} \mu_l(x_*^i) p(l|x_*^i)$  where  $\mu_l(x_*^i)$  is the predicted mean using the  $l$ th local model given by (11) and  $p(l|x_*^i)$  is the weight of each local GP, which is given by:

$$p(l|x_*^i) = \text{similarity}(l) / \sum_{\eta \in L} \text{similarity}(\eta), \quad (12)$$

where  $\text{similarity}(l)$  measures the similarity between  $x_*^i$  and the center of the  $l$ th local model given by (10). Hence, we calculate the weighted prediction by:

$$\begin{aligned} y_*^i &= \sum_{l \in L} \left( \frac{\text{similarity}(l)}{\sum_{\eta \in L} \text{similarity}(\eta)} \right) \mathcal{N}(\mu_l(x_*^i), \sigma_l(x_*^i)) \\ &= \sum_{l \in L} \sum_{\zeta \in \mathcal{S}} \left( \frac{\text{similarity}(l)}{\sum_{\eta \in L} \text{similarity}(\eta)} \right) w_{l\zeta}^i y_{l\zeta}^i, \end{aligned} \quad (13)$$

where  $L$  is total number of the nearby local models,  $\mathcal{S}$  is the size of training samples in each local model,  $\zeta$  is the index of local training samples in  $\mathcal{S}$ ,  $w_{l\zeta}^i$  is the element of  $k(x_*^i, X_\zeta^i) K_{\zeta, \zeta}^{-1}$ , and  $y_{l\zeta}^i$  is one of the offsets in  $\mathcal{S}$  that belongs to  $l$ th local model. The prediction process is summarized in the second part of Algorithm 1.

The prediction of our model is computationally inexpensive as those local models are learnt from a very small set of neighborhoods.  $L$  and  $\mathcal{S}$  are parameters of our model, and typical small values are sufficient to generate satisfactory results. In our implementation, the size  $\mathcal{S}$  of each nearby local model is 50 and the number of nearby local models  $L$  is 9. The influence of  $L$  and  $\mathcal{S}$  are discussed in Section 5.3.

#### 4.2.3 Incremental Updating of Local Gaussian Processes

One limitation of the data-driven method for posture reconstruction is that the reconstruction quality might drop significantly if we cannot find similar postures in the database. To relieve this problem, our model should be able to learn from the newly estimated samples such that we are more likely to adapt to unknown postures that are different from those in the database.

Here, we explain the major process of incremental updating of local GP models. During the local GPs learning process, we learn the hyper-parameters of  $\Phi$  and factorize the covariance matrix  $K$  by (14). At the prediction stage, the local models would predict the offset  $y_*^i$  given a new input  $x_*^i$ . During the incremental updating process, we preserve the samples  $(x_*^i, y_*^i)$  with high reliability and low predictive

TABLE 1  
Computational Complexity: The Main Computational Cost of Our Method is the Offline Learning while the Incremental Updating is Fast

	Proposed Method	Full GP
Learning	$O(2QS^3 + QdN)$	$O(N^3)$
Prediction	$O(LS^2)$	$O(N^2)$
Incremental Updating	$O(S^2)$	N/A

variance and append it into the nearest local model using the similarity measurement given by (10).

We calculate the similarity between  $x_*^i$  and the mean of each local Gaussian Process. If the similarity values with all local GPs are smaller than a predefined threshold  $w_{\text{similar}}$ , we create a new local model centers at  $x_*^i$ . Otherwise, we update the local GP with the highest similarity value. Notice that during the incremental learning process, the number of newly added samples can be further reduced by posture budgeting introduced in Section 3.2.

To update a local GP, we need to first update both the prediction vector and the mean of the local model. To update the prediction vector  $\alpha = K^{-1}Y^i$ , we adapt [42] in which the  $K^{-1}$  is updated by adjusting Cholesky factorization. As  $K$  is a symmetric, positive-definite matrix, we can uniquely factorize  $K$  as:

$$K = U^T U, \quad (14)$$

where  $U$  is an upper triangular matrix with positive diagonal elements. We then update the mean of the corresponding local model.

Given a new input  $x_*^i$  of joint  $i$ , we need to add additional rows and columns to  $K$  and  $U$  as follows:

$$K_{\text{new}} = \begin{bmatrix} K & \mathbf{k}_{\text{new}} \\ \mathbf{k}_{\text{new}}^T & k_{\text{new}} \end{bmatrix} \quad (15)$$

$$U_{\text{new}}^T = \begin{bmatrix} U^T & 0 \\ u^T & u_* \end{bmatrix}, \quad (16)$$

where  $\mathbf{k}_{\text{new}} = k(X^i, x_*^i)$ ,  $k_{\text{new}} = k(x_*^i, x_*^i)$ . Then, we can solve  $u$  and  $u_*$  by completing  $K_{\text{new}} = U_{\text{new}}^T U_{\text{new}}$  as:

$$Uu = \mathbf{k}_{\text{new}}, u_* = \sqrt{k_{\text{new}} - u^T u}. \quad (17)$$

Once we have solved  $U_{\text{new}}$ , we can update the prediction vector  $\alpha$  in  $U_{\text{new}}^T U_{\text{new}} \alpha_{\text{new}} = Y^{i_{\text{new}}}$  through back-substitution. The cost of back-substitution for a local model is  $O(S^2)$ , where  $\mathcal{S}$  is the number of training samples in a local model. Finally, we recalculate the corresponding local model using (8).

Table 1 compares the complexity of the full GP and our method. The computation of the Cholesky factorization is  $O(QS^3)$ , where  $Q$  is the number of local GP models and  $\mathcal{S}$  is the number of training samples in a local model. The prediction cost is  $O(LS^2)$ , where  $L$  is the number of nearby local GPs given an input. Thus, the offline learning complexity,  $O(2QS^3 + QdN)$ , dominates the main computational complexity of our method. The cost of incremental updating is  $O(S^2)$  due to the update of Cholesky factorization, which enables our system to incrementally update a specific local

Gaussian Process in real time. Algorithm 2 summarizes the incremental learning of local models.

---

**Algorithm 2.** Incremental learning of local models
 

---

```

1 Input: new input,  $x_*^i$ , of joint  $i$ 
2  $L$ : the number of nearby local models of  $x_*^i$ 
3  $C_l$ : the center of the  $l$ th local model, where  $l \in L$ 
4  $Q$ : total number of local GP models
5  $\mathcal{B}$ : the training samples  $\mathcal{B} = (X^i, Y^i)$  and  $\mathcal{B}_q$  represents the
   samples of the  $q$ th local model
6 Predict the offset,  $y_*^i$ , by  $L$  nearby local models (see
   Algorithm 1)
7 for  $l = 1$  to  $L$  do
8    $similarity(l) = \exp(-\frac{1}{2}(x_*^i - C_l)^T W(x_*^i - C_l))$ 
9 end
10 Find the most similar  $j$ th local model
11  $max_{similar} = \max(similarity)$ 
12 if  $max_{similar} < w_{similar}$  then
13   Create a new local model:
14    $C_{Q+1} = \{x_*^i\}$ 
15    $\mathcal{B}_{Q+1} = \{(x_*^i, y_*^i)\}$ 
16 else
17   Append  $\{x_*^i, y_*^i\}$  to the nearest local model  $j$ 
18    $\mathcal{B}_{j_{new}} = \{\mathcal{B}_j; (x_*^i, y_*^i)\}$ 
19   Update the mean of  $j$ th model
20    $C_{j_{new}} = mean(X_{j_{new}}^i)$ 
21   Update  $\alpha = K^{-1}Y^i$  of the  $j$ th local model:
22   Compute  $u$ ,  $u_*$ , and  $U_{new}$ 
23   Compute  $\alpha_{new}^i$  by back-substitution
24 end

```

---

### 4.3 Temporal Prediction

The above spatial prediction considers each posture independently. To ensure the temporal smoothness between consecutive frames, the relationship between frames is modeled as a second order temporal model, which has been verified to be effective in preserving temporal smoothness [43]. Specifically, we adopt a constant velocity variation to smooth velocity, which is formulated as below:

$$E_T = \ln p(M_t | M_{t-1}, M_{t-2}). \quad (18)$$

$M_t$ ,  $M_{t-1}$ , and  $M_{t-2}$  are the reconstructed postures at time slices  $t$ ,  $t-1$ , and  $t-2$ . We have the following relationship between the reconstructed posture, input posture and the residual offset:

$$M_t = Y_t + X_t. \quad (19)$$

Therefore, we can rewrite (18) as :

$$\begin{aligned}
 E_T &= \ln p(Y_t + X_t | M_{t-1}, M_{t-2}) \\
 &= \|(M_t - M_{t-1}) - (M_{t-1} - M_{t-2})\|^2 \\
 &= \|M_t - 2M_{t-1} + M_{t-2}\|^2 \\
 &= \|Y_t - (-X_t + 2M_{t-1} - M_{t-2})\|^2
 \end{aligned} \quad (20)$$

which facilitates the continuity in the reconstructed motions.

### 4.4 Reliability Embedding

The accuracy of each tracked joint is different depending on the degree of occlusion. The incorrectly tracked joints from

Kinect will incorrectly guide the system to infer the joint positions. The residual offset,  $Y_t = M_t - X_t$ , of the correctly tracked joints should be smaller as they are closer to the corrected posture, namely  $M_t$ . Thus, it is essential to consider the reliability of each joint to constrain the residual offsets of these joints with higher confidence during the prediction of  $Y_t$ . We use a reliability term  $E_R$  to penalize the residual offset of each joint based on its reliability, which implicitly ensures that the reconstructed posture resembles the input posture from Kinect as much as possible. More specifically, the residual offset value  $y_t^i$  of joint  $i$  should be smaller if the corresponding joint is with higher reliability.

We adopt the strategy proposed by [7] to evaluate the reliability of the tracked joints from Kinect. They evaluate the reliability in three aspects: behavior reliability, kinematics reliability, and tracking state reliability. The behavior reliability refers to abnormal behavior of a tracked joint, which is calculated by the cosine similarity between two consecutive displacement vectors of one joint. The kinematics reliability represents the kinematic correctness of the tracked joints, which measures the change of bone length for bones connecting with the joint. The tracking state reliability tells if a joint is tracked, inferred or not tracked when it is completely occluded. More details about the calculation of the reliability of each joint can be found in [7]. As a result, the reliability rate of each joint is a value between 0.0 and 1.0 (inclusive). We embed the reliability of each joint into the optimization framework and formulate the following reliability term:

$$E_R = \|RY_t\|_F^2. \quad (21)$$

$\|\cdot\|_F$  is the Frobenius norm. The entry of  $R$  is the reliability of each joint, which ensures the reconstructed posture does not deviate from the input posture from Kinect. Intuitively, while minimizing the objective function, the value of  $y_t^i$  tends to be small when its reliability value is large.

### 4.5 Energy Minimization Function

With the terms defined in the above sections, the posture reconstruction problem is formulated as the following optimization function:

$$E = \arg \min_{Y_t} \{w_S E_S + w_T E_T + w_R E_R\}, \quad (22)$$

where  $w_S$ ,  $w_T$ , and  $w_R$  are the weights of the energy terms. In our implementation, they are empirically set to be 0.6, 0.2, and 0.2, respectively. We optimize (22) by using the gradient descent method. We sample a number of potential postures in the solution space in each iteration. The posture that minimizes the cost function will be considered as the initial posture sample in the next iteration. Our posture reconstruction system is frame-based. The initial posture for optimization at each frame is defined as the previous reconstructed posture, which allows the system to have higher chance to find the optimized posture. The optimization procedure stops when an optimal solution is found or the number of iterations reaches a predefined threshold.

There are some principles to tune the values of the weights. The weight of the spatial prediction term should

be set the largest, since this term drags the reconstructed posture to the corrected posture as closely as possible. Second, the temporal prediction term ensures the temporal stability of the posture sequences. The reliability term makes sure the reconstructed posture is as similar as the Kinect posture, since the primary purpose of the system is to reconstruct Kinect postures. We will evaluate how these terms affect the accuracy of the system in Section 5.5.

The proposed framework for posture reconstruction is summarized here. At the offline stage, we learn a spatial prediction model using Gaussian Process with pairwise Kinect data and marker-based motion capture data. It ensures that the reconstructed posture is as accurate as the *MOCAP* data. We also embed the temporal and reliability terms in offline process so as to generate temporal smoothness and reliable postures. At the online stage, the system obtains an optimized posture with live captured data from Kinect, which ensures the reconstructed posture resembles the input posture from Kinect while maintaining the temporal smoothness between previous frames.

## 5 EXPERIMENTAL RESULTS

In this section, we will show the experimental results and present the comparisons with alternative approaches including Kinect SDK [2], as well as the algorithms proposed by [19] and [9]. We first show postures with severe self-occlusions reconstructed by our approach. Qualitative and quantitative analysis were conducted to evaluate the accuracy.

The experimental results were conducted on a desktop computer with Intel Core 2 Duo 3.17 GHZ processor. If not otherwise mentioned, we use Kinect official SDK [2] to obtain posture data. Here, we consider the Kinect device as one additional reflective marker of the optical motion capture system to eliminate the interference between Kinect and the optical motion capture system. The setup environment of Kinect and optical motion capture system is shown in Fig. 2.

### 5.1 Posture Reconstruction

The proposed approach works for users with different body sizes and proportions, because we normalize and retarget the Kinect input posture as explained in Section 3.1. We evaluate our system on a wide range of human motions, including sports activity such as Tai Chi, bending, golf swinging, and daily actions such as crossing arms, waving right hand, clapping hands, rolling hands up and down, rolling hands forward and backward. The number of frames in the training database used in our method, [19] and [9] are reported in Table 2, which shows that our database is 35 percent smaller than that of Zhou et al. [9], and 83 percent smaller than that of Shen et al. [19]. In addition, our newly proposed local mixture of GPs algorithm allows us to combine all types of training motion in Table 2 as a single database, while in [9] a separate training database is built for each type of motion.

We choose these motions because all these motions contain severe self-occlusions, which are not well tracked by the Kinect system. However, the proposed method can well reconstruct these inaccurate postures even if a number of

TABLE 2  
The Number of Frames for Each Type of Training Motion Database Used in Our Method, Shen et al. 2012, and Zhou et al. 2014

Motions	Our method	Zhou et al. 2014	Shen et al. 2012
Tai Chi	411	650	2,320
Bending	201	320	1,580
Golf Swinging	296	460	1,765
Crossing Arms	245	380	1,685
Waving Right Hand	221	350	1,650
Clapping Hands	270	420	1,720
Rolling Hand Up and Down	308	480	1,840
Rolling Hand Forward and Backward	306	475	2,050
Bending Leg	243	385	1,890
<b>Mixed motion database</b>	<b>2,501</b>	<b>-</b>	<b>16,500</b>

joints cannot be tracked by the Kinect sensor. Fig. 5 showed several frames of our results, the lower right avatar represents the postures reconstructed by our method and the lower left avatar corresponds to the estimated posture by Kinect SDK [2]. The upper half shows the RGB and depth images respectively. We can observe that certain parts of the postures from Kinect SDK are twisted when there exist occlusions while our method can reconstruct the postures very well.

### 5.2 Qualitative Analysis

In this section, we evaluate the perceptual score for the correctness of postures reconstructed by our method, postures from Kinect, postures by the method proposed by [9], [19] and postures captured by an optical motion capture system.

In order to evaluate the perceptual correctness according to the user performed motion, we measure the perceptual score for the postures of each method using a survey-based evaluation. Such an experiment has also been performed in [7] and [9]. Notice that while some recent research such as [44] analyzes how viewers perceive interactions between virtual characters, since the focus of our research is about posture reconstruction process from noisy data, we do not include detailed perceptual analysis in the scope of this research.

A total of 15 participants were invited to conduct this experiment. All of them had little or no experience about motion capture and 3D animation. The purpose of this experiment is to assess the relative correctness of the obtained postures from these five methods. We create a set of posture sequences with these five methods together with the RGB video so that the participants know what the actual actions are. Participants were asked to give a score for each motion based on its correctness according to the performed motion without knowing what method is used. The score ranges from 1 to 10 (inclusive), where 1 means the most incorrect, and 10 means the most correct.

The score distribution for Kinect SDK [2], [19], [9], our method and *MOCAP* is shown in Fig. 6. The overall average scores of these five methods are 5.20, 6.42, 7.51, 7.49 and 9.16 respectively, and the standard deviations are 1.187, 0.578, 0.236, 0.240, and 0.255. As expected, *MOCAP* data achieve the best scores. We can see that our method performs better than Kinect and [19] in general. In particular,

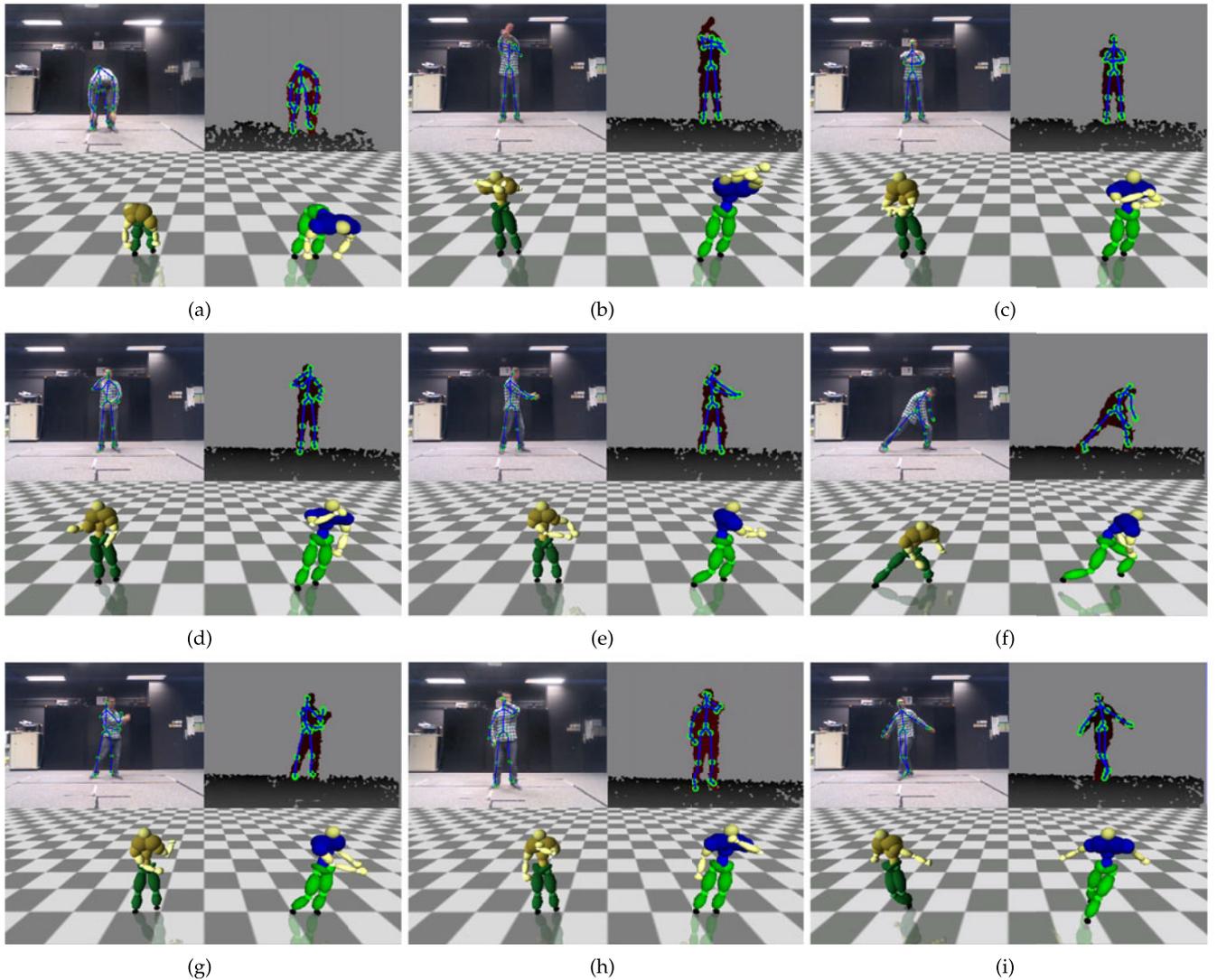


Fig. 5. Postures from Kinect and their corresponding reconstructed postures. In each picture, the upper half shows the RGB and depth images, in which the blue skeleton is the tracked results from Kinect. The lower left and right parts represent the 3D Kinect posture and our reconstructed posture respectively. (a) Bending over; (b) Crossing arms; (c) Rolling hands forward and backward; (d) Rolling hands up and down; (e) Clapping hands; (f) Bending leg; (g) Golf swinging; (h) Waving left hand; (i) Walking.

our method significantly outperforms Kinect and [19] for motions with more occlusions such as bending over and rolling hands, as shown in Figs. 5a, 5c, 6b and 6h. The reason is that we embed the reliability term into our optimization framework, which implicitly ensures the system to recover these joints more than those with higher reliability. As shown in Table 2, our method generates postures of similar quality compared with [9] with a significantly smaller

motion database. It should be noted that for motion that involves a large range of movement such as Tai Chi, our method could synthesize postures that are closer to the ground truth compared to [9]. This is because the weighted prediction of local models allows synthesizing postures that are not available in the motion database and more possible solutions are explored.

### 5.3 Quantitative Analysis

In this section, we quantitatively analyze the correctness of the proposed method. We assume the data from optical motion capture system is the ground truth data. To evaluate the accuracy of the reconstructed postures, we define an error function to measure the distance between reconstructed postures and ground truth postures:

$$E(F_1, F_2) = \frac{1}{IT} \sum_{i=1}^I E_i(F_1, F_2), \quad (23)$$

where  $F_1$  and  $F_2$  are the two sets of postures,  $I$  is the total number of joints, and  $T$  is the total number of postures.  $E_i$

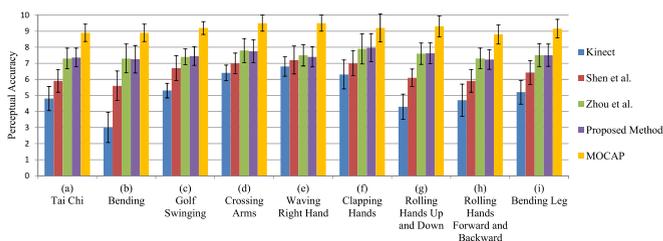


Fig. 6. The perceptual score for the correctness of postures from Kinect, Shen et al. 2012, Zhou et al. 2014, proposed method, and an optical motion capture system.

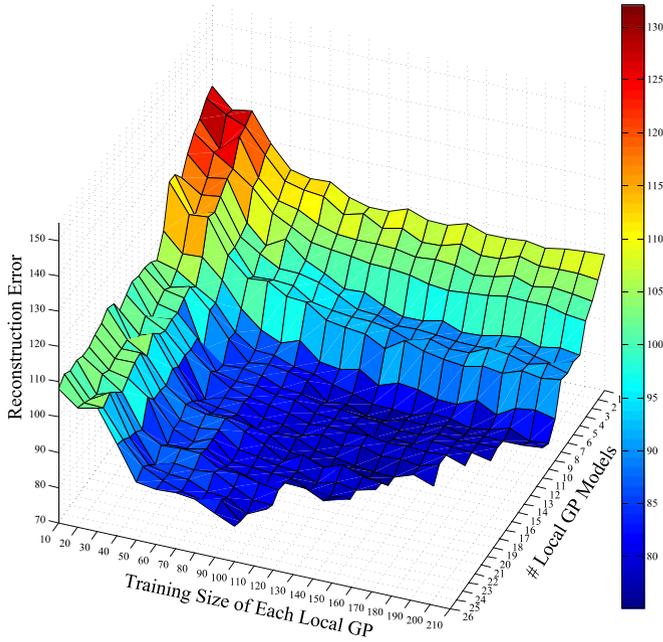


Fig. 7. Influence of the training size and the number of local GP models on the 3D joint reconstruction error.

is the reconstruction error of joint  $i$  between two set of postures, which is defined as:

$$E_i(F_1, F_2) = \sum_{t=1}^T D(F_{1t}^i, F_{2t}^i), \quad (24)$$

where  $F_{1t}^i$  is the  $i$ th joint of the posture at time  $t$  from  $F_1$ .  $D$  is the Euclidean distance between two joints of two postures:

$$D(P_1^i, P_2^i) = \sqrt{(P_{1x}^i - P_{2x}^i)^2 + (P_{1y}^i - P_{2y}^i)^2 + (P_{1z}^i - P_{2z}^i)^2}. \quad (25)$$

With the error function defined in (23), we first study the influence of the training size of each local model,  $S$ , and the number of local models,  $L$ , on the reconstruction error. Fig. 7 shows that when we fix the  $S$  for each local model, the 3D joint reconstruction error decreases as the  $L$  increases. Similarly, for any specific  $L$ , the system accuracy can be enhanced by increasing  $S$ . However, the improvement is not significant when  $S$  is raised to 50 and  $L$  reaches 9, because the postures become redundant and do not contribute to the reconstruction process. Thus, the value of  $S$  and  $L$  are empirically set to 50 and 9 respectively.

As an example, Fig. 8 shows the trajectory of the left hand in a golf swinging movement using offline local GPs (LGP-offline) and local GPs with incremental

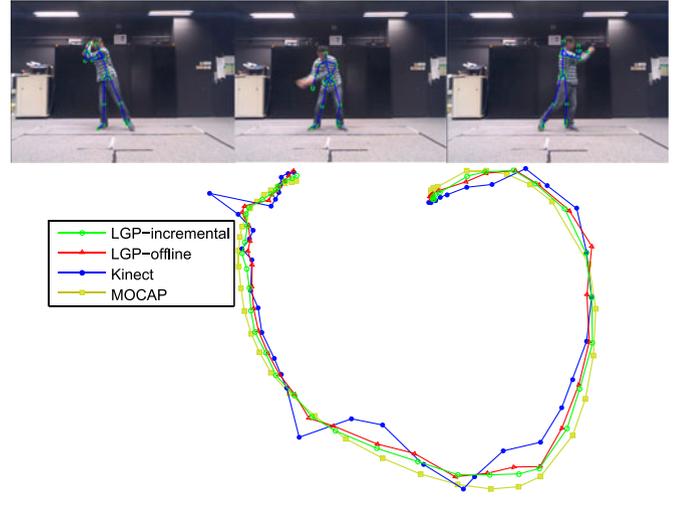


Fig. 8. Trajectory of the left hand when performing golf swinging motion.

updating (LGP-incremental). We can see that the ground truth data (MOCAP) is smooth and the Kinect data is noisy due to the self-occlusions and sensor error. The mean error of Kinect, LGP-offline, and LGP-incremental is 12.36, 8.1, and 7.7 cm. Compared with LGP-offline, the LGP-incremental is closer to the ground truth in general, which verify the effectiveness of the incremental learning framework. We can also see that a small number of local GP models ( $L = 9$ ) is sufficient to reconstruct postures.

More comparisons of different type of testing motions between LGP-offline, LGP-incremental, [19], and [9] can be found in Table 3. Here we choose five types of motion for evaluation: clapping hands, crossing arms, bending, Tai Chi, and waving right hand. As expected, the error of Kinect was large in general. Our method outperforms [19] as we take into account the reliability of each joint such that the inaccurately tracked joints will not guide the system to infer the postures. For all classes of motions, our method consistently outperforms the Kinect and [19], which verifies the effectiveness of the proposed method in terms of reconstruction accuracy. It should be noted that the LGP-incremental can generate comparable system accuracy compared to [9] while the running time is less than [9]. The computational time of [9] and our system are 37 and 29 ms per frame, respectively.

#### 5.4 Comparison Between Randomized Forests and Our Method

In this particular experiment, we do not use Kinect SDK to extract joint positions. To ensure a fair comparison between our method and randomized forests, which is the method

TABLE 3  
Reconstruction Error of Kinect, Shen et al. 2012, Zhou et al. 2014, and the Proposed Method on the Testing Data Sets

Motion Type	Number of Frames for Testing	Kinect (cm)	Shen et al. (cm)	Zhou et al. (cm)	Proposed Method (cm)	
					LGP-offline	LGP-incremental
Crossing Arms	2,052	12.5	9.8	7.2	7.9	7.4
Bending Over	1,835	13.7	9.5	8.4	9.2	8.7
Tai Chi	2,885	14.5	10.2	7.5	8.0	7.4
Waving Right Hand	1,568	12.5	8.8	6.5	6.9	6.6

TABLE 4  
Reconstruction Error of Randomized Forests and Our Method Using Five-fold Cross Validation (cm)

Motion Type	Reconstruction Error with Randomized Forests	Reconstruction Error with Our Method
Crossing Hands	13.8	8.1
Golf Swinging	14.9	9.4
Waving Hands	13.5	7.3
Rolling Hand	13.7	7.4
Left and Right		
Clapping Hands	14.3	8.5
T-pose	7.8	7.1

used to train Kinect, we construct a common training database for both methods.

Our database contains a large number of synthetic depth images that are created as follow. First, we create a 3D mesh model in which each body part is labeled. Second, we retarget Mocap data to drive the movement of the 3D mesh model. Third, we render depth information of the scene into depth images frame by frame. Since our 3D model comes with body part labels, we can automatically label body part information for each pixel in the rendered depth images. Finally, we trained the randomized forests with the labeled depth images and estimated the joint positions using mean shift. We also trained our GP models with the same data and the joint positions found by mean shift using the body parts estimated by randomized forests. Our training database consists of 17K synthesized depth images generated by Mocap data, including actions such as golf swing, waving hands, crossing hands and clapping hands.

We use five-fold cross validation to compare the performance of randomized forests [3] and our algorithm. Table 4 shows the comparison of average reconstruction error. It can be observed that both methods have similar performance for simpler motions such as T-pose. However, for more challenging motions that involve self-occlusion such as crossing hands and golf swing, our method generates better reconstruction results with smaller reconstruction error.

### 5.5 Effects of Optimization Terms

In this section, we analyze the reconstruction accuracy by examining the effectiveness of different terms in the objective function of (22). We used Tai Chi motion, bending over and crossing arms for evaluation because of their complicated movement features. The results are reported in Table 5.

We found that both the temporal prior term and the reliability term improve the reconstruction accuracy, especially for the movements with severe self-occlusions. Although setup (c) achieves better results than setup (b), the obtained movements are jerky, because setup (c) predicts postures

TABLE 5  
Reconstruction Error of the Proposed Framework with Different Constraint Terms

Setup	Terms Used	Reconstruction Error (cm)
(a)	$E_S$	12.0
(b)	$E_S, E_T$	10.5
(c)	$E_S, E_R$	9.7
(d)	$E_S, E_T, E_R$	7.9

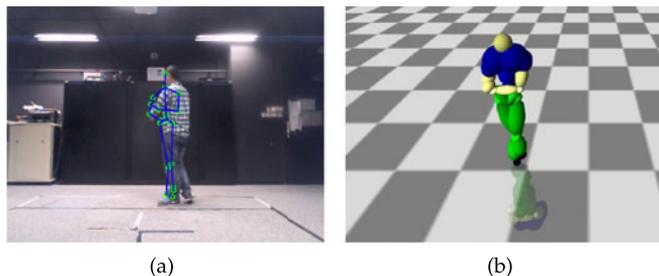


Fig. 9. Turning around motion. (a) The RGB image of turning around motion (facing backward); (b) The tracking result of Kinect corresponding to (a) (facing forward).

independently without considering relationship between consecutive frames.

## 6 CONCLUSIONS AND DISCUSSIONS

In this paper, we present a probabilistic framework to reconstruct live captured postures from Kinect. Postures from Kinect are noisy, however, such a noise is not a random signal and we observed that there are some underlying patterns in it. In this research, we can thus assume that the noisy data contain useful information in helping us to find the solution. Then, we apply a machine learning algorithm to learn the correlation between Kinect data and Mocap data so as to predict the offset given Kinect data. Finally, we verify our assumption with accurately reconstructed posture results.

To overcome the problem of incorrectly tracked and missing joints in Kinect, we adopt Gaussian Process model as a spatial prior to leverage position data obtained from Kinect and an optical motion capture system. Specifically, we model the residual offset between postures obtained from Kinect and MOCAP system instead of using pairwise posture relationship. While GP works well in small training data sets, it is not competent in systems that require a large database, such as motion-based gaming, due to its high computational complexity. To solve this problem, we propose a new method based on the local mixture of Gaussian Processes to speed up the learning and prediction. Our system allows incrementally updating of local models in real time, which boosts the reconstruction accuracy of run-time postures that are different from those in the database.

For our method to work well, the motion performed by the user should belong to one of the action classes in the database. The proposed method is useful for real-time applications such as motion-based gaming and sport training where the user is expected to perform a motion from a set of common moves that are known in advance. While our system utilizes neighboring joints for prediction, it is difficult to deal with heavily occluded postures such as turning around, in which there are only few valid joints. As shown in Fig. 9, Kinect incorrectly recognizes the posture. The incorrect joint positions and the decrease of reliability of body joint greatly impact the recognition quality. In such cases, the amount of correct data present is so little that our system cannot produce very good result. One possible solution would be using multiple Kinects to capture postures from different directions.

There is still room to improve the proposed reconstruction system. The assigned weights for the terms in the objective function are empirically set to be fixed in the proposed system. However, the weights can be different for different types of motion to obtain optimal reconstructed postures. One possible improvement would be to formulate the weights as a function of the residual offset, which is used to measure the importance of each term. Therefore, the weights can be adaptively determined according to the type of motion. The incorporation of physical constraints into the proposed framework is another interesting direction as the reconstructed postures in this work are not necessary physically correct. One possible implementation would be modeling the physical attributes (i.e., force field) between the Kinect data and *MOCAP* data as a prior distribution, and embed them in the optimization framework to generate physically valid postures. Last but not least, integrating our system with other simple yet stable devices such as inertia-based *Mocap* system would be an interesting topic, because Kinect can only detect limited range of movements while motion sensor can be used as a complement, e.g., detecting the occluded body part.

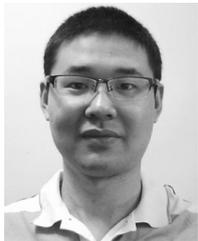
## ACKNOWLEDGMENTS

The work described in this paper was partially supported by a grant from City University of Hong Kong (Project No. 7004548) and the Engineering and Physical Sciences Research Council (EPSRC) (Ref: EP/M002632/1).

## REFERENCES

- [1] J. Chan, H. Leung, J. Tang, and T. Komura, "A virtual reality dance training system using motion capture technology," *IEEE Trans. Learn. Technol.*, vol. 4, no. 2, pp. 187–195, Apr. 2011.
- [2] (2013). Microsoft Corporation, "Kinect for Xbox 360." [Online]. Available: <http://www.xbox.com/en-US/xbox-360/accessories/kinect>
- [3] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 1297–1304.
- [4] T. Morgan, D. Jarrell, and J. Vance, "Poster: Rapid development of natural user interaction using kinect sensors and vrpn," in *Proc. IEEE Symp. 3D User Interfaces*, Mar. 2014, pp. 163–164.
- [5] J. Chai and J. K. Hodgins, "Performance animation from low-dimensional control signals," in *Proc. ACM SIGGRAPH 2005 Papers*, 2005, pp. 686–696.
- [6] H. Liu, X. Wei, J. Chai, I. Ha, and T. Rhee, "Realtime human motion control with a small number of inertial sensors," in *Proc. Symp. Interactive 3D Graph. Games*, 2011, pp. 133–140.
- [7] H. P. H. Shum, E. S. L. Ho, Y. Jiang, and S. Takagi, "Real-time posture reconstruction for microsoft kinect," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1357–1369, Oct. 2013.
- [8] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA, USA: MIT Press, 2005.
- [9] L. Zhou, Z. Liu, H. Leung, and H. P. H. Shum, "Posture reconstruction using kinect with a probabilistic model," in *Proc. 20th ACM Symp. Virtual Real. Softw. Technol.*, 2014, pp. 117–125.
- [10] I. Tashev, "Kinect development kit: A toolkit for gesture- and speech-based human-machine interaction [best of the web]," *IEEE Signal Process. Mag.*, vol. 30, no. 5, pp. 129–131, Sep. 2013.
- [11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 559–568.
- [12] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013.
- [13] S. W. Bailey and B. Bodenheimer, "A comparison of motion capture data recorded from a vicon system and a microsoft kinect sensor," in *Proc. ACM Symp. Appl. Perception*, 2012, pp. 121–121.
- [14] J. Kim, Y. Seol, and J. Lee, "Human motion reconstruction from sparse 3d motion sensors using kernel CCA-based regression," *Comput. Animation Virtual Worlds*, vol. 24, no. 6, pp. 565–576, 2013.
- [15] T. Helten, M. Muller, H.-P. Seidel, and C. Theobalt, "Real-time body tracking with one depth camera and inertial sensors," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1105–1112.
- [16] M. Sigalas, M. Pateraki, I. Oikonomidis, and P. Trahanias, "Robust model-based 3d torso pose estimation in rgb-d sequences," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Dec. 2013, pp. 315–322.
- [17] H. P. H. Shum and E. S. L. Ho, "Real-time physical modelling of character movements with microsoft kinect," in *Proc. 18th ACM Symp. Virtual Real. Softw. Technol.*, 2012, pp. 17–24.
- [18] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 1092–1099.
- [19] W. Shen, K. Deng, X. Bai, T. Leyvand, B. Guo, and Z. Tu, "Exemplar-based human action pose correction and tagging," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1784–1791.
- [20] H. Yasin, B. Krüger, and A. Weber, "Model based full body human motion reconstruction from video data," in *Proc. 6th Int. Conf. Comput. Vis. / Comput. Graph. Collaboration Techn. Appl.*, 2013, pp. 1:1–1:8.
- [21] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 188:1–188:12, Nov. 2012.
- [22] L. Bo and C. Sminchisescu, "Twin Gaussian processes for structured prediction," *Int. J. Comput. Vis.*, vol. 87, no. 1-2, pp. 28–52, 2010.
- [23] L. Bo and C. Sminchisescu, "Structured output-associative regression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 2403–2410.
- [24] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, 2003, pp. 750–757.
- [25] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh, "Pose machines: Articulated pose estimation via inference machines," in *Proc. Comput. Vis.*, 2014, pp. 33–47.
- [26] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1653–1660.
- [27] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Trans. Graph.*, vol. 33, no. 5, p. 169, 2014.
- [28] J. Quiñero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *The J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, 2005.
- [29] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse gaussian process methods: The informative vector machine," in *Proc. 16th Annu. Conf. Neural Inf. Process. Syst.*, 2003, pp. 609–616.
- [30] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1257–1264.
- [31] V. Tresp, "Mixtures of Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 654–660.
- [32] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, vol. 2, pp. 881–888.
- [33] R. Urtasun and T. Darrell, "Sparse probabilistic regression for activity-independent human pose inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2008, pp. 1–8.
- [34] L. Bo, C. Sminchisescu, A. Kanaujia, and D. Metaxas, "Fast algorithms for large scale conditional 3d prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2008, pp. 1–8.
- [35] E. Snelson, "Local and global sparse Gaussian process approximations," in *Proc. Artif. Intell. Statist.*, 2007, pp. 524–531.
- [36] D. Nguyen-Tuong, J. R. Peters, and M. Seeger, "Local Gaussian process regression for real time online model learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1193–1200.

- [37] S. Schaal and C. G. Atkeson, "From isolation to cooperation: An alternative view of a system of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 605–611.
- [38] L. Sigal, R. Memisevic, and D. J. Fleet, "Shared kernel information embedding for discriminative inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 2852–2859.
- [39] (2015). Motion Analysis Corporation [Online]. Available: <http://www.motionanalysis.com>
- [40] N. Lawrence. (2014). Gaussian processes tool-kit [Online]. Available: <http://cran.r-project.org/web/packages/gpstk>
- [41] X. Zhao, Y. Fu, and Y. Liu, "Human motion tracking by temporal-spatial local Gaussian process experts," *IEEE Trans. Image Process.*, vol. 20, no. 4, pp. 1141–1151, Apr. 2011.
- [42] M. Seeger, "Low rank updates for the cholesky decomposition," University of California at Berkeley, Berkeley, CA, USA, Tech. Rep, EPFL-REPORT-161468, 2004.
- [43] H. Sidenbladh and M. Black, "Learning image statistics for Bayesian tracking," in *Proc. 8th IEEE Int. Conf. Comput. Vis.*, 2001, vol. 2, pp. 709–716.
- [44] L. Hoyet, R. McDonnell, and C. O'Sullivan, "Push it real: Perceiving causality in virtual interactions," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 90:1–90:9, Jul. 2012.



**Zhiguang Liu** is currently working toward the PhD degree in the Department of Computer Science, City University of Hong Kong. His research interests include character animation, computer graphics, and machine learning.



**Liuyang Zhou** received the PhD degree in computer science from the City University of Hong Kong in 2014. He is currently a researcher of Wisers Research, Wisers Information Limited, Hong Kong. His research interests include deep learning, computer vision and computer animation, focusing on large-scale object recognition and detection, human motion retrieval, and character animation synthesis.



**Howard Leung** received the BEng degree in electrical engineering from McGill University, Canada, in 1998, the MSc degree and the PhD degree in electrical and computer engineering from Carnegie Mellon University in 1999 and 2003, respectively. He is currently an assistant professor in the Department of Computer Science at the City University of Hong Kong. He is supervising the 3D Motion Capture Laboratory at City University of Hong Kong. His current research interests include 3D Human motion analysis and retrieval, intelligent tools for chinese handwriting education, web-based learning technologies and brain informatics. He has received the Best Paper Award during the 31st Computer Graphics International (CGI 2014).



**Hubert P.H. Shum** He received the MSc and BEng degrees from the City University of Hong Kong and the PhD degree from the School of Informatics in the University of Edinburgh. He is a senior lecturer (associate professor) at Northumbria University. Before joining the university, he worked as a lecturer in the University of Worcester, a post-doctoral researcher in RIKEN Japan, as well as a research assistant in the City University of Hong Kong. His research interests include character animation, machine learning, human motion analysis, and computer vision.

▷ **For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).**