Exemplar-Based Human Action Pose Correction

Wei Shen, Ke Deng, Xiang Bai, Tommer Leyvand, Baining Guo, *Fellow, IEEE,* and Zhuowen Tu *Member, IEEE,*

***Abstract*—The launch of Xbox Kinect has built a very successful computer vision product and made a big impact to the gaming industry; this sheds lights onto a wide variety of potential applications related to action recognition. The accurate estimation of human poses from the depth image is universally a critical step. However, existing pose estimation systems exhibit failures when faced severe occlusion. In this paper, we propose an exemplar-based method to learn to correct the initially estimated poses. We learn an inhomogeneous systematic bias by leveraging the exemplar information within specific human action domain. Furthermore, as an extension, we learn a conditional model by incorporation of pose tags to further increase the accuracy of pose correction. In the experiments, significant improvements on both joint-based skeleton correction and tag prediction are observed over the contemporary approaches, including what is delivered by the current Kinect system. Our experiments for facial landmark correction also illustrate that our algorithm is applicable to improve the accuracy of other detection/estimation systems.**

***Index Terms*—Kinect, random forest, pose correction, skeleton, pose tag.**

I. Introduction

WITH the development of high-speed depth cameras [1], the computer vision field has experienced a new opportunity of applying a practical imaging modality for building a variety of systems in gaming, human computer interaction, surveillance, and visualization. A depth camera provides depth information as different means to color images captured by the traditional optical cameras. Depth information gives extra robustness to color as it is invariant to lighting and texture changes [2], although it might not carry very detailed information of the scene.

The human pose estimation/recognition component is a key step in an overall human action understanding system. High speed depth camera with the reliable estimation of the human skeleton joints [3] has recently led to a new consumer electronic product, the Microsoft Xbox Kinect [1].

Even though the depth data provides invariant and informative cues, existing systems (e.g. classification-based approaches for skeleton joint detection [3]) are not all satisfactory due to severe occlusions. Fig. 1 illustrates a pipeline of pose recognition, which includes three important steps: (1) background removal, (2) initial pose estimation, and (3) pose correction. After the background is removed from the depth image, skeleton joints are estimated from the foreground depth information using e.g. [3]–[7]. Note that, serious errors exist in the estimated skeleton in Fig. 1(c). The pose correction stage further takes in these initial per-frame estimations (as "noisy" input) and tries to correct the errors and deliver more robust results. In this paper, we focus on the third step, *pose correction*, which also plays a very important role in the overall pose detection/recognition pipeline. To show to what extent the third stage can help, the pose correction stage performs "de-noising" by working on extracted "noisy" skeleton joints only, without looking back at the original depth data.

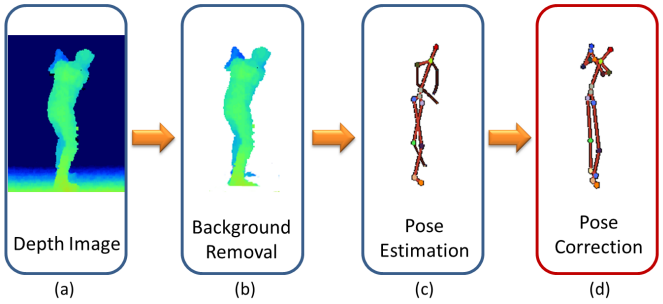


Fig. 1. Pose recognition pipeline. From a depth image of single frame, the pipeline includes background removal, initial pose estimation, and pose correction. The skeleton joints marked by color dots in (c) are the ones with high confidence in estimation whereas the ones without color dots are with the low confidence.

To perform pose correction, e.g. obtaining the result shown in Fig. 1(d) from a noisy estimation in Fig. 1(c), two types of information can be leveraged: (1) temporal motion consistency and (2) the systematic bias. Using the temporal information to enforce the motion consistency has been extensively studied in the literature [8], [9] but studying the systematic bias has received relatively less attention. Generally, the biases are nonlinear and are associated with complex data manifolds. As illustrated in Fig. 2, the systematic biases do exist, especially in domain specific actions.

For a general estimation problem, its systematic bias might be significant and has an explicit analytic function [10]. In our task of human action pose recognition, each pose is represented by a number of skeleton joints; each joint is estimated/extracted using local and contextual depth features [3]. The bias estimation problem in our task observes two properties: (1) human action has certain (sometimes strict) regularity especially when some actions, e.g. golf swing or football throwing, are performed, and (2) the bias is not homogeneous in the data manifold. For example, when a person is facing the camera with no occlusion, the initial estimations are accurate; when a person is standing in a side-

W. Shen is with the Department of Communication Engineering, Shanghai University, Shanghai, 200072, China. E-mail: wei.shen@shu.edu.cn.

K. Deng and T. Leyvand are with Microsoft Corporation, Redmond, WA. E-mail:{kedeng,tommerl}@microsoft.com.

X. Bai is with the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, Hubei Province, 430074, China. Email: xiang.bai@gmail.com.

B. Guo and Z. Tu are with Microsoft Research Asia, Beijing, 100091, China. Z. Tu is also with the Department of Cognitive Science, University of California, San Diego, CA. Email: bainguo@microsoft.com, zhuowen.tu@gmail.com.

view with certain hand motion, there is severe occlusion and the initial estimation may not be all correct, as illustrated in Fig. 1. In this paper, the main contribution is learning the inhomogeneous bias function to perform pose correction for one domain specific action and we emphasize the following four points: (1) exemplar-based approach serves a promising direction for pose correction in depth images, (2) learning an inhomogeneous regression function should naturally perform data-partition, abstraction, and robust estimation. With a thorough experimental study, our approach shows encouraging results, (3) learning an regression function conditioned on a incorporated global parameter gives more nature data partition, thus further improve the performance of pose correction, (4) our regression based approach is general, it's applicable to correcting not only the pose estimated from Kinect sensor, but also estimation errors involved from other sensors.

The remainder of this paper is organized as follows: Sec. II reviews the works related to human pose estimation and correction. Sec. introduces the skeleton data and ground truth used in our work. Sec. IV describes the proposed approach to pose correction in detail. The experimental results on joint-based skeleton correction, pose tag prediction and facial landmark correction are presented in Sec. V. Finally, Sec. VI draws conclusions and points out potential directions for future research.

This paper extends our preliminary work [11] with the following contributions: (1) the proposal of a conditional regression model as well as the cascaded one, which further enhance the pose correction performance, (2) tag prediction experiments on a new football throwing data set, (3) landmark correction experiments to show the generality of the proposed method.

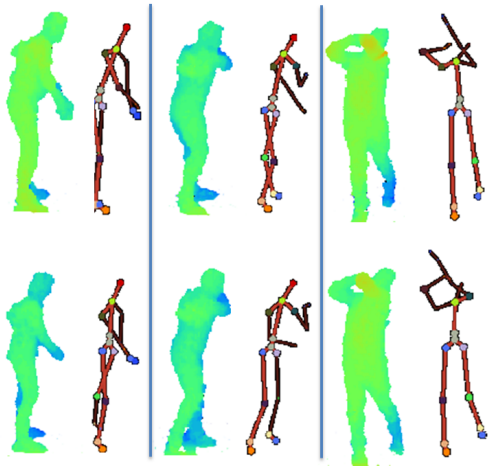


Fig. 2. Three groups of poses. In each group, the poses are similar, and the errors of the estimated skeletons are somewhat systematic, e.g. the right forearms of the skeletons in the second group and those in the third group.

II. Related Work

Motion and action recognition from optical cameras has been an active research area in computer vision; typical approaches include 3D feature-based [12], part-based (poselets) [13], and segment-based [14] approaches. Although insights can be gained, these methods are not directly applicable to our problem.

Benefit from rapid development of the real-time depth camera, action recognition from depth images has been a hot topic in recent years. Shotton et al. [3] cast pose estimation as a per-pixel classification problem. They compute the depth comparison feature [15] of each pixel, then classify each pixel into human parts. Finally the modes of probability mass for each part are take as the joint position proposals. Girshick et al. [4] improve Shotton's method, they predict the offset between each pixel and each joint by regression instead. Sun et al. [5] follows Girshick's regression-based method, they lean the regression model conditioned on several global parameters, such as height and torso orientation. Ye et al. [6] propose to search the most similar pose in the database by matching the cloud points of the depth images and then refine the best matching pose to be the estimated pose. Baak et al. [7] also propose a data driven approach for pose estimation, instead of matching cloud points, they extract features from the depth data and then measure the similarity between the extracted features. A recent survey on the human action analysis with Kinect depth sensor can be found in [16]. These methods provide the direct input for our method, so our method will benefit from the improvement of these pose estimation methods.

From a different view, bias estimation has been a long standing problem in statistics [10]. Related work in the pose correction task uses physical-model-based approaches [17], Kalman-like filters [18], or exemplar-based approaches but with very specific design, which is hard to adapt to the general task [19]. Here we adopt the random forest regressor, which takes in both the estimated solutions and their estimation uncertainties. We show significant improvement over other regressors such as nearest neighborhood [20], Gaussian process regression [21], support vector regression [22], and logistic regression [23]. Our approach is real-time and can be directly applied to the Kinect system.

III. Data

In this section, we introduce the data used for our pose correction problem. The recently launched Kinect camera [1] is able to give 640×480 image at 30 frames per second with depth resolution of a few centimeters. Employing the Kinect camera, we are able to generate a large number of realistic depth images of human poses. The human skeleton estimated from the depth image by the current Kinect system [3] is the direct input for our approach, which is called *ST* (Skeleton esTimation) in the rest of this paper. As shown in Fig. 3(a), there are 20 body joints in a skeleton, including hips, spine, shoulders, head, elbows, wrists, hands, knees, ankles, and feet.

As suggested by the recent work [3], [4], the ground truth positions of the body joints can be captured by motion capture (mocap) system. We obtain a set of mocap of human actions as the ground truth of the estimated skeletal joints. The mocap data is also called *GT* (Ground Truth) in the rest of this paper. In our experiments, we limit the rotation of the user to $\pm 120^\circ$ in both training and testing data. Fig. 3(b) shows several pairs of *ST* and *GT*.

Fig. 3. Skeleton data. (a) A template skeleton and its joints. The skeleton is a simple directed graph, in which the directions are denoted by the arrows beside the skeleton edges. For denotational simplicity, we do not show the arrows in other figures. (b) Several pairs of input noisy skeletons (upper) and ground truth skeletons (lower).

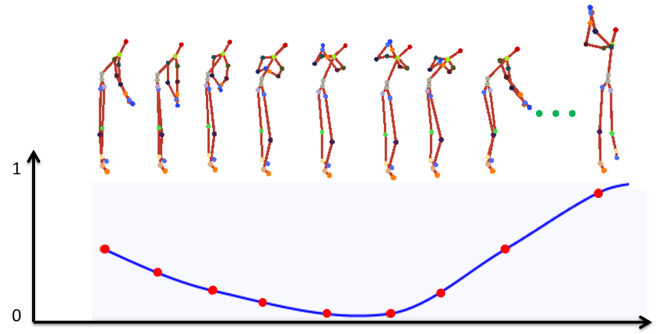


Fig. 4. Pose tag. Each type of poses are assigned to a tag, e.g. the tag of "front swing" (the last pose) is about 0.95.

As for gaming applications, skeletons with 20 joints are too much data for games to process, since they usually only need one or two global parameters to describe the motion. So pose tag is introduced to be used to drive the avatar to put on the same pose as the player performs in the somatosensory game. The pose tag is a real value ranging from 0.0 to 1.0, indicating a specific stage in a particular action, e.g. golf swing. In a domain of specific action, each type of poses is assigned a tag manually. For example, the tag value of the poses of "front swing" and "back swing" are within the intervals $[0.8\ 1.0]$ and $[0.0\ 0.2]$ respectively. The accurate value is determined by the amplitude of swing. Fig. 4 shows the tag values of a typical golf swing action. We will demonstrate how to enhance pose correction by incorporating pose tags.

IV. Pose Correction

A. Objectives

We focus on two tasks: joint-based skeleton correction and pose tag prediction. Our inputs are m estimated skeletons $\mathbf{st} = (ST_1, \ldots, ST_m)$ from a video sequence of m depth image frames. Each skeleton estimation ST includes n ($n = 20$ here) joints: $ST = (\hat{\mathbf{x}}_j, c_j; j = 1, \ldots, n)$, where $\hat{\mathbf{x}}_j \in \mathbb{R}^3$ denotes the world coordinates of the jth body joint, as shown in Fig. 3. c_j indicates the confidence for the estimation $\hat{\mathbf{x}}_j$ by the skeleton joint detector, i.e. if joint j has high confidence, $c_j = 1$; Otherwise, $c_j = 0$. For example, in Fig. 1(c), the skeleton joints marked by color dots are the ones with high confidence in estimation whereas the ones without color dots are with the low confidence.

The first task (joint-based skeleton correction) is to predict the "true" position of each joint: $\hat{\mathbf{x}}_j \rightarrow \mathbf{x}_j$ and the "true" skeletons $\mathbf{gt} = (GT_1, \ldots, GT_m)$ where each $GT = (\mathbf{x}_j; j = 1, \ldots, n)$ and $\mathbf{x}_j \in \mathbb{R}^3$. In training, we are given a training set of $\{(\mathbf{st}, \mathbf{gt})_k\}_{k=1}^K$ of K pairs of $\mathbf{st}$ and $\mathbf{gt}$; in testing, we want to predict the "true" $\mathbf{gt}$ from a given input $\mathbf{st}$.

The second task (pose tagging) is to predict the pose tag $\boldsymbol{\Upsilon} = (\Upsilon_1, \ldots, \Upsilon_m)$ from a given $\mathbf{st} = (ST_1, \ldots, ST_m)$. In training, we are given a training set of $\{(\mathbf{st}, \boldsymbol{\Upsilon})_k\}_{k=1}^K$ of K pairs of $\mathbf{st}$ and $\boldsymbol{\Upsilon}$; in testing, we want to predict the tag values $\boldsymbol{\Upsilon}$ from a given input $\mathbf{st}$.

The tag is actually a low dimensional manifold coordinate of the pose. As for gaming applications, tag predication is important for games even when perfect skeletons are available. Both of the two tasks are performed to recover the pose from a noisy initial estimation, so we categorize them into the tasks of pose correction.

B. Normalized Skeleton Joints Coordinates

From the world coordinates, we want to have an intrinsic and robust representation. Based on the $n = 20$ joints, we show the kinematic skeleton template as displayed in Fig. 3(a), which is a directed graph. Each joint is a node of the graph. Given an $ST = (\hat{\mathbf{x}}_j, c_j; j = 1, \ldots, n)$, we compute its normalized coordinates for our problem, denoted as $H(ST) = (\mathbf{r}_j, c_j; j = 1, \ldots, n)$. Since $\hat{\mathbf{x}}_j$ denotes the world coordinate, we normalize them to the template to remove not only the global translation but also the variation in individual body differences. We use the skeleton joint C. Hip as the reference point, the origin, $\mathbf{r}_1 = (0, 0, 0)$, and map the other joints to the sphere as $\mathbf{r}_j = \frac{\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{j_o}}{\|\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{j_o}\|_2}$ where joint j_o is the *direct predecessor* of joint j on the skeleton (directed graph).

The design of the transformed coordinates $H(ST)$ is motivated by the kinematic body joint motion. $H(ST)$ observes a certain level of invariance to translation, scaling, and individual body changes. We can actually drop $\mathbf{r}_1$ since it is always on the origin. One could add other features computed on ST

to $H(ST)$ to make it more informative but this could be a topic of future research.

C. Joint-based Skeleton Correction

1) Joint offset inference: To perform skeleton correction from a noisy input ST, instead of directly predicting the "true" positions of the body joints, we infer the offsets between the joints in the ST and those in the GT. This has its immediate significance: when a person is facing the camera with no occlusion, ST is actually quite accurate, and thus has nearly zero difference to GT; when a person is playing the game in side view with severe occlusions, there is often a large and inhomogeneous difference between ST and GT. This is essentially a manifold learning problem. Certain clusters of ST on the manifold can directly be mapped to, e.g. very low values, if we would predict the offsets; predicting the direct coordinates of GT however would have to explore all possible ST in the data space.

Now we show how to compute the offsets between the joints in $ST = (\hat{\mathbf{x}}_j, c_j; j = 1, \ldots, n)$ and those in $GT = (\mathbf{x}_j; j = 1, \ldots, n)$, where $\hat{\mathbf{x}}_j, \mathbf{x}_j \in \mathbb{R}^3$ are the world coordinates of joint j in ST and GT, respectively. To ensure the scale invariance of the offsets, skeletons are normalized based on the default lengths of the skeleton edges in the template shown in Fig. 3(a). First, we choose a set of stable joints $J_S = \{$Spine, C. Shoulder, Head, L. Hip, R. Hip, L. Knee, R. Knee, L. Ankle, R. Ankle $\}$. We call them stable joints because other joints in the ST, such as Hand and Wrist, are often occluded by the human body, thus the skeleton edge lengths between them are prone to errors. Given an ST, for each skeleton edge between the stable joints and their direct predecessors, we compute the proportion to the template skeleton edge length: $\lambda(j, j_o) = \|\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{j_o}\|_2 / \|\mathcal{T}_j - \mathcal{T}_{j_o}\|_2$, where $\mathcal{T}_j$ is the jth joint for the template $\mathcal{T}$ (shown in Fig. 3), which is fixed in this problem. Suppose that there is no error in the ST, then for each $j \in J_S$, $\lambda(j, j_o)$ is nearly identical. However, the estimation error may result in a serious bias when computing $\lambda(j, j_o)$. Although we only consider the stable joints, their estimated positions may also be wrong. Therefore, we should exclude the joints with large estimation error, and average the skeleton edge lengths between the rest joints to obtain a robust measure of the scale of the ST. To this end, the scale proportion of the ST is formulated by

$$\lambda(ST) = \frac{\sum_{j \in J_S} \lambda(j, j_o) \cdot \delta(\|\lambda(j, j_o) - \frac{\sum \lambda(j, j_o)}{|J_S|}\|_1 \leq th)}{\sum_{j \in J_S} \delta(\|\lambda(j, j_o) - \frac{\sum \lambda(j, j_o)}{|J_S|}\|_1 \leq th)}, \tag{1}$$

where $\delta(\cdot)$ is an indicator function which is a robust measure to exclude the outliers and

$$th = 3\sqrt{\frac{\sum_{j \in J_S} (\lambda(j, j_o) - \frac{\sum \lambda(j, j_o)}{|J_S|})^2}{|J_S|}}. \tag{2}$$

Here we define the threshold th as the triple standard deviation to exclude the outliers according to the *3-sigma rule*. Finally, the offset of a joint j between the pair of $\hat{\mathbf{x}}_j$ and $\mathbf{x}_j$ is computed as

$$\mathbf{\Delta}_j = (\mathbf{x}_j - \hat{\mathbf{x}}_j)/\lambda(ST), \tag{3}$$

and $\mathbf{D} = (\mathbf{\Delta}_1, \ldots, \mathbf{\Delta}_n)$ for each skeleton of n joints. For the entire sequence of m images, we have $\mathbf{d} = (\mathbf{D}_1, \ldots, \mathbf{D}_m)$. Note that we do not need to explicitly align the pair of ST and GT, since they are obtained from the same depth image.

2) Learning the regression for joint offsets : In this section, we discuss how to learn a regression function to predict the offset to perform pose correction. We are given a training set of $\mathcal{S} = \{(\mathbf{st}, \mathbf{gt})_k\}_{k=1}^K$ of K pairs of $\mathbf{st}$ and $\mathbf{gt}$ (for denotational simplicity, we let $K = 1$ and thus k can be dropped for an easier problem understanding). Using the normalization step in Sec. IV-B, we obtain $h(\mathbf{st}) = (H(ST_1), \ldots, H(ST_m))$ where each $H(ST) = (\mathbf{r}_j, c_j; j = 1, \ldots, n)$; using the offset computing stage in Eq. 3, we compute the offset, $\mathbf{d} = (\mathbf{D}_1, \ldots, \mathbf{D}_m)$. Thus, our goal is to predict the mapping $h(\mathbf{st}) \rightarrow \mathbf{d}$.

We first learn a function to directly predict the mapping $f_d : H(ST) \rightarrow \mathbf{D}$ by making the independent assumption of each pose. From this view, we rewrite the training set as $\mathcal{S} = \{(H(ST_i), \mathbf{D}_i)\}_{i=1}^m$.

Random forest [24]–[26] includes an ensemble of tree predictors that naturally perform data-partition, abstraction, and robust estimation. For the task of regression, tree predictors take on target values and the forest votes for the most possible value. Each tree in the forest consists of split nodes and leaf nodes. Each split node stores a feature index with a corresponding threshold τ to decide whether to branch to the left or right sub-tree and each leaf node stores some predictions. Each leaf node stores a set of exemplars in a partitioned feature subspace with similar target values. The prediction of the tree is the abstraction of the target values of the exemplars within one leaf node. Therefore, it's proper to apply random forest regression to predict the inhomogeneous systematics bias.

Our objective is to learn a random forest regression function $f_d : H(ST) \rightarrow \mathbf{D}$. Following the standard greedy decision tree training algorithm [3], [4], [15], each tree in the forest is learned by recursively partitioning the training set $\mathcal{S} = \{(H(ST_i), \mathbf{D}_i)\}_{i=1}^m$ into left $\mathcal{S}_l$ and right $\mathcal{S}_r$ subsets according to the best splitting strategy $\theta^* = \arg\min_\theta \sum_{p \in \{l,r\}} \frac{|\mathcal{S}_p(\theta)|}{|\mathcal{S}|} e(\mathcal{S}_p(\theta))$, where $e(\cdot)$ is an error function standing for the uncertainty of the set and θ is a set of splitting candidates. If the number of training samples corresponding to the node (node size) is larger than a maximal κ, and $\sum_{p \in \{l,r\}} \frac{|\mathcal{S}_p(\theta^*)|}{|\mathcal{S}|} e(\mathcal{S}_p(\theta^*)) < e(\mathcal{S})$ is satisfied, then recurse for the left and right subsets $\mathcal{S}_l(\theta^*)$ and $\mathcal{S}_r(\theta^*)$, respectively.

The selection of the error function $e(\cdot)$ is important for learning an effective regressor. Here, we employ the rooted mean squared differences:

$$e(\mathcal{S}) = \sqrt{\frac{\sum_{i=1}^m \|\mathbf{D}_i - \frac{\sum_{i=1}^m \mathbf{D}_i}{|\mathcal{S}|}\|_2^2}{m}}. \tag{4}$$

In the training stage, once a tree t is learned, a set of training samples $S_t^{lf} = \{\mathbf{D}_i^{lf}\}_{i=1}^{|S_t^{lf}|}$ would fall into a particular leaf node lf. Obviously, it is not effective to store all the samples in S_t^{lf} for each leaf node lf. Instead, we would do an abstraction for the learning purpose. One choice is to store the mean

$\bar{\mathbf{D}}^{(lf)} = \sum_i \mathbf{D}_i^{lf} / |S_t^{lf}|$ of the set S_t^{lf}. One could store other abstractions such as the histogram of S_t^{lf} as well. In addition, each tree t would assign a leaf node label $L_t(H(ST_i))$ for a given $H(ST_i)$.

In the testing stage, given a test example $ST = (\hat{\mathbf{x}}_j, c_j; j = 1, \ldots, n)$, for each tree t, it starts at the root, then recursively branches left or right. Finally, it reaches the leaf node $L_t(H(ST))$ in tree t, then the prediction given by tree t is $F_t(H(ST)) = \delta(lf = L_t(H(ST))) \cdot \bar{\mathbf{D}}^{(lf)}$, where $\delta(\cdot)$ is an indicator function. The final output of the forest (T trees) is a probability function:

$$P_{H(ST)}(\mathbf{D}) = \frac{1}{T}\sum_{t=1}^{T} \exp(-\| \frac{\mathbf{D} - F_t(H(ST))}{h_D} \|_2^2), \quad (5)$$

where h_D is a learned bandwidth. The mean can be considered as another output of the learned regression function $f_d(H(ST)) = E_{P_{H(ST)}}[\mathbf{D}]$ where $E_{P_{H(ST)}}[\cdot]$ indicates the expectation. The corrected skeleton is obtained by (if we would use the $f_d(H(ST))$ as the output)

$$CT = ST^- + \lambda(ST) \cdot f_d(H(ST)), \quad (6)$$

where $ST^- = (\hat{\mathbf{x}}_j; j = 1, \ldots, n)$ and the components in CT are $CT = (\mathbf{z}_j; j = 1, \ldots, n)$. In the experiments, we refer to this method (using the $f_d(H(ST))$ as the output) as RFR.

3) Regression cascade: In the recent work [27], an algorithm named cascaded pose regression (CPR) is proposed, which iteratively trains a series of regressors to approach the ground truth. Inspired by CPR, we propose a regression cascade (RFRC) here.

As described in Sec. IV-C2, we learn a regression function $f_d : H(ST) \rightarrow \mathbf{D}$. Here, we rewrite it as

$$f_d^{(0)} : H(ST) \rightarrow \mathbf{D}^{(0)}. \quad (7)$$

Then we obtain the corrected skeleton $CT^{(1)}$ by

$$CT^{(1)} = ST^- + \lambda(ST) \cdot f_d^{(0)}(H(ST)). \quad (8)$$

Then we compute the normalized skeleton joint coordinates $H(CT^{(1)})$ and learn the second layer of regression function:

$$f_d^{(1)} : (H(ST), H(CT^{(1)})) \rightarrow \mathbf{D}^{(1)}, \quad (9)$$

where $\mathbf{D}^{(1)}$ is the offsets between $CT^{(1)}$ and GT computed by the stage in Eq. 3. Then repeat the process mentioned above. The $(i+1)$th layer of regression function is

$$f_d^{(i)} : (H(ST), H(CT^{(i)})) \rightarrow \mathbf{D}^{(i)}. \quad (10)$$

The output skeleton is

$$CT^{(i+1)} = CT^{(i)} + \lambda(ST) \cdot f_d^{(i)}(H(ST), H(CT^{(i)})). \quad (11)$$

For consistency, we define $CT^{(0)} = \emptyset$ and obtain

$$CT^{(i+1)} = ST^- + \lambda(ST) \cdot \sum_{\iota=0}^{i} f_d^{(\iota)}(H(ST), H(CT^{(\iota)})). \quad (12)$$

Fig. 5 shows an illustration for the process of the regression cascade.

4) Temporal constraint: Taking the motion consistency into account, we could use the temporal constraint to improve our correction results. Our learned regression function outputs a probability distribution as shown in Eq. (5), which can be used to employ the temporal information. Given the estimated skeleton sequence $\mathbf{st} = (ST_1, \ldots, ST_m)$, our goal is to obtain $h(\mathbf{st}) \rightarrow \mathbf{d}$, where $\mathbf{d} = (\mathbf{D}_1, \ldots, \mathbf{D}_m)$, with the corresponding corrected skeleton sequence $\mathbf{ct} = (CT_1, \ldots, CT_m)$. To meet the real time requirement, our approach follows a causal model, i.e. the current prediction only depends on past/current inputs/outputs. For the ith input estimated skeleton ST_i, its offset is computed as

$$\mathbf{D}_i = \begin{cases} f_d(H(ST_i)) & \text{if } i = 1 \\ \arg\min_{\mathbf{D} \in \mathbb{R}^{n \times 3}} E(\mathbf{D} | ST_i, ST_{i-1}, \mathbf{D}_{i-1}) & \text{otherwise} \end{cases}, \quad (13)$$

where $E(\cdot)$ is an energy function defined as

$$E(\mathbf{D}|ST_i, ST_{i-1}, \mathbf{D}_{i-1}) = \alpha \cdot (-\log(P_{H(ST_i)}(\mathbf{D}))) + (1-\alpha)\|ST_i^- + \lambda(ST_i)\mathbf{D} - (ST_{i-1}^- + \lambda(ST_{i-1})\mathbf{D}_{i-1})\|_2^2, \quad (14)$$

where α is a weight factor. We use coordinate descent to solve Eq. 14. Finally, the corrected skeleton CT_i is

$$CT_i = ST_i^- + \lambda(ST_i)\mathbf{D}_i. \quad (15)$$

In the experiments, we refer to the random forest regression and cascade methods under temporal constraint as RFRT and RFRCT respectively. In the cascaded method, the temporal constraint is only used in the regressor of the last layer and the regressors of the former layers use the expectation as the output.

D. Pose tag prediction

In this section we discuss how to learn a regression function to predict the tag of a pose. The learning process is the same as what we did for the skeleton correction; so we follow the notions in Sec. IV-C2 and IV-C4 except for replacing the offset $\mathbf{D}$ by the tag value Υ. As stated in the previous section: we are given a training set $\{(\mathbf{st}, \boldsymbol{\Upsilon})_k\}_{k=1}^K$ of K pairs of $\mathbf{st}$ and $\boldsymbol{\Upsilon}$ (for denotational simplicity, we let $K = 1$ and thus k can be dropped for easier problem understanding). Using the normalization step in Sec. IV-B, we obtain $h(\mathbf{st}) = (H(ST_1), \ldots, H(ST_m))$, where each $H(ST) = (\mathbf{r}_j, c_j; j = 1, \ldots, n)$. Thus our objective is to obtain $h(\mathbf{st}) \rightarrow \boldsymbol{\Upsilon}$, where $\mathbf{st} = (ST_i; i = 1, \ldots, m)$ and $(\boldsymbol{\Upsilon} = \Upsilon_i; i = 1, \ldots, m)$. Similarly, a random forest regression function to directly predict the tag only based on the individual skeleton $f_v : H(ST) \rightarrow \Upsilon$ is also learned first. Here, each leaf node in tree t also stores the mean tag values of the samples falling into the leaf node. In the testing stage, given a test example ST, the prediction $F_t(H(ST))$ given by tree t is also computed similarly as in Sec. IV-C2. The final output of the forest (T trees) is a regression in probability function:

$$P_{H(ST)}(\Upsilon) = \frac{1}{T}\sum_{t=1}^{T} \exp(-\| \frac{\Upsilon - F_t(H(ST))}{h_\Upsilon} \|^2), \quad (16)$$

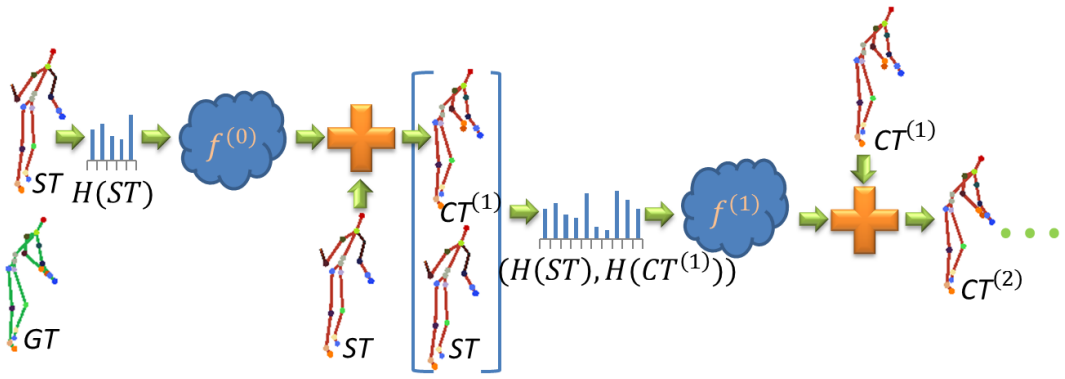


Fig. 5. Illustration of the corrected human skeleton updated at each iteration.

where h_Υ is a learned bandwidth. The mean is also considered as another output of the learned regression function $f_v(H(ST)) = E_{P_{H(ST)}}[\Upsilon]$.

A tag is a point on the manifold of the coherent motion, therefore the temporal constraint would be much more useful and effective in predicting the tag value. Our approach for tag prediction is also a cause model. Given the estimated skeleton sequence $\mathbf{st} = (ST_1, \ldots, ST_m)$, the goal is to obtain the tag sequence $(\Upsilon_1, \ldots, \Upsilon_m)$. For the ith input estimated skeleton ST_i, similarly Υ_i is predicted as:

$$\Upsilon_i = \begin{cases} f_v(H(ST_i)) & \text{if } i = 1 \\ \arg\min_{\Upsilon \in [0,1]} E(\Upsilon | ST_i, \Upsilon_{i-1}) & \text{otherwise} \end{cases}, \tag{17}$$

where

$$E(\Upsilon | ST_i, \Upsilon_{i-1}) = \alpha \cdot (-\log(P_{H(ST_i)}(\Upsilon))) + (1-\alpha)(\Upsilon - \Upsilon_{i-1})^2, \tag{18}$$

where α is the weight factor. The minimization of the above energy can be done similarly as before.

E. Skeleton correction conditioned on pose tag

In this section, we demostrate a extension version of the proposed regression based skeleton correction method. In domain specific actions, the tag is a global parameter of the pose. As we stated before, to perform pose correction, the learned regression function should naturally perform data-partition. According to tags, pose data space is naturally partitioned into several subspaces, therefore we can learn several conditional probabilities over the tag space instead of learning a regression function on the whole pose data space as we did in Sec. IV-C2. The motivation is that conditional probabilities are easier to learn since the regression trees do not have to deal with all pose variations. This is inspired by Dantone's work [28], which learns conditional regression forests to detect facial feature.

Now we discuss how to learn a regression function conditioned on pose tag to perform skeleton correction. For simplicity, we assume each pose is independent, so the temporal constraint is no longer considered here. Given the training set $\mathcal{S} = \{(H(ST_i), \mathbf{D}_i, \Upsilon_i)\}_{i=1}^m$, recall that a regression function for skeleton correction aims to learn the probability $P_{H(ST)}(\mathbf{D})$ (Eq. 5), while a conditional regression forest models the conditional probability $P_{H(ST)}(\mathbf{D}|\Upsilon)$ and estimates $P_{H(ST)}(\mathbf{D})$ by

$$P_{H(ST)}(\mathbf{D}) = \int P_{H(ST)}(\mathbf{D}|\Upsilon) P_{H(ST)}(\Upsilon) \mathrm{d}\Upsilon, \tag{19}$$

where $P_{H(ST)}(\Upsilon)$ is modeled by Eq. 16.

To learn the conditional probability $P_{H(ST)}(\mathbf{D}|\Upsilon)$, we discretize the tag space into several disjoint set $\mathcal{T}_k$ and then split the training set into subsets $\mathcal{S}_k$ according to the tag space discretization. Thus, Eq. 19 can be rewritten as

$$P_{H(ST)}(\mathbf{D}) = \sum_k (P_{H(ST)}(\mathbf{D}|\mathcal{T}_k) \int_{\Upsilon \in \mathcal{T}_k} P_{H(ST)}(\Upsilon) \mathrm{d}\Upsilon). \tag{20}$$

The conditional probability $P_{H(ST)}(\mathbf{D}|\mathcal{T}_k)$ is learned on each of the training subsets $\mathcal{S}_k$ by Eq. 5. Take the mean as the output of the forest learned on each training set, we can obtain the output of the learned conditional forest:

$$f_c(H(ST)) = \sum_k (f_{d_k}(H(ST) \int_{\Upsilon \in \mathcal{T}_k} P_{H(ST)}(\Upsilon) \mathrm{d}\Upsilon), \tag{21}$$

where $f_{d_k}(H(ST)) = E_{P_{H(ST),\mathcal{T}_k}}[\mathbf{D}]$ and recall that $E[\cdot]$ indicates the expectation. Finally, the corrected skeleton is obtained by

$$CT = ST^- + \lambda(ST) \cdot f_c(H(ST)). \tag{22}$$

We refer to this conditional random forest regression method as CRFR.

Since the correction process reduces the original estimation error, learning the regression model on the corrected skeletons instead should improve the accuracy of tag prediction. Then a more accurate tag prediction model leads to a more accurate conditional regression model for skeleton correction. This naturally brings about a cascaded regression model, in each layer of which we not only learn the difference between the ground truth and the corrected skeletons obtained by the previous layer, but also retrain the tag prediction model based on the corrected skeletons. We follow the notation in Sec. IV-C3 and the output of the $(i+1)$th layer of the cascaded

model is obtained by

$$CT^{(i+1)} = ST^{-} + \lambda(ST) \cdot \sum_{\iota=0}^{i} f_c^{(\iota)}(H(ST), H(CT^{(\iota)})), \tag{23}$$

where

$$f_c^{(\iota)}(H(ST), H(CT^{(\iota)}))) = \sum_k (f_{d_k}^{(\iota)}(H(ST), H(CT^{(\iota)})) \int_{\Upsilon \in \mathscr{T}_k} P_{H(ST),H(CT^{(\iota)})}(\Upsilon) \mathrm{d}\Upsilon), \tag{24}$$

Similar as we trained skeleton correction model in Sec. IV-C3, here we concatenate normalized skeleton joint coordinates of the corrected skeletons $H(CT^{(\iota)})$ to $H(ST)$ to train the tag prediction model of each layer: $P_{H(ST),H(CT^{(\iota)})}(\Upsilon)$. We refer to the proposed cascaded conditional random forest regression method as CRFRC.

V. Experimental Results

In this section, we show the experimental results and give the comparisons between alternative approaches, including what is delivered in the current Kinect system. In the remainder of this section, unless otherwise specified, we set the parameters for learning random forest as: the number of trees $T = 50$ and leaf node size $\kappa = 5$. The bandwidths h_D and h_Υ and the weight factor α are optimized on a hold-out validation set by grid search (As an indication, this resulted in $h_D = 0.01$m, $h_\Upsilon = 0.03$ and $\alpha = 0.5$). We set the number of the layers of regression cascade as $L = 2$.

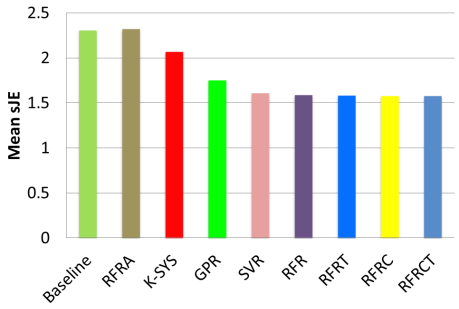
A. Joint-based skeleton correction

To evaluate our algorithm, we show the performance on a challenging data set. This data set contains a large number of poses, $15,815$ frames in total, coming from 81 golf swing motion sequences. Some pose examples are shown in Fig. 3. We select 19 sequences containing $3,720$ frames as the training data set. The rest $12,095$ frames are used for testing.

1) Error Metrics: Given a testing data set $\{(ST_i, GT_i)\}_{i=1}^m$, we obtain the corrected skeletons $\{CT_i; i = 1, \ldots, m\}$. We measure the accuracy of each corrected skeleton $CT = (\mathbf{z}_j; j = 1, \ldots, n)$ by the sum of joint errors (sJE) $GT = (\mathbf{x}_j; j = 1, \ldots, n)$: $\varepsilon = \sum_j \|\mathbf{z}_j - \mathbf{x}_j\|_2$. To quantify the average accuracy on the whole testing data, we report the mean sJE (msJE) across all testing skeletons: $\sum_i \varepsilon_i / m$ (unit: meter).

2) Comparisons: In this section, we give the comparison between the methods for joint-based skeleton correction.

Current Kinect approach. To illustrate the difficulty of the problem, we compare with the approach in the current Kinect system. The current Kinect system for skeleton correction is complex, which employs several strategies such as temporal constraint and filtration. The main idea of the approach is nearest neighbor search. For a testing ST, The approach searches its nearest neighbor in the estimated skeletons in the training set. Then the ground truth of the nearest neighbor is scaled with respect to ST to be the corrected skeleton of ST. The details of this system is unpublished. We refer to the current approach in Kinect system as K-SYS in the rest of paper. On the whole testing set, K-SYS achieves 2.0716 msJE, while RFR achieves 1.5866. We illustrate some examples of the corrected skeletons obtained by K-SYS and our algorithm in Fig. 7. The skeletons obtained by our algorithm are more similar to the ground truths.



	Mean sJE
Baseline	2.3039
RFRA	2.3204
K-SYS	2.0716
GPR	1.7498
SVR	1.6084
RFR	1.5866
RFRT	1.5831
RFRC	1.5766
RFRCT	1.5757

Fig. 6. Comparison with several methods on our testing data set. The quantitative results are illustrated in the left-hand bar graph and the accurate values are listed in the right-hand table. The baseline is the msJE of the input testing estimated skeletons.

Regress the absolute joint position. To show the significance of learning the offsets between joints in ST and GT, we also give the result by directly predicting the absolute joint position by random forest regression (RFRA). To learn the absolute position, for each $GT = (\mathbf{x}_j; j = 1, \ldots, n)$ in the training set $\mathcal{S} = \{(ST_i, GT_i)\}_{i=1}^m$, we translate the C. Hip $\mathbf{x}_1$ to $(0, 0, 0)$ to align them. The absolute joint position of each $GT = (\mathbf{x}_j; j = 1, \ldots, n)$ is obtained by $\tilde{\mathbf{x}}_j = (\mathbf{x}_j - \mathbf{x}_1)/\lambda(GT)$. Then a regression function $\tilde{f} : H(ST) \rightarrow (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n)$ is learned as the process in Sec. IV-C2. Given a testing $ST = (\hat{\mathbf{x}}_j; j = 1, \ldots, n)$, the joint positions of the corrected skeleton $CT = (\mathbf{z}_j; j = 1, \ldots, n)$ are obtained by $CT = \lambda(ST)\tilde{f}(H(ST))$. Finally, the C. Hip $\mathbf{z}_1$ of the obtained CT is translated to $\hat{\mathbf{x}}_1$ to align the CT with the ST. As shown in Fig. 6, RFRA does not perform as well as RFR.

Other regression algorithms. We also apply other regression algorithms to joint-based skeleton correction, such as Gaussian process regressor [21] (GPR) and support vector regressor [22] (SVR). The implementation of GPR was taken from the GPML toolbox [29], which learns the parameters of the mean and covariance functions of Gaussian processes automatically. GPR achieves 1.7498 msJE. Fig. 6(a) shows the apparent advantage of RFR over GPR. We employ the implementation of SVR taken from the package of LIBSVM [30]. We utilize radial basis function (RBF) kernel for SVR and obtain 1.6084 msJE. The result obtained by SVR is also worse than RFR. Besides, to train the model of SVR, quite a few parameters need to be tuned.

We illustrate the performances of all the methods mentioned above in Fig. 6. The baseline is the msJE of the input estimated skeletons, which is 2.3039 msJE. RFRT, RFRC and RFRCT achieve 1.5831, 1.5766 and 1.5757, respectively. The results demonstrate that both performing in the way of cascade and adding temporal constraint can help improve the performance. Different joints are under occlusions of different degrees, e.g. the head is rarely under occlusion and the elbows are

Fig. 7. Examples of the human skeletons. In each example we see the GT, the ST, the CT obtained by NNS and the CT obtained by RFRCT. The CTs obtained by RFRCT are more similar as the GTs.

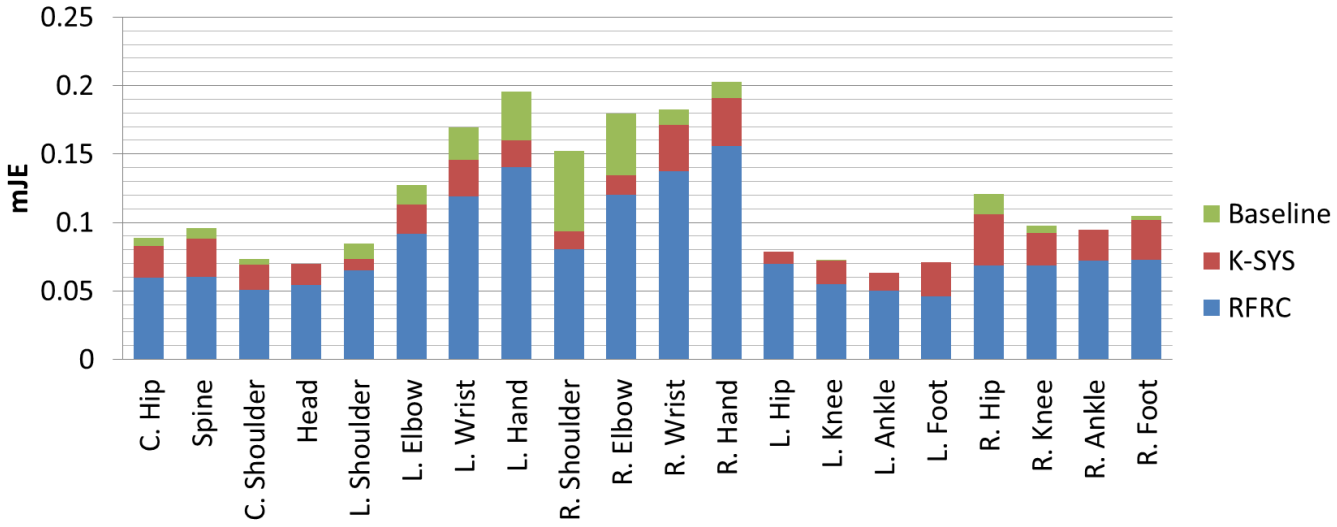


Fig. 8. The error of each joint. Baseline is the mJE of the initial estimated skeletons.

often occluded by the torso. To better illustrate the correction performance, we compute each joint error (JE) and report the mean JE (mJE) of each joint across all testing skeletons in Fig. 8. Baseline is the mJE of the initial estimated skeletons. Compared to the baseline, our approach decreases the errors caused by occlusion significantly. Generally, our approach decreases 44% and 16% error with and without occlusion respectively. Compared to K-SYS, the correction performances of all joints are better.

3) Parameter Discussion: We investigate the effects of several training parameters on regression accuracy. As shown in Fig. 9(a), the mean sJE decreases as the number of trees increases. We also show the performance by varying the leaf node size. The tree depth is controlled by the leaf node size. The smaller is the leaf node size, the deeper is the tree. As shown in Fig. 9(b), the performance of RFR would decrease when setting larger leaf node size. However, encouragingly, RFRC even obtains better result when setting larger leaf node size. The is because setting larger leaf node size reduces the risk of over-fitting of each layer which leads to more performance improvement when using the cascade way.

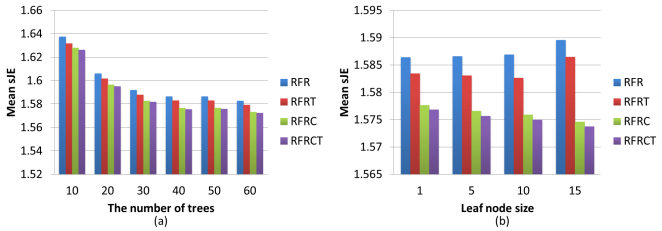


Fig. 9. Training parameters *vs*. performance on joint-based skeleton correction. (a) Number of trees. (b) Leaf node size.

B. Pose tag prediction

To evaluate our method for pose tag prediction, we collect two data sets. One contains 185 golf swing motion sequences, the other contains 104 football throwing motion sequences (mocap data is unnecessary here, so it's convenient to collect more data). A typical motion sequence of football throwing is shown in Fig. 10. Note that, mocap data is not collected for this task, so we illustrate football throwing motions by the initial estimated skeletons. We annotate all the sequences and test our approach on the two data sets separately. For the golf

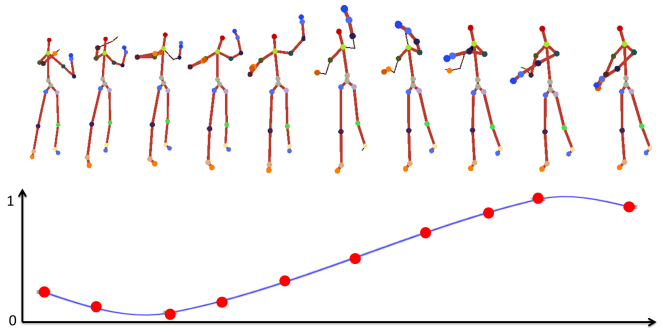
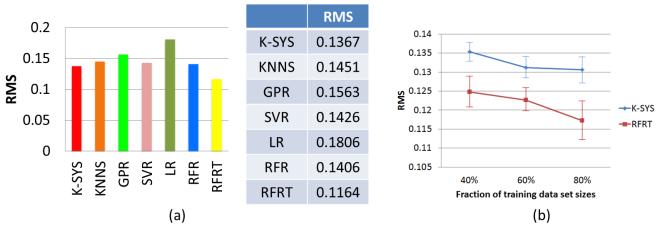
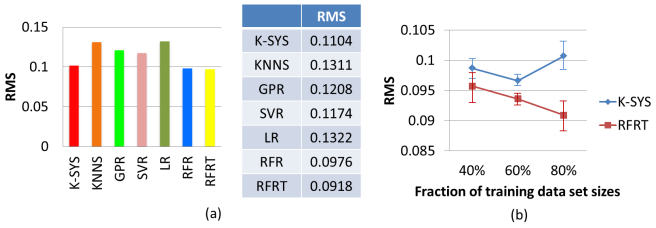


Fig. 10. A typical motion sequence of football throwing.



	RMS
K-SYS	0.1367
KNNS	0.1451
GPR	0.1563
SVR	0.1426
LR	0.1806
RFR	0.1406
RFRT	0.1164

Fig. 11. Comparison with several methods on the golf swing data set. (a) The quantitative results obtained by several methods. (b) Accuracy versus size of training data set.



	RMS
K-SYS	0.1104
KNNS	0.1311
GPR	0.1208
SVR	0.1174
LR	0.1322
RFR	0.0976
RFRT	0.0918

Fig. 12. Comparison with several methods on the football throwing data set. (a) The quantitative results obtained by several methods. (b) Accuracy versus size of training data set.

swing data set, 37 sequences containing $3,066$ frames are used for training; the rest 148 sequences containing $12,846$ frames are used for testing. For the football throwing data set, 21 sequences containing $1,277$ frames are used for training; the rest 83 sequences containing $4,108$ frames are used for testing. In addition, we also did various folds of cross-validation to test the stability of our approach.

1) Error Metrics: Given the testing data set $\{(ST_i, \Upsilon_i)\}_{i=1}^m$, where Υ_i is the annotated tag of ST_i, we obtain the predicted tags $\{\hat{\Upsilon}_i\}_{i=1}^m$. The tag prediction accuracy on the whole testing data set is quantified by the rooted mean square (RMS) distance: $\sqrt{\Sigma_{i=1}^m(\hat{\Upsilon}_i - \Upsilon_i)^2/m}$.

2) Comparisons: In this section, we give the comparison between the methods for tag prediction.

Current Kinect approach. The current approach in Kinect system (K-SYS) for tag prediction also contains many details such as imposing temporal constraint and filtering. The main idea of K-SYS is also nearest neighbor search. For a testing ST, K-SYS searchs its nearest neighbor in training set, then the predicted tag of its nearest neighbor is taken as the tag of the ST. On the football throwing data set, K-SYS achieves 0.1104 RMS, which is worse than RFR (0.0976 RMS). RFRT is the best, which achieves 0.0918 RMS. On the golf swing data set, K-SYS achieves 0.1376 RMS, which is even better than RFR (0.1406 RMS). The reason may be the golf swing data set is more challenging (there are much more occlusions), so the prediction from a single frame is more inaccurate. In this case, the temporal constraint plays a more important role in tag prediction. Our algorithm RFRT significantly improves RFR, it achieves 0.1164 RMS. We also compare with K-SYS by varying the size of training data set. We divide each data set into 10 folds with equal sizes, then randomly select N_t folds for training and use the rest for testing. We compute the mean and the standard deviation of the RMS distances obtained by repeating the random selection for 5 times. The results for $N_t = 4, 6, 8$ using 10 trees on the golf swing data set and the football throwing data set are shown in Fig. 11(b) and Fig. 12(b) respectively.

K-nearest neighbors search. To show the advantage of random forest in data abstraction and robust estimation, we compare with K-nearest neighbors search (KNNS). Given a testing sequence $(ST_1, \ldots, ST_m)$, for each $ST_i (i \in (1, \ldots, m))$, K nearest neighbors are searched from the training set, and the tags of the neighbors are obtained: $\boldsymbol{\Upsilon}_K = (\Upsilon_k; k = 1, \ldots, K)$. Then we obtain the probability distribution $P_{H(ST_i)}(\Upsilon) = \frac{1}{K}\sum_{k=1}^K \exp(-\|\frac{\Upsilon - \Upsilon_k}{h_\Upsilon}\|^2)$. Then considering the temporal consistency, using the method in Sec. IV-D, the optimal value is searched from the distribution $P_{H(ST_i)}(\Upsilon)$ as the predicted tag of ST_i. KNNS only achieves 0.1451 RMS and 0.1311 RMS on the golf swing data set and the football throwing data set, respectively.

Other regression algorithms. We apply GPR and SVR to tag prediction, which achieve 0.1563 RMS and 0.1426 RMS on the golf swing data set respectively and achieve 0.1208 RMS and 0.1174 RMS on the football throwing data set respectively. The tag is a real value ranging from 0.0 to 1.0, so we also apply logistic regression (LR) [23] to tag prediction. However, it only achieves 0.1806 RMS and 0.1322 RMS on our two data sets, respectively. Unlike RFR, which benefits from optimization under the temporal constraint, GPR, SVR and LR have no such advantage.

We illustrate the results of tag prediction on our two data sets in Fig. 11(a) and Fig. 12(a), respectively. Fig. 13 and Fig. 14 shows tag curves of four example sequences from the two data sets, respectively. The horizontal and vertical axes of the tag curve are the frame index of the sequence and the tag value, respectively. The curves obtained by RFRT (black) fit the annotated curves (green) best.

3) Parameter Discussion: The effects of the training parameters, including the number of trees and leaf node size, on tag prediction accuracy is shown in Fig. 15. Generally, using more trees and setting smaller leaf node size help improve the performance.

C. Skeleton correction conditioned on pose tag

Not all the frames in the data set used in Sec. V-A are annotated with tag values, so we select the annotated frames from it which form a subset to evaluate the conditional random

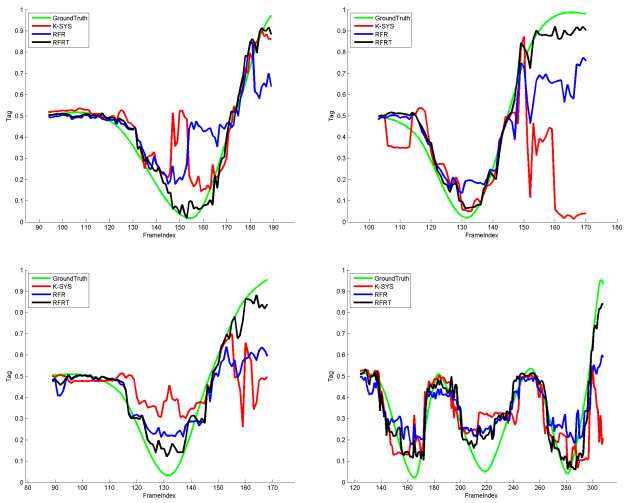


Fig. 13. The tag curves of four example sequences from the golf swing data set. In each example, we show the annotated curve (green) and those obtained by K-SYS (red), RFR (blue) and RFRT (black). The black curves fit the green best.

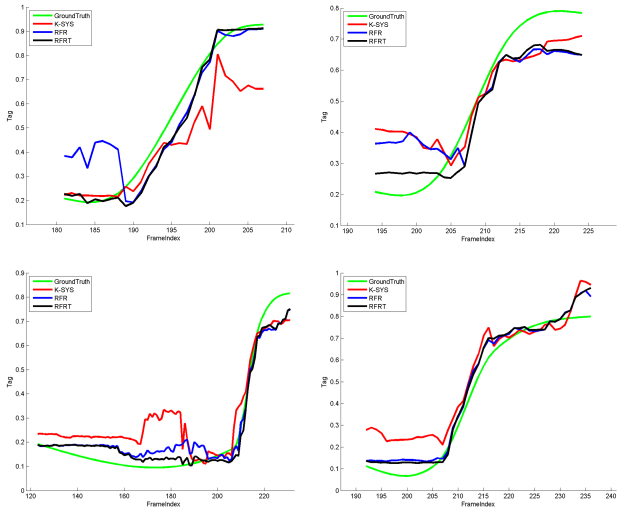


Fig. 14. The tag curves of four example sequences from the football throwing data set. In each example, we show the annotated curve (green) and those obtained by K-SYS (red), RFR (blue) and RFRT (black). The black curves fit the green best.

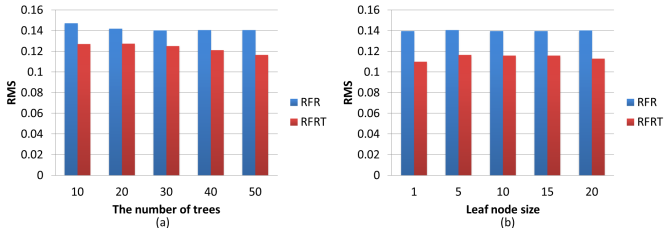
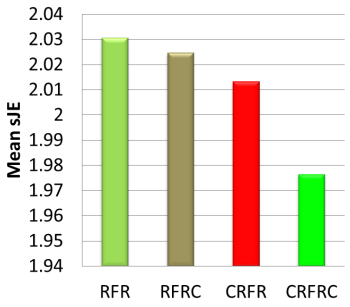


Fig. 15. Training parameters *vs*. performance on tag prediction. (a) Number of trees. (b) Leaf node size.

forest regression method. The subset contains $6,452$ frames in total, from which $2,352$ are randomly selected as the training set and the rest are used as the testing set. To train the conditional random forest, we uniformly partition the tag space



	Mean sJE
Baseline	3.2374
K-SYS	2.6578
RFR	2.0306
RFRC	2.0246
CRFR	2.0132
CRFRC	1.9764

Fig. 16. Comparison with several methods on the subset formed by annotated frames. The quantitative results are illustrated in the left-hand bar graph and the accurate values are listed in the right-hand table. The baseline is the msJE of the input testing estimated skeletons.

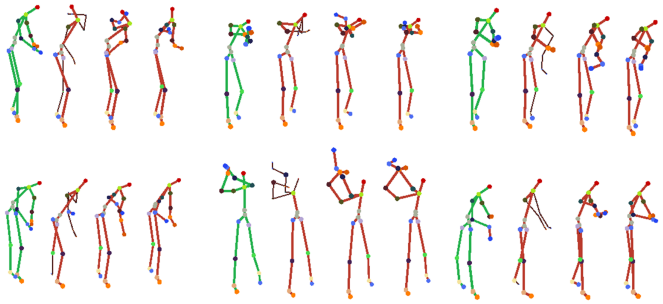


Fig. 17. Examples of the human skeletons. In each example we see the GT, the ST, the CT obtained by RFRC and the CT obtained by CRFRC. CRFRC further corrects the estimated skeletons to approach the ground truth.

into three intervals, i.e. $[0.0\ 1/3]$, $[1/3\ 2/3]$ and $[2/3\ 1.0]$, then split the training set into three according to tags.

1) Comparisons: In this section, we mainly compare the conditional regression method to the standard one. As show in Fig. 16, CRFR indeed enhances the performance of RFR and RFRC, and CRFRC further decreases the error, which achieves 1.9764 msJE. We also illustrate some examples of the corrected skeletons obtained by RFRC and CRFRC in Fig. 17 for comparison. We can see from this figure that CRFRC further corrects the estimated skeletons to approach the ground truth. This is because the incorporation of tags leads to a more accurate data partition, which provides more robust bias prediction. Note that, the testing error of this subset is generally larger than the original whole data set. This is because the action motions of the annotated frames are more drastic than the unannotated ones and severe occlusions usually exist in the annotated frames (see the significant error in the STs in Fig. 17). For clarity, we also list the baseline and the result obtained by K-SYS on this subset in the right-hand table in Fig. 16. Recall that baseline is the msJE of the initial estimated skeletons. The baseline of this subset (3.2374 msJE) is much worse than the one of the original whole data set (2.3039 msJE) also shows the its difficulty. Our methods significantly decrease the joint errors on this subset and achieve much higher performance than K-SYS.

2) Parameter Discussion: There is a new parameter introduced in the proposed conditional regression model: the number of tag intervals (we always partition the tag space uniformly). We vary the number of tag intervals and the results

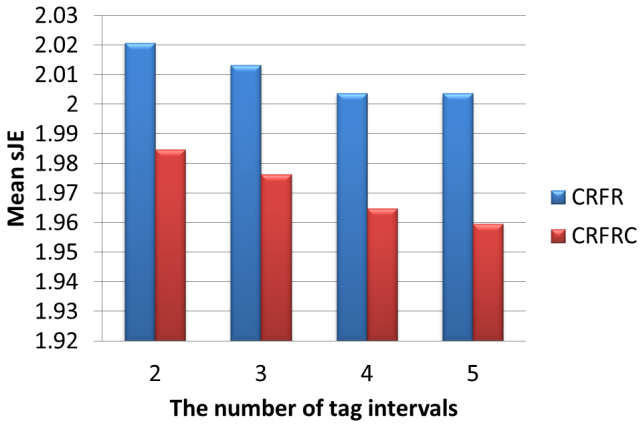


Fig. 18. Joint error versus the number of tag intervals.

TABLE I
THE RUNTIME OF OUR APPROACH ON XBOX. THE RUNTIME IS MEASURED BY MILLISECOND PER FRAME. FOR THE TWO TASKS, JOINT-BASED SKELETON CORRECTION AND TAG PREDICTION, WE REPORT THE RUNTIME OF RFR AND RFRT, RESPECTIVELY.

Number of trees	10	20	30
Skeleton correction (RFR)	6.6ms	11.8ms	17.2ms
Tag prediction (RFRT)	5.9ms	8.8ms	11.8ms

are shown in Fig. 18. Fig. 18 depicts that higher performance will be obtained by partitioning the tag space into more disjoint subsets.

D. Runtime

Our approach is real-time and can be directly embedded into the current Kinect system. We give the runtime of our approach on Xbox in Tab. I.

E. Facial landmark correction

Our regression based correction method is general, it's applicable to correct other estimation/detection errors involved from other sensors. To show this, we apply our method to improve the accuracy of the positions of facial landmarks detected by the method in [31]. Facial landmarks detection is a essential step in face recognition. The method in [31] detects the facial landmarks by applying a generative model of the positions of landmarks combined with a discriminative model of the appearances of landmarks. Generally, when detecting landmarks on a frontal face, the detection error is small; however, when detection is performed on a lateral face, the error is much larger. Therefore, our method, to learn a inhomogeneous bias function to perform correction is also suitable here. Given a face image I, we employ the facial landmarks detector in [31] to obtain the initial positions $\hat{\mathbf{S}}^{(0)}$ of $n = 9$ landmarks, including 4 eye corners, 3 nose corners and 2 mouth corners. To apply our regression based method to correct the errors introduced in landmarks detection, we first compute the SIFT descriptor [32] at each landmark, which lead to a $n \times 128$ dimensional face descriptor $H(I, \hat{\mathbf{S}}^{(0)})$. Then a random forest regression function $f : H(I, \hat{\mathbf{S}}^{(0)}) \rightarrow \mathbf{\Delta}^{(0)}$ is learned as the process in Sec. IV-C2, where $\mathbf{\Delta}^{(0)}$ is the difference between $\hat{\mathbf{S}}^{(0)}$ and the ground truth $\mathbf{S}$. To ensure this difference is scale invariant, we normalized the coordinates of landmarks by the transforms estimated by least square fit according to the default positions of landmarks. The corrected positions of landmarks are obtained by $\hat{\mathbf{S}}^{(1)} = \hat{\mathbf{S}}^{(0)} + f(H(I, \hat{\mathbf{S}}^{(0)}))$. The cascaded way can be also applied as the same as in Sec. IV-C3: $\hat{\mathbf{S}}^{(i+1)} = \hat{\mathbf{S}}^{(0)} + \sum_{k=0}^{i} f(H(I, \hat{\mathbf{S}}^{(0)})), H(I, \hat{\mathbf{S}}^{(k)}))$.

We follow the error metric proposed in [33] to evaluate the accuracy of the positions of landmarks: $\frac{1}{nd_e}\sum_{i=1}^{n}\delta_i$, where n is the number of landmarks, δ_i is the Euclidean point to point error for each individual landmark and d_e is the ground truth distance between the left and right eyes.

We test our method on two publicly data sets, the first one is LFPW [34], which consists of $1,432$ faces from images downloaded from the web using simple text queries on sites such as google.com, flickr.com, and yahoo.com. This data set is quite challenging due to the large variations of poses, lighting conditions, and facial expressions contained in its face images. The face images are taken under wild condition, and have already been divided into a training set and a testing set, which contain $1,132$ and 300 face images respectively. Due to copyright issues, this data set only provides the image URLs. Some URLs are not available, so we only obtain 832 training face images and 231 testing testing face images. We learn the regression function on the training set and apply it to the testing set. In Tab. II, we report the landmarks errors of the correction results obtained by RFRC ($L = 1, 2, 3$) on the testing set of LFPW. The baseline is the landmarks error of the initial detection result obtained by the method in [31]. Tab. II shows that our method indeed improves the accuracy of facial landmark localization, it reduces about 20% localization error. Some selected correction results from LFPW are shown in Fig. 19.

The other is BioID data set [35], which consists of 1521 images of 23 different persons with a larger variety of illumination, background and face size. BioID does not provide training and testing sets, following [34], [36] we directly test the correction function learned on the training set of LFPW on the whole BioID data set. Note that this increases the difficulty of correction, since there are considerable differences in the viewing conditions of these two data sets. BioID does not provide the ground truths for the three nose corners, so we only give the correction results of the other 6 landmarks. As shown in Tab. III, our method RFRC ($L = 1, 2, 3$) can also improve the accuracy of facial landmark localization on BioID significantly even applying the regression function learned on the training set of LFPW, it reduce about 26% localization error. In Fig. 20, we show some selected correction results from BioID [35].

VI. CONCLUSION

We have presented a new algorithm for pose correction and tagging from the initially estimated skeletons from Kinect depth images. Our exemplar-based approach serves a promising direction and we highlighted the importance of learning the inhomogeneous systematic bias. We also emphasize that

TABLE II
THE RESULTS OF LANDMARK DETECTION AND CORRECTION ON LFPW [34]

	Baseline [31]	RFRC, $L = 1$ (RFR)	RFRC, $L = 2$	RFRC, $L = 3$
Landmarks error	0.0695	0.0571	0.0562	0.0556

Fig. 19. Selected results from LFPW [34]. In each image, the green, red and blue points are the ground truth, detection result [31] and correction result, respectively.

TABLE III
THE RESULTS OF LANDMARK DETECTION AND CORRECTION ON BIOID [35]

	Baseline [31]	RFRC, $L = 1$ (RFR)	RFRC, $L = 2$	RFRC, $L = 3$
Landmarks error	0.0796	0.0590	0.0586	0.0583

Fig. 20. Selected results from BioID [35]. In each image, the green, red and blue points are the ground truth, detection result [31] and correction result, respectively.

learning the bias conditioned on a global parameter leads to a more accurate pose correction model. Performing cas-

caded regression and imposing the temporal consistency also improves pose correction. Our experimental results for both pose joints correction and tag prediction show significant improvement over the contemporary systems. Our regression based correction method is general, it's applicable to improve the accuracy of other detection/estimation systems.

Future works may include designing more powerful skeletal features, employing motion analysis techniques for pose correction and recognizing actions based on the corrected poses.

VII. Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61303095 and Grant 61222308, in part by Innovation Program of Shanghai Municipal Education Commission under Grant 14YZ018, and in part by the Ministry of Education of China Project NCET-12-0217.

References

[1] Microsoft Corp. Kinect for XBOX 360. Redmond WA.
[2] L. Liu and L. Shao, "Learning discriminative representations from rgb-d video data," in *Proc. IJCAI*, 2013.
[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 a single depth image," in *Proc. CVPR*, 2011.
[4] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *Proc. ICCV*, 2011.
[5] M. Sun, P. Kohli, and J. Shotton, "Conditional regression forests for human pose estimation," in *Proc. CVPR*, 2012.
[6] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, "Accurate 3d pose estimation from a single depth image," in *Proc. ICCV*, 2011.
[7] A. Baak, M. Müller, 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. ICCV*, 2011.
[8] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *Int'l J. of Comp. Vis.*, vol. 29, no. 1, pp. 5–28, 1998.
[9] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. PAMI*, vol. 25, no. 5, pp. 564–575, 2003.
[10] A. Birnbaum, "A unified theory of estimation," *The Annals of Mathematical Statistics*, vol. 32, no. 1, 1961.
[11] W. Shen, K. Deng, X. Bai, T. Leyvand, B. Guo, and Z. Tu, "Exemplar-based human action pose correction and tagging," in *Proc. CVPR*, 2012.
[12] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *ICCV VS-PETS*, 2005.
[13] L. D. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *Proc. ICCV*, 2009.
[14] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *Proc. ECCV*, 2010.
[15] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Proc. CVPR*, 2005.
[16] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Trans. Cybernetics*, 2013.
[17] S. Tak and H.-S. Ko, "Physically-based motion retargeting filter," *ACM Transactions on Graphics*, vol. 24, no. 1, 2005.
[18] J. Lee and S. Y. Shin, "Motion fairing," in *Proceedings of Computer Animation*, 1996, pp. 136–143.
[19] H. Lou and J. Chai, "Example-based human motion denoising," *IEEE Tran. on Visualization and Computer Graphics*, vol. 16, no. 5, 2010.
[20] R. Duda, P. Hart, and D. Stork, *Pattern Classification and Scene Analysis*. John Wiley and Sons, 2000.
[21] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
[22] B. Schölkopf, A. Smola, R. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, pp. 1207–1245, 2000.
[23] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, "A simulation study of the number of events per variable in logistic regression analysis," *J Clin Epidemiol*, vol. 49, no. 12, pp. 1373–1379, 1996.
[24] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[25] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, 1986.
[26] A. Liaw and M. Wiener, "Classification and regression by randomforest," 2002.
[27] P. Dollár, P. Welinder, and P. Perona, "Cascaded pose regression," in *Proc. CVPR*, 2010.
[28] M. Dantone, J. Gall, G. Fanellli, and L. V. Gool, "Real-time facial feature detection using conditional regression forests," in *Proc. CVPR*, 2012.
[29] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (gpml) toolbox," *Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, 2010, software available at http://www.gaussianprocess.org/gpml.
[30] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[31] M. Everingham, J. Sivic, and A. Zisserman, ""hello! my name is... buffy" - automatic naming of characters in tv video," in *Proc. BMVC*, 2006.
[32] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman, "Sift flow: Dense correspondence across different scenes," in *Proc. ECCV*, 2008, pp. 28–42.
[33] D. Cristinacce and T. Cootes, "Feature detection and tracking with constrained local models," in *Proc. BMVC*, 2006.
[34] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," in *Proc. CVPR*, 2011.
[35] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz, "Robust face detection using the hausdorff distance," in *Conf. on Audio- and Video-Based Biometric Person Authentication*, 2010, pp. 90–95.
[36] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," in *Proc. CVPR*, 2012.

Wei Shen received his B.S. and Ph.D. degree both in Electronics and Information Engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2007 and in 2012. From April 2011 to November 2011, he worked in Microsoft Research Asia as an intern. Now he is a faculty of School of Communication And Information Engineering, Shanghai University. His research interests include computer vision and pattern recognition.

Ke Deng received the M.S. degree from Tsinghua University. He is a senior software development engineer in Microsoft Corporation.

Xiang Bai received the B.S., M.S., and Ph.D. degrees from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2003, 2005, and 2009, respectively, all in electronics and information engineering. He is currently an Associate Professor with the Department of Electronics and Information Engineering, HUST. From January 2006 to May 2007, he was with the Department of Computer Science and Information, Temple University, Philadelphia, PA. From October 2007 to October 2008, he was with the University of California, Los Angeles, as a joint Ph.D. student. His research interests include computer graphics, computer vision, and pattern recognition.

Tommer Leyvand received the M.Sc. degree in Computer Science from Tel-Aviv University. His main interests include Computer Graphics, Computer Vision and Machine Learning. He is a principal development lead for the XBOX NUI Vision R&D team at Microsoft Corporation.

Baining Guo is Deputy Managing Director of Microsoft Research Asia, where he also leads the graphics lab. Prior to joining Microsoft Research in 1999, he was a senior staff researcher with the Microcomputer Research Labs of Intel Corporation in Santa Clara, California. Dr. Guo graduated from Beijing University with B.S. in mathematics. He attended Cornell University from 1986 to 1991 and obtained M.S. in Computer Science and Ph.D. in Applied Mathematics. Dr. Guo is a fellow of IEEE and ACM.

Dr. Guo's research interests include computer graphics, visualization, and natural user interface. He is particularly interested in studying light transmission and reflection in complex, textured materials, with applications to texture and reflectance modeling. He also worked on real-time rendering and geometry modeling. Dr. Guo was on the editorial boards of IEEE Transactions on Visualization and Computer Graphics (2006–2010) and Computer and Graphics (2007 – 2011). He is currently an associate editor of IEEE Computer Graphics and Applications. In 2014 he serves as the papers chair for the ACM SIGGRAPH Asia conference. He has served on the program committees of numerous conferences in graphics and visualization, including ACM SIGGRAPH and IEEE Visualization. Dr. Guo holds over 40 US patents.

Zhuowen Tu received the ME degree from Tsinghua University and the PhD degree from Ohio State University. He is an assistant professor in the Department of Cognitive Science, University of California, San Diego (UCSD). Before joining UCSD, he was an assistant professor at University of California, Los Angeles (UCLA). He was a recipient of the David Marr Prize in 2003 and NSF CAREER award in 2009.