Human Upper Body Posture Recognition and Upper Limbs Motion Parameters Estimation

Jun-Yang Huang[1] Shih-Chung Hsu[1] and Chung-Lin Huang[1,2]

1. *Department Of Electrical Engineering, National Tsing-Hua University, Hsin-Chu, Taiwan*
Email:s9961530@m99.nthu.edu.tw, d9761817@oz.nthu.edu.tw,clhuang@ee.nthu.edu.tw
2. *Department of Applied Informatics and Multimedia, Asia Univeristy, Tai-Chung, Taiwan.*
Email: clhuang@asia.edu.tw

***Abstract*— We propose a real-time human motion capturing system to estimate the upper body motion parameters consisting of the positions of upper limb joints based on the depth images captured by using Kinect. The system consists of the action type classifier and the body part classifiers. For each action type, we have a body part classifier which segment the depth map into 16 different body parts of which the centroids can be linked to represent the human body skeleton. Finally, we exploit the temporal relationship between of each body part to correct the occlusion problem and determine the occluded depth information of the occluded body parts. In the experiments, we show that by using Kinect our system can estimate upper limb motion parameters of a human object in real-time effectively.**

I. INTRODUCTION

A robust real-time human posture recognition system has many applications, such as the man-machine interface, security monitoring and home care. Recently Kinect[8] has been developed for real-time motion capturing. Compared with other high-precision motion capture devices, Kinect-based device provides a more user-friendly interface without putting sensors on the human body.

Most of the motion capture (MC) technologies [6, 7] are based on vision-based pose estimation techniques. Recent advent of instant depth video camera promotes the development of MC technology [2, 9~13]. Vision-based posture estimation can be roughly divided into three categories: (1) model-based [2, 4, 9], (2) example-based [5, 11~15], and (3) label-based methods [1, 31]. The first method uses the prediction-estimation technique to track the limbs of human objects and capture human pose parameters based on the pre-stored human model. The latter exploits K-NN rule to the input data for estimating the human pose parameters. The third method segments a single depth image into a dense probabilistic body part labeling, with the parts defined as spatially localized near skeletal joints of interest.

The major disadvantages of model-based method are that body tracking are time-consuming and error-prone due to error propagation. For human pose estimation, the complexity of high-dimensional search space and large data sets make example-based methods infeasible for real-time applications. In [25, 26], human pose and hand pose estimation uses sophisticated shape matching and NN search, however, its usage is limited to general articulated pose estimation. Belongie *et al.* [27] propose a shape matching method by using so-called shape context applied to example-based technique for the human pose estimation. To overcome the high-dimensional space search problem Local-Sensitive Hashing (*LSH*)[28] and *Parameter Sensitive Hashing* [29, 30] are proposed to retrieve approximate nearest neighbors.

The label-based methods segment the depth image into several 3-D body parts for labeling which is formulated by a decision tree classifier. The centroid of each segmented 3-D body part is a skeleton joint. Shotton *et al.* [1] propose a method to predict 3D positions of body joints from a single depth image. To avoid over-fitting hundreds of thousands of training data, they train a random decision forest classifier. Wang *et al*. [31] propose an upper body motion capturing system using one or more cameras and a color shirt, and then roughly classifying the regions for pose estimation.

To avoid these problems, Shotton *et al*. [1, 18] propose a method to predict 3D positions of body joints from a single depth image. To avoid over-fitting the training data, they train a random decision forest classifier to do the parts segmentation. Random decision tree can be implemented with GPU acceleration [23]. The decision tree is based on the leaflets of depth image of identification, such as architecture complementary fit with some appropriate tracking algorithm [19-22]. The shape context is a reliable feature for recognizing object [3] and the structure of the human body [14].

Different from [1], we propose the action recognition and the human body part segmentation. The former is an action type classifier which is a frame-based classifier and the latter is a human body part classifier which is a pixel-based classifier. Figure 1 shows the flow chart of our system. The depth maps with depth contexts converted to feature vectors are used to train the action type classifier. For different action type, we have a different body part classifier. The advantages are (1) simpler decision tree with few layers, (2) less training time, and (3) the layer body part classifier can be trained separately. Each classifier which consists of several random decision trees is also called a random forest. In the experiments, the sequence of categorized action types follows the temporal dependency, so we may identify the errors and then correct any discrepancy of the recognized action type sequence.

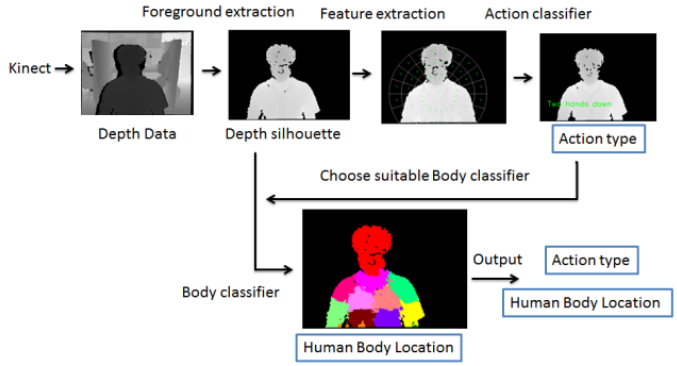

Fig. 1. System flow chart.

II. Training Sample Collection

We need to collect as many training samples as possible from the depth maps with the corresponding labels for action type classifier and body part classifier training. Each input sample depth mape is labeled by a certain action type, whereas each input sample depth pixel will also be labeled by a certain body part.

A. Depth Image Pre-processing

For each depth map image, we can easily apply the image pre-processing techniques to remove the background using the difference of depths and obtain the depth silhouette. Then we apply the recursive flood-filling algorithm to remove the noise of the depth silhouette and background. A clean silhouette will be obtained and the pixels with salient depth different will be removed.

B. Action Type Classifier Training Samples

The object sitting in front of the camera can be roughly categorized into four action types which are regular sitting, left-hand raised, right-hand raised, and both-hand raised. Each action type consists of many different postures as shown in Figure 2. To collect as many postures as possible, we may simulate the action of a virtual avatar by a real human object.

Fig. 2. Four action types: (a) both-hand-raised, (b) normal sitting, (c) left-hand-raised, and (d) right-hand- raised.

To train the action type classifier, we collect the depth information of human silhouette as depth context which can be used to train random tree classifier for action type classification. Depth context is characterized by a 1-D vector with 60 components extracted from the depth map of a human silhouette. Here, we collect 4000 training samples (labeled depth maps) to train the action type classifier.

C. Body Part Classifier Training Samples

The object silhouette image can be divided into 16 body part regions such as: head, neck, left-shoulder, left-upper arm, left-lower arm, left hand, right-shoulder, right-upper arm, right-lower arm, right hand, left-upper chest, left-lower chest, right-upper chest, and right-lower chest as shown in Figure 3.

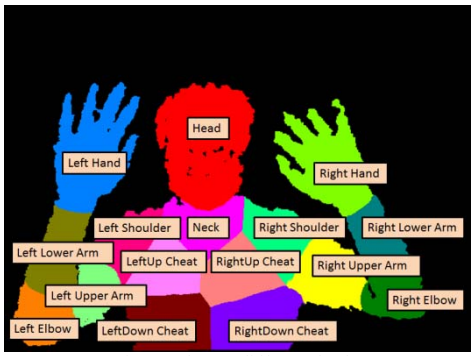

Fig. 3. The 16 different body parts.

To collect the training samples for body part classifier, we use a semi-automatic labeling method to create the training samples. For each depth silhouette, we manually label the centroids of the 16 different regions of the depth map. Then, a segmentation process will segment the silhouette into 16 regions. The training sample generation consists of the following steps:

Step1 : Manually select the centroid of each body part in the silhouette image.

Step2 : Convert the depth map into point cloud in which each 3-D point represent the 2-D pixel with depth information. Similarly, the 2-D centroid with depth is also converted into a 3-D point.

Step3 : For each 3-D point, we find the closest 3-D centroid of certain body part and assign this point to the corresponding body part region.

The training sample generation process which assigns the human silhouette depth map with different body part labels is shown in Figure 4. Each body part is denoted by a certain color, and each 3-D point inside the body part is assigned a specific label with a corresponding color. To train the body part classifiers, we need much more training samples. Here, we generate 500 human silhouette depth maps with different body part labels and then randomly select approximately 200 training samples in each depth map. Totally, there are 100,000 training samples (labeled pixels) collected for training the body part classifiers.

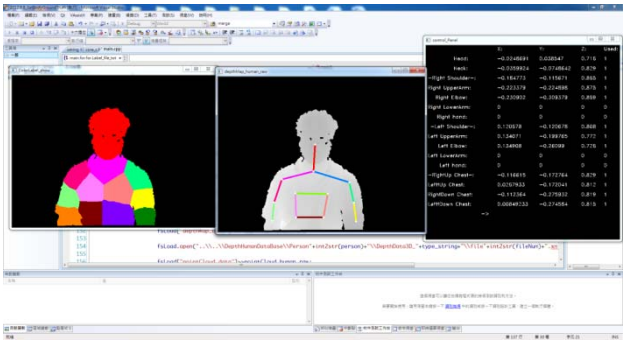

Fig. 4. The user interface for generating the training samples of the body parts.

III. Depth Context

Here, we evenly-divide the depth map into sub-areas and then describe the distribution of the depth counts of the sub-areas. The conventional shape context [28~30] describes the image based on the points of the outline contour points. Similar to shape context, we propose the depth context.

A. Depth Context Extraction

To obtain the depth context, we place circular scope on the depth image which is divided into K lattices. The central part is a circle lattice with radius d. For every distance d, we draw a ring with a total of R rings. Each ring is divided into G grids. Therefore, the depth context contains $K = R \cdot G + 1$ lattices. As shown in Figure 5, we let $G=20$, $R=4$ and segment the circular scope into 81 grids ($20 \cdot 4+1=81$) and define the depth context by a 2-D histogram as $Diagram(r, g)$ with $r=1\sim4$, and $g=1\sim20$. We let $Diagram(0, 0)$ indicates the center portion of the scope. Depth Context is defined by 2-D histogram is polar coordinate as $Diagram(r, g)$, which can be converted to 1-D histogram as bin(k), where $k=1,\ldots K$. The mapping between 2-D and 1-D is $k=1+r \cdot g$, and $k=1$ for $r=g=0$. In our example, we let K=81.

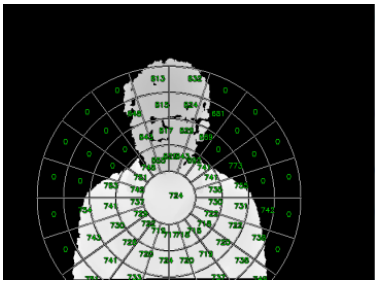

Fig. 5. The Depth Context

Let $\boldsymbol{I}$ denote a silhouette image, $\mathbf{DC}(\boldsymbol{I})$ denote a K-dimension Depth context feature, and dc(k) denote k-th feature value .

$$\mathrm{dc}(k)=\begin{cases} \frac{\sum \mathrm{d_I}(\boldsymbol{x})}{\mathrm{N_k}} : \boldsymbol{x} \in \mathrm{bin(k)} \\ 0 \ , \text{if } \mathrm{N_k} < \frac{\text{Area of bin(k)}}{4} \end{cases} \quad (1)$$

Given a silhouette image $\boldsymbol{I}$ and a pixel $\boldsymbol{x} \in \boldsymbol{I}$, the depth value (in mm) of pixel $\boldsymbol{x}$ is denoted as $d_{\boldsymbol{I}}(\boldsymbol{x})$. N_k is number of pixels in "bin" (k). Figure 4 shows the depth context representing the average depth inside certain lattice. If the number of pixels inside certain lattice is too small, then the average depth indication is not valid, and it is reset to zero.

B. Quantized Depth Context

Because smaller depth variation has little influence on the shape context, we quantize the depth context. The depth of the background is set to zero and the depth of undefined pixel is also set to zero. We define the depth reversely by allowing the depth of the point closer to camera be re-assigned a larger value. The depth of the background pixel and un-identified point is set to zero. The depth of human body farthest from the camera is set to be zero. The depth map in front of the depth zero is quantized for every 10 cm. The depth quantization level is reset as 5 cm. We set the pixel located on the human body with depth zero so that the depth map in front of the body is quantized with a larger value.

Depth context quantization. Given a silhouette image $\boldsymbol{I}$, we denote K-dimension feature by QDC($\boldsymbol{I}$), of which the k^{th} feature value is

$$q_{dc}(k) = \begin{cases} 0 & , \text{if} \quad \mathrm{dc(k)} = 0 \\ 1 & , \text{if } |\mathrm{d_I^{max}} - \mathrm{dc(k)}| < 100 \\ \left[\frac{|\mathrm{d_I^{max}} - \mathrm{dc(k)}| - 100}{50}\right] + 1 & , \text{if } |\mathrm{d_I^{max}} - \mathrm{dc(k)}| \geq 100 \end{cases}$$

where $\mathrm{d_I^{max}}$ is max value of the depth map and the unit of depth in the equation is *mm*.

IV. Action Type Classifier

With the depth context, we may apply the classifier to identify the action type. However, because of the severe non-linearity of the feature space, the conventional classifiers (such as Bayesian classifier or SVM) do not produce good classification results. The shape context which is a 2-D histogram is converted into 1-D feature vector, the spatial characteristics of the shape context is not fully exploited for classification. Therefore, we apply the random forest classifier which is tree-based classification. In each stage of the tree, the split node is a simple classifier which is determined in the training phase.

A. Random Forest Classifier

Random forest consists of multi-stage non-balance binary tree classifiers. Each tree classifier is a multi-stage decision tree as shown in Figure 6. The classification of the input sample is determined by the number of samples in the leaf node of the decision tree. The random forest [16] algorithm is originated from the random decision tree [24]. It is a combination of bootstrap aggregating and random subspace method.

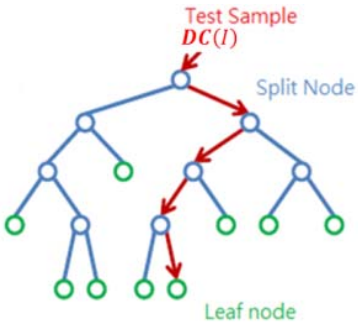

Fig. 6. Decision tree.

Decision tree consists of split nodes and leaf nodes. Each split node consists of a split function. Once an input data is tested by the decision tree, the split function will determine whether the input data belongs to the left or right sub-tree. In each leaf node of binary decision tree, we have the statistical distribution of both classes. Based on the distribution, we may know the posteriori probability of the input data.

After the training phase, we may have an un-balanced binary decision tree. Each input sample will go through the split node of the decision tree. In each split node, there is a hypothesis which will determine the left or the right sub-tree that the input data belongs to. In the leaf node, the class posteriori probability of the input data will be determined. Random forest consists of a set of decision trees obtained in the training process. In the training process, we use the bootstrap and random sample technique [16]. Finally the classification of each input sample is determined by the multiple decision trees.

B. Random Forest Training

Before the training process, we need to collect sufficient number of labeled training samples. Then, we randomly select 80% of the labeled training samples. To create each split node, we need to find a hypothesis so that the input samples of the same class will be separated into the same sub-tree. The split functions f_θ can be randomly generated of which the best one is determined as the hypothesis for the split node. The best split function which separates the training sample into correct sub-tree will be found as the hypothesis of the split node.

For each training sample set Q=$\{\boldsymbol{x}\}$, we find the split function $f_\theta(\boldsymbol{x})$ which separates Q into two sub-sets Q_l and Q_r based on certain feature parameter and threshold as

$$\begin{cases} Q_l(\varphi) = \{\boldsymbol{x} |\, f_\theta(\boldsymbol{x}) > \tau\} \\ Q_r(\varphi) = Q \backslash Q_l(\varphi) \end{cases} \quad (2)$$

where the split node parameter $\varphi = (\theta, \tau)$, it consists of the split function parameters θ, and thresholds τ. In the training process, we define a gain function to measure the effectiveness of the split function based on different split node parameter φ. Here, we define the gain function as follows:

$$G(\varphi) = H(Q) - \sum_{s\in\{l,r\}} \frac{|Q_s(\varphi)|}{|Q|} H\big(Q_s(\varphi)\big) \quad (3)$$

where H(Q) is the Shannon entropy used to calculate entropy of the normalized histogram of type labels L_I for all $\boldsymbol{DC}(I) \in Q$. For each split node parameter, $\varphi = (\theta, \tau)$, we may separate the training sample set and compute the gain function. The larger gain function indicates a better split function. In the training process, the split node parameter φ is randomly generated. The best split node parameter is determined as

$$\varphi^* = \text{argmax}_\varphi\, G(\varphi) \quad (4)$$

The succeeding two split nodes will also be determined using the two separated subsets Q_r and Q_l. The split nodes will be generated iteratively. Under the following two conditions, the two subsets will not be separated further: (1) the number of the samples is not sufficient, and (2) the entropy is too small indicating that the most of the samples in the two separated set are of the same class. Based on the labeled training samples in the leaf node, we may calculate the posteriori of each class C of the t^{th} decision tree as $P_t(C|\boldsymbol{x})$.

C. Action Type Classifier Training

There are four action types: normal, raise left-hand-raised, right-hand-raised, and both-hand-raised. Each action type consists of various postures. In the training process, we collect sufficient number of training samples. Each training sample consists of the depth context and the corresponding action type label. For the training sample set Q, we determine the split function of each split node and separate Q into two subsets Q_l and Q_r. We iteratively separate Q_l or Q_r further into another two sub-sets until certain criteria are satisfied. Then the sample subsets are stored in the leaf node which can be used to determine the posterior $P_t(L|\boldsymbol{DC}(\boldsymbol{I}))$, where t indicate the t^{th} decision tree, and L is class label.

For each split node, given the training sample set Q, we find the split node parameter φ which consists of a split function parameter θ and a threshold τ defined as

$$f_\theta = \begin{cases} dc(k1) - dc(k2)\,, & if\ k1 \neq k2 \\ dc(k1) \qquad\quad , & if\ k1 = k2 \end{cases} \quad (5)$$

where $\theta = \{k1, k2\}$ and split node parameter $\varphi = (\theta, \tau)$. If $(f_\theta > \tau)$, then the sample is separated to the left split node, else to the right split node. There are three parameters for split function k1, k2, and τ. k1 and k2 represents the locations in the depth context. If they are different, then the depth difference is computed, else the depth value is selected. The selected depth difference is compared with the threshold τ to determine whether the left or right node the sample will be classified to.

D. Action Type Classification

Each split node consists of a split function with trained parameter, and each leaf node provides the posterior probability of each class. For each input sample, the tree classifier will decide to which leaf node the sample belongs. The posteriori probability of the input sample will be determined. The split node parameter and the posteriori

probability of the leaf node are determined in the training process as shown in Figure 7. Random forest consists of more than one decision tree. The final outcome is determined by adding the posteriori probability of the decision tress. The final class L^* is determined as

$$L^* = \underset{L}{\operatorname{argmax}} \sum P_t\left(L|\boldsymbol{DC}(I)\right) \quad (6)$$

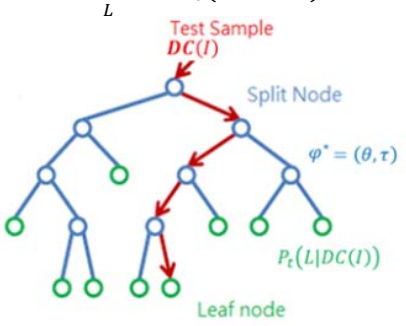

Fig. 7. For each input sample, the t^{th} decision tree will determine its class posteriori probability as $P_t\left(L|\boldsymbol{DC}(I)\right)$.

E. Temporal Correction

For a sequence depth images, there are temporal relationship among the recognized action types. If the current action type is different from the action types of those frames before or after the current frame, then an error occurs. Here, we apply the mean filtering to remove the miss-identified action type. The mean filter finds the majority action type of five consecutive frames. If the recognized action label is different from the majority action type label, then the action type is miss-identified and will be corrected.

V. Body Part Classifier

Body part classifier segments the depth map of the human silhouette into different body part regions. The body part classifier is pixel-based classifier. Each input pixel will be classified to certain body part based on the depth different of its two near neighboring pixels which are also chosen randomly. Here, we need to train the body part classifier, a random forest classifier, which will classify each input sample pixel to one of the sixteen body part regions. The training samples are randomly selected from the labeled body part. For each action type, we have a corresponding body part classifier. The advantages are (1) simpler decision tree with few layers, (2) less training time, and (3) the body part classifiers can be trained separately.

A. Body Part Classifier Training

To train the body part classifier, we randomly collect a lot of patches. Each one has its centroid located at pixel $\boldsymbol{x}$. Based on the depth distribution in each patch we may classify it to different body part. Instead of analyzing the depth distribution, we compare the depth difference of two locations randomly selected inside the patch to determine to which body part it belongs. For each patch with centroid located at $\boldsymbol{x}$, we randomly select the depth of two neighbors of pixel $\boldsymbol{x}\in$ I as the split function parameter $\theta=\{\boldsymbol{u},\boldsymbol{v}\}$.

The training samples are collected by randomly selecting pixel $\boldsymbol{x}$ in the human silhouette depth map $\boldsymbol{I}$. Given the training set Q={$(\boldsymbol{I}, \boldsymbol{x})$}, we find the best split function f_θ with parameter φ as

$$f_\theta = d_I\left(\boldsymbol{x} + \frac{\boldsymbol{u}}{d_I(\boldsymbol{x})}\right) - d_I\left(\boldsymbol{x} + \frac{\boldsymbol{v}}{d_I(\boldsymbol{x})}\right) \quad (7)$$

where $d_I(\boldsymbol{x})$ is the depth at pixel $\boldsymbol{x}$ in image I, and parameter $\theta=\{\boldsymbol{u},\boldsymbol{v}\}$. The normalization of the offsets by $1/d_I(\boldsymbol{x})$ ensures the features are depth invariant. Here, we illustrate the effectiveness of the classifier as shown in Figure 8. For certain proper parameter $\theta=\{\boldsymbol{u}, \boldsymbol{v}\}$, we can find f_θ for different body parts. It is because the depth distributions in different body parts are different.

Fig. 8. Properly selected split parameter $\theta=\{\boldsymbol{u}, \boldsymbol{v}\}$ with different f_θ value can be used to differentiate the class of the patch sample.

In the training process, the randomly generated split node with parameter $\varphi=\{\boldsymbol{\theta}, \tau\}$ may separate the training sample sets into two sib-sets. We select the split node with φ^* that makes the best separation of the training sample set into two sub-sets. The decision tree of the body part classifier generation is similar to the action type classifier. Once the entropy or the number of the training samples divided into two sub-sets is smaller than certain criteria, the split function is found. Based on the training samples in the leaf node, we may have the posteriori probability of the input sample as $P_t(c|I,\boldsymbol{x})$, where t denotes the t^{th} decision tree and c indicates the class of human body part as shown in Figure 9.

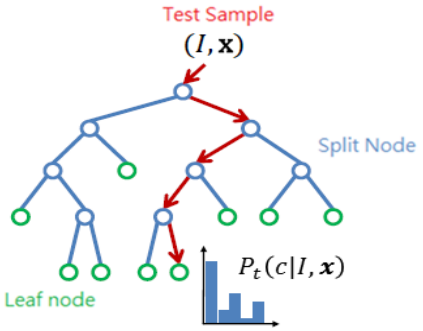

Fig. 9. The input test sample is assigned to a leaf node with class posteriori $P_t(c|I,\boldsymbol{x})$.

B. Body Part Classification

Each input un-labeled sample will be assigned to a certain leave node. The leave node provides the posteriori probability

of the input sample because each leaf node of tree t consists of a learned distribution $P_t(c|I,\boldsymbol{x})$ for body part c.

$$\text{Body type } c^* = max_c\, P_t(c|I,\boldsymbol{x}) \tag{8}$$

For each input sample, the sample distribution of different classes in the corresponding leaf node of decision tree will be used to decide its category. A category decision is made by examining the posteriori of different classes based on the corresponding class samples in the leaf node and then voting. If the number of decision trees is k and the number of training samples n increases to ∞, then all the posteriors provided by the decision trees will converge to the real posterior. If $P(\omega_n \mid \boldsymbol{x})$, where $\omega_n \in \{\text{body part labels}\}$, is the largest, then the Bayes decision selects class ω_n. The decision criterion is based on the outcome of majority of the decision trees ($i > (k-1)/2$) which determine the label of class ω. The class is determined based on the largest probability as

$$C = Argmax_{\omega_m} \sum_{i=(k+1)/2}^{k} \binom{k}{i} P(\omega_m|\boldsymbol{x})^i \left[1 - P(\omega_m|\boldsymbol{x})\right]^{k-i} \tag{9}$$

As the number of decision tree k increases, the largest probability increases, indicating that the uncertainty decreases. The probability of error (miss-classify ω_m) is defined as

$$P(e|\omega_m, x) = \sum_{i=0}^{(k-1)/2} \binom{k}{i} P(\omega_m|\boldsymbol{x})^i \left[1 - P(\omega_m|\boldsymbol{x})\right]^{k-i} \tag{10}$$

However, there exist miss-classified pixels. Here we apply the spatial relationship by using the moving average algorithm to change the pixels. Then, we use the component labeling algorithm to group the pixels of the same class. Finally we may remove the small regions with very few classified pixels. For each body part region, we find it centroid and link the centroid to the other centroid of its neighboring region and create the human body skeleton as shown in Figure 10. To increase the processing speed, we sub-sample the number of pixels.

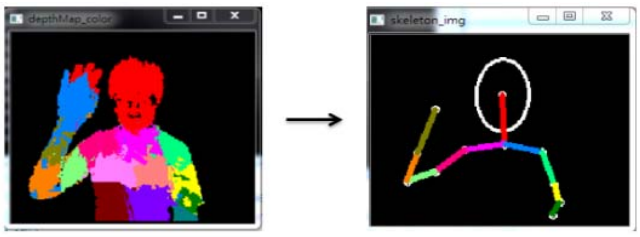

Fig. 10. Connect the centroids of the neighboring regions as the human skeleton.

C. Depth Correction

The occlusion problem often occurs for human articulated motion. For hand-raised posture, the depth map cannot provide the depth information of the chest and neck. The centroids of the chest and neck region are no longer correct and the generated skeleton is no longer valid as shown in Figure 11. For instance, Figures 11(a) and 11(b) illustrate that the neck region changes because the left hand moves in front of the neck and make occlusion. Here, we need to make correction of the depth information of the neck region based on the extrapolation as shown in Figures 11(c) and 11(d). The depth map correction process consists of two steps:

(1) Find the occluded pixels by comparing the depth maps of neck region shown in Figures 11(a-1) and 11(a-2). If the pixel belongs to a new body part (hand) and its depth changes then the pixel is occluded,

(2) Assign a new depth to this original body part pixel as shown in Figure 11(b-2).

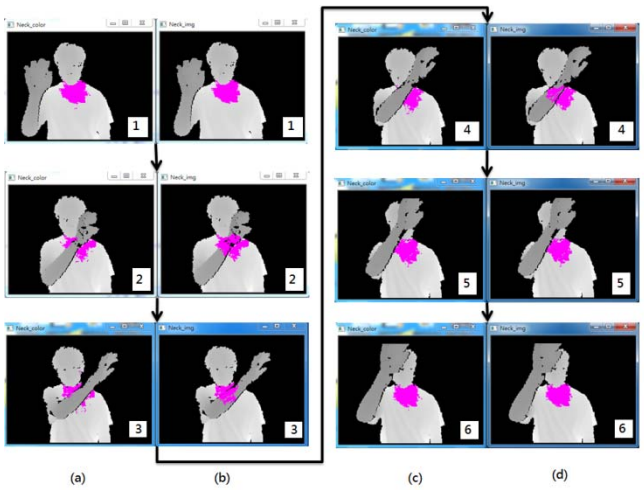

Fig. 11. Correction based on the temporal correlation.

The depth map of the human silhouette $\{\mathrm{d}_\mathrm{I}^\mathrm{t}(x)\}$ is divided into several body part depth maps as $\{\mathrm{b}_\mathrm{c}^\mathrm{t}(\boldsymbol{x})\}$, where c indicates the c^{th} body part to which $\boldsymbol{x}$ belongs, t denotes the t^{th} frame. It can also be described as

$$\{\mathrm{d}_\mathrm{I}^\mathrm{t}(x)\} = \{\mathrm{b}_1^\mathrm{t}(x)\} \cup \{\mathrm{b}_2^\mathrm{t}(x)\} \ldots \{\mathrm{b}_\mathrm{c}^\mathrm{t}(x)\}$$

where $\{\mathrm{b}_\mathrm{c}^\mathrm{t}(x)\} \subset \{\mathrm{d}_\mathrm{I}^\mathrm{t}(x)\}$.

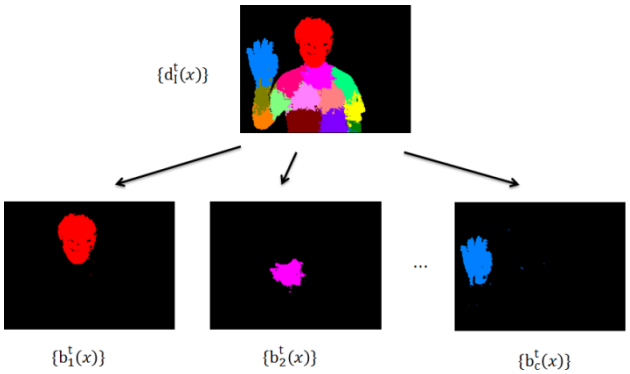

Fig. 12. The silhouette depth map is divided into several body part depth maps.

To recover the occluded depth map $\{\mathrm{b}_\mathrm{c}^\mathrm{t}(x)\}$, we propose the following equation

$$\begin{aligned} &\forall x^*, \mathrm{c}_1^* : \mathrm{b}_{\mathrm{c}_1^*}^{\mathrm{t}-1}(\boldsymbol{x}^*) \neq 0 \\ &\quad \textbf{If } \left(\left(\mathrm{b}_{\mathrm{c}_1^*}^{\mathrm{t}-1}(x^*) - \mathrm{d}_\mathrm{I}^\mathrm{t}(x^*)\right) > \tau\right) \wedge (\mathrm{c}_1^* \neq \mathrm{c}_2^*) \\ &\quad \textbf{Then } \mathrm{b}_{\mathrm{c}_1^*}^\mathrm{t}(x) = \mathrm{b}_{\mathrm{c}_1^*}^\mathrm{t}(x) \cup \mathrm{b}_{\mathrm{c}_1^*}^{\mathrm{t}-1}(x^*) \end{aligned} \tag{11}$$

where $c_2^* = \mathrm{Arg}\{ b_c^{t-1}(\boldsymbol{x}^*) \neq 0 \}$. For every $b_c^{t-1}(\boldsymbol{x})$ in the previous frame, if the depth variation of pixel $\boldsymbol{x}$ is larger than certain threshold τ and the pixel $\boldsymbol{x}$ is assigned to a different body part, then this pixel is occluded and its depth information is incorrect. To recover the depth of $b_c^t(\boldsymbol{x})$, in current frame *t*, we may extrapolate $b_c^t(\boldsymbol{x})$ by using $b_c^{t-1}(\boldsymbol{x})$ in the previous frame. Here we let the threshold τ=50mm.

VI. Experimental Results

The Kinect depth camera is equipped before the screen with a distance of 10 cm and the user operate at a distance of 40~60 cm in front of the screen. To train the action classifier, we collect 4000 depth images as the training samples from 5 different objects of which the resolution is 640×480. The training samples are also used for self-testing. The miss-classified samples are repeated used for training. We continue adding weak training samples until some desired training error have been achieved.

Each training sample receives a weight that determines it probability of being selected for another training of a split node in an individual tree classifier. If a training pattern is accurately classified, then its chance of being used again in the lower-level split node of the tree classifier training is reduced, otherwise, the chance is raised. We focus on the informative or "difficult" training samples which are often miss-classified. Here, we randomly select 80% of the collected training samples to develop three random trees which comprise a random forest classifier. The parameters used to generate the depth context are R=3 and G=24 which is proved to generate the best accuracy.

To train the body part classifier, we collect 500 depth images from 5 different human objects. The training samples are random selected from the depth image. There are 200 training sample selected in each depth image. Totally, there are 100,000 training samples for each body part classifier training. Each body part classifier is a random forest which consists of three trees. The parameter of the split function is θ={$\boldsymbol{u}$, $\boldsymbol{v}$}, where the distance between $\boldsymbol{u}$ and $\boldsymbol{v}$ is less than 100 pixels.

In the experiments, we collect 750 depth images from three videos for testing. We manually label the action types and the centroids of the body parts as the ground truth for action classifier and body part classifier respectively. Here, the recognition accuracy of our action classifiers is 96.53%. The parameters of the depth context are R=3 and G=24. Three random trees are developed and temporal correction is applied.

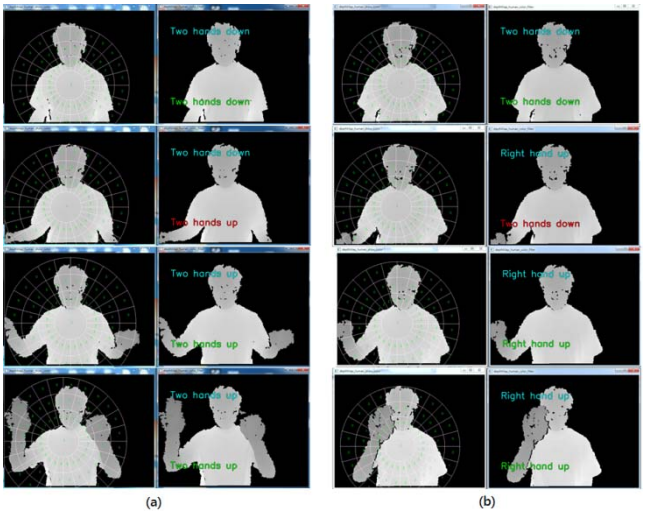

Fig. 13. The action type is miss-identified on the action type transition boundary between (a) normal sitting and both-hand-raised, or (b) normal sitting and left-hand-raised.

After applying the temporal correction, we find that the recognition errors often occur at the action type transition boundary as shown in Figure 13. These kinds of errors will make little influence on the performance of the body part classifier. With *R*=4, the outer boundary of the depth map provides little information but sacrifices the internal depth information and degenerates the recognition accuracy. In the experiments, the accuracy decreases significantly when the number of grid G>40. The small grid will not provide sufficient information for the classifier and degenerate the accuracy. The best selected depth context parameter sets {R=3, G=12} and {R=3, G=24} are used for the random forest action type classifier with three random trees. We also apply the temporal correction so that the action type recognition accuracy reaches 96.5368%.

To train the body part classifier, we collect 750 depth images from previous three videos. We manually set the centroid of every body part. Then we may cluster the depth map image into 14 body part sections. The centroids of all body part regions are identified and linked as the human skeleton. The preciseness of each estimated body part centroid is based on the distance between the estimated centroids of the segmented body part and the real the body part centroid. If it is below 20 pixels then the estimation is correct. Therefore, we define the accuracy of every estimated body part as

$$Accuracy = \#\ of\ precise\ centroids\ /\ total\ \#\ of\ frames$$

The accuracy of every body part is shown in Figure 14.

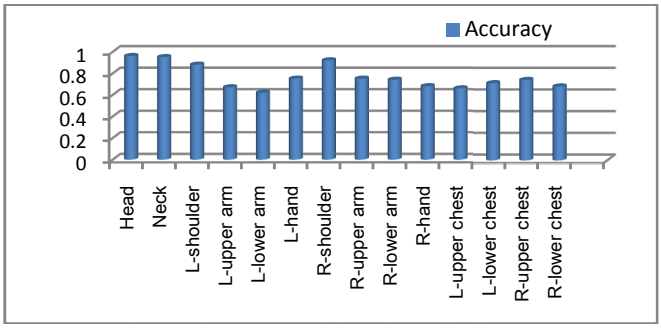

Fig. 14. The accuracy of the estimated body parts.

In the body part limb hierarchical structure, we may have the upper-limb body part such as head and shoulder, and the lower-limb body part such as elbow and hand. Due to the human articulate motion, the movement distance of the upper limb is much smaller than the lower limb. Therefore, the accuracy of the lower limb body part is much larger than the higher limb body part. The experimental results are shown in Figure 15.

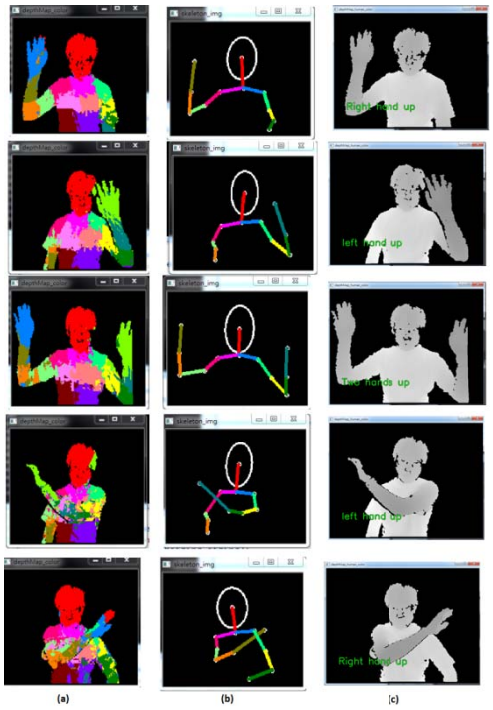

Fig. 15. The body part classifier (a) pixel classification (b) centroid of each body part, and (c) the input depth silhouette of different action types.

The Kinect SDK from Microsoft has the motion capturing capability for close-up human object. The optimal distance between the Kinect and human object is 1 meter for the best system performance. Here, we compare our result with the newest version of Microsoft SDK [32] and show the comparison in Figure 16. At the same distance, we compare the performance of MS SDK (upper row) with ours (lower row). Figure 16 shows the input color images, the results of depth silhouette, and output human skeleton. Figures 16(d) and 16(e) show that our results are better.

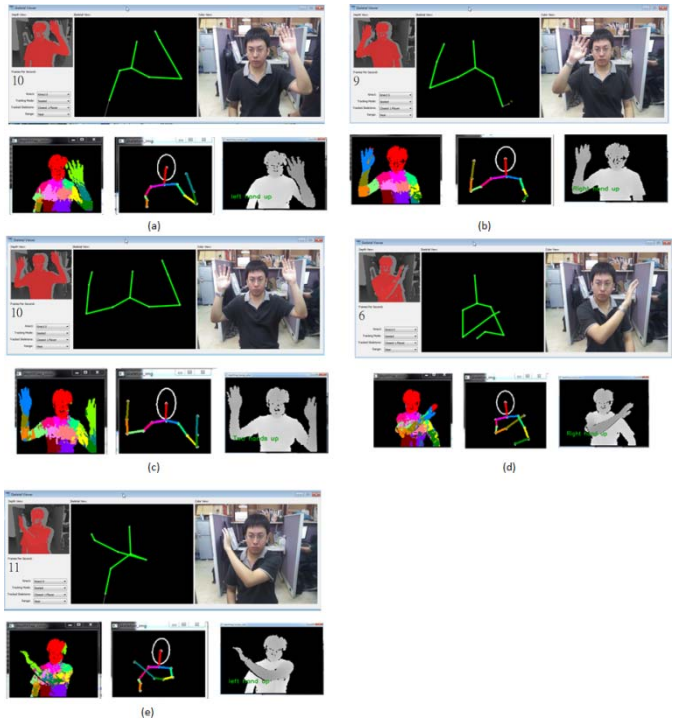

Fig. 16. The comparison between ours and MS SDK.

VII. Conclusions

This paper proposes a real-time upper-body motion capturing system. First, we apply the action type classify to identify the action type of the upper-body part. Based on the different action types, we may apply the part classifier to classify the depth silhouette map into different body regions. The centroids of body parts are connected to illustrate the skeleton of the human object of which the locations are captured as the motion parameters. Here, we apply the random forest on the action type classifier and the body part classifier and design a suitable training samples collection and training algorithm. In the experiments, we demonstrate the effectiveness of our system to track the human upper body part motion

Reference

[1] 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, CVPR 2011

[2] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. Proc. CVPR, 2010.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. PAMI, 24(4):509–522, 2002.

[4] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, H. Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. ICCV 2011.

[5] K. Hara. Real-time Inference of 3D Human Poses by Assembling Local Patches. WACV, 2009.

[6] T. Moeslund, A. Hilton, and V. Kr¨uger. A survey of advances in vision-based human motion capture and analysis. CVIU, 2006.
[7] R. Poppe. Vision-based human motion analysis: An overview. CVIU, 108, 2007.
[8] Microsoft Corp. Redmond WA. Kinect for Xbox 360.
[9] D. Grest, J. Woetzel, and R. Koch. Nonlinear body pose estimation from depth images. Proc. DAGM, 2005
[10] S. Knoop, S. Vacek, and R. Dillmann. Sensor fusion for 3D human body tracking with an articulated 3D body model. In Proc. ICRA, 2006.
[11] Y. Zhu and K. Fujimura. Constrained optimization for human pose estimation from depth sequences. Proc. ACCV, 2007.
[12] M. Siddiqui and G. Medioni. Human pose estimation from a single view point, real-time range sensor. CVCG at CVPR, 2010.
[13] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-time identification and localization of body parts from depth images. Proc. ICRA, 2010.
[14] G. Mori and J. Malik. Estimating human body configurations using shape context matching. Proc. ICCV, 2003
[15] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. Proc. ICCV, 2009.
[16] L. Breiman. Random forests. *Mach. Learning,* 45(1):5–32, 2001.
[17] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. Proc. CVPR, pages 2:775–781, 2005.
[18] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. Proc. CVPR, 2008
[19] R.Wang and J. Popovi´c. Real-time hand-tracking with a color glove. Proc. ACM SIGGRAPH, 2009.
[20] C. Bregler and J. Malik. Tracking people with twists and exponential maps. Proc. CVPR, 1998.
[21] D. Grest, J. Woetzel, and R. Koch. Nonlinear body pose estimation from depth images. Proc. DAGM, 2005.
[22] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard. Tracking looselimbed people. In Proc. CVPR, 2004.
[23] T. Sharp. Implementing decision trees and forests on a GPU. Proc. ECCV, 2008.
[24] Ho, Tin Kam . "Random Decision Forest". Proceedings of the 3rd ICDAR, Montreal, QC, 14–16 August 1995. pp. 278–282.
[25] G. Morio and J. Malik, “Estimating Human Body Configuration using Shape Context Matching,” ECCV, 2002.
[26] V. Athitsos and S. Sclaroff, “Estimating 3D Hand Pose from Cluttered Image,” IEEE Conf. on CVPR, 2003.
[27] S. Belongie, J. Malik, and J. Puzicha, “Shape Matching and Object Recognition Using Shape Contexts,” IEEE Trans. on PAMI. Vol. 24, no.4, 2002.
[28] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. 25th Int. Conf. on Very Large Data Base, 1999.
[29] G. Shakhnarovich, P. Viola, and T. Darrell, “Fast Pose Estimation with Parameter Sensitive Hashing,” Proc. of ICCV 2003.
[30] K. Hara, “Real-time Inference of 3D human Poses by Assembling Local Patches” IEEE Workshop of Applications of Computer Vision 2009.
[31] R. Wang, S. Paris, and J. Popovic, “ Practical Color-based Motion Capture,” Proc. of 21th ACM SIGGRAPH/ Eurographics Symposium on Computer Animation., 2011.
[32] Installing OpenNI, NITE and SensorKinect for Mac OS X, http://developkinect.com /resource/mac-os-x/install-openni-nite-and-sensorkinect-mac-os-x.