Fast Random-Forest-Based Human Pose Estimation Using a Multi-scale and Cascade Approach

Ju Yong Chang and Seung Woo Nam

Since the recent launch of Microsoft Xbox Kinect, research on 3D human pose estimation has attracted a lot of attention in the computer vision community. Kinect shows impressive estimation accuracy and real-time performance on massive graphics processing unit hardware. In this paper, we focus on further reducing the computation complexity of the existing state-of-the-art method to make the real-time 3D human pose estimation functionality applicable to devices with lower computing power. As a result, we propose two simple approaches to speed up the random-forest-based human pose estimation method. In the original algorithm, the random forest classifier is applied to all pixels of the segmented human depth image. We first use a multi-scale approach to reduce the number of such calculations. Second, the complexity of the random forest classification itself is decreased by the proposed cascade approach. Experiment results for real data show that our method is effective and works in real time (30 fps) without any parallelization efforts.

Keywords: Human pose estimation, depth data, random forest, interactive digital contents.

I. Introduction

The reliable and fast estimation of a 3D human body pose is one of the most important problems in the area of computer vision. Its applications include the following fields: interfaces for human-computer interaction, motion capturing in computer graphics, gesture/action recognition in visual surveillance, and healthcare. In particular, it has attracted a lot of attention as a core technology enabling a natural user interface for smart electronic devices and interactive digital contents.

Although this problem has been researched for decades [1]-[17], many difficulties still remain, including the following. Variations of human appearance in images and lighting conditions are countless, and it is therefore nearly impossible to formulate either a generative or discriminative model for intensity images. Partial occlusions and the layering of objects frequently occur in a scene. The dimensionality of the search space is high, which results in existing methods suffering from local minima and ambiguous configurations. In addition, there must be loss of 3D information in the projection process.

Low-cost depth sensors, including the recently launched Microsoft Xbox Kinect [18], make the human pose estimation problem more easily solvable by enabling a simple segmentation of the human body and providing 3D information invariant to human appearances. In a breakthrough paper [7], Shotton and others proposed a robust and efficient method for 3D human pose estimation using Kinect. Their algorithm works well without any initial calibration of the camera or human body, and does not catastrophically fail for a period of hours. Moreover, it achieves both memory and computation efficiency sufficient to be applied to consumer electronic devices. As a result, it provides real-time 3D human pose estimation under the Microsoft Xbox platform. To

Manuscript received Apr. 1, 2013; revised Oct. 27, 2013; accepted Nov. 5, 2013.

This work was supported by the Culture Technology (CT) Research & Development Program of Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA), Rep. of Korea (APP0120130417001).

Ju Yong Chang (phone: +82 42 860 6686, juyong.chang@etri.re.kr) and Seung Woo Nam (swnam@etri.re.kr) are with the Creative Content Research Laboratory, ETRI, Daejeon, Rep. of Korea.

accomplish a fast speed, their method can be parallelized using a graphics processing unit or multi-core CPU.

It is a natural extension to enable a gesture- and motion-based natural user interface to be applicable to smart mobile devices, which have recently become popular. However, because of a limited computational power, directly applying the 3D human pose estimation technique of Microsoft Kinect to a mobile device is not easy. This is therefore the motivation of our work. In this paper, to achieve real-time pose estimation performance (30 fps) on a single-core CPU machine, we propose a novel method for speeding up the existing algorithm without any parallelization effort.

In the state-of-the-art algorithm by Shotton and others [7], a 3D human pose is represented as a set of predefined 3D joints arranged in a tree structure. Their method independently estimates the 3D position of each joint using the following two steps: (1) computing the per-pixel and per-joint probabilities by the random forest classifier [19] and (2) detecting the modes in the probability map of each joint by the mean shift algorithm [20]. The most dominant part of this calculation is the computation of the joint probabilities performed in all pixels. Reducing the complexity of this probability computation is the core of our speed-up approach. For this purpose, we first use a multi-scale approach to densely compute the joint probabilities in only the vicinity of the rough position detected at a smaller scale. Second, we apply a cascade approach to the probability computation by the random forest, resulting in additional gain in the computation time. To be concrete, if the expected probability value is too small, the computation process terminates early without passing all nodes or trees of the random forest.

The remainder of this paper is organized as follows. In the next section, we report other related works. In section III, we describe the human pose estimation algorithm by Shotton and others [7] as a reference method. In section IV, we present our approach to improve the computational efficiency of the reference algorithm. Experiment results are then shown in section V. Finally, conclusions are drawn in section VI.

II. Related Works

A full survey of human pose estimation is out of the scope of this paper. Readers may refer to survey papers for further information [21]-[23]. In this section, we only review works in which the input is assumed to be the range data, such as depth image or 3D points.

In this paper, we roughly classify the human pose estimation methods into three types: (1) generative approach, (2) discriminative approach, and (3) hybrid approach. In generative approaches, as in [4], [5], [13]-[15], a geometric or probabilistic human body model that can explain the observed 3D data is first defined. Human pose estimation is then performed by optimizing the objective function for geometric model fitting or by computing either the maximum likelihood (ML) estimate or the maximum *a posteriori* probability (MAP) estimate in the probabilistic model. In [4], [5], the human body was assumed to be composed of rigid fixed meshes in each body segment or a set of degenerated cylinders with a tree-like hierarchy. These geometric models were fitted into the input depth image using the iterative closest point (ICP) algorithm and subsequent optimization of the cost function, resulting in the output 3D pose. In [15], human pose estimation was formulated as a probabilistic tracking problem including the measurement model inspired by the ICP algorithm and was solved using the unscented Kalman filter. To increase the robustness of the model fitting process, the method in [14] uses the coarse body part labeling result obtained by the optimization of the heuristically defined cost function. The algorithm in [13] also tries robust pose estimation by combining a probabilistic method for detecting the key feature points and a model-based approach for recovering the pose from the detected features. These generative approaches have several limitations. They require the initial calibration for the body model, and the global optimization of the high dimensional nonlinear cost function is almost infeasible. Finally, to avoid the local minima, there should be no fast or abrupt motions.

In discriminative approaches, as in [3], [6], [7], [10], the pose parameters are directly predicted from the input images using the discriminatively learned classifier or regressor or by matching the input depth with the exemplar depth/pose database. In [6], Plagemann and others proposed a method of estimating both the 3D position and the orientation of several body parts (for example, head, hand, and foot). For this purpose, they introduced AGEX (Accumulative Geodesic EXtrema), which are the interest points for the range data, and applied the boosted classifier to these detected points. In [7], using random-forest-based pixel-wise body part recognition and mode detection in the part density, the robust human pose estimation was performed with impressive accuracy in real time. The approach uses a large quantity of synthesized motion capture (mocap) data to train the random forest, and this classifier has been proven to be powerful enough to generate reliable pose estimates without any dependency assumption between body joints. To avoid the arbitrary definition of body parts and effectively handle the occlusions, the method in [3] uses a random forest, not for the body part classification, but for the regression of the joint positions. In [10], Sun and others proposed a conditional regression forest to incorporate the dependency between joints, which improves the estimation accuracy while still maintaining the real-time performance.

Previous generative and discriminative approaches can be combined into hybrid approaches for better robustness and accuracy [1], [2], [9], [11], [12]. In [9], the posterior probability of the generative model was optimized based on the data-driven Markov chain Monte Carlo framework. Unlike the plain generative approaches, this method performs more efficient inference using the proposals from the part detection process. Similarly, the method in [2] exploits the results of the body part detection [6] for the local optimization of the likelihood for robust and fast pose estimation. In [1], the generative optimization algorithm is stabilized by a discriminative component, which is implemented as a database lookup based on the geodesic extrema features. Hypotheses from both the local optimization and database lookup are fused into a robust pose estimate. The method of [12] takes a similar approach, wherein the input depth map is first matched with a motion database by the method based on principal component analysis. The obtained body pose is then further optimized using a non-rigid point registration process. In [11], the ICP algorithm usually adopted in the generative method was replaced with the regression forest that directly estimates correspondences in a single optimization step, achieving more robust and accurate results in real time.

III. Problem Statement

Our goal in this paper is to propose several novel ideas to further speed up the state-of-the-art human pose estimation method by Shotton and others [7]. In this section, we provide an overview of this approach and specify some different implementation details from it.

1. Training Data

In [7], the authors introduced an intermediate body part representation system. According to this, the human figure in a depth map is assumed to be segmented into a number of labeled body part regions. These parts are manually predefined to directly localize the particular skeletal joints of interest or to be combined to predict the position of other joints. Based on this representation, their method first applies the random-forest-based body part recognition algorithm to each pixel of the human region in the input depth image. To learn the classifier, the human depth-map images and their corresponding ground-truth labeled body part regions are necessary as the training data. For this purpose, a massive amount of mocap data for human actions is first captured and is then retargeted onto various kinds of realistic 3D human mesh models. As a result, the 3D mesh models are rendered to synthetically generate the training depth and body part images.

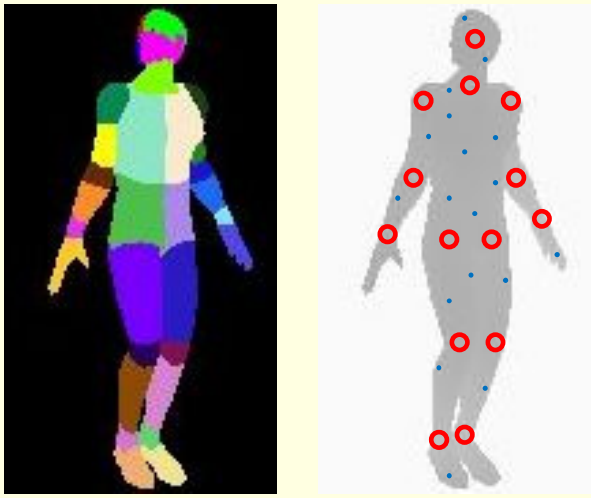

Fig. 1. In [7], human region in depth image is assumed to be segmented into several body part regions as in (a). Training pixels are uniformly sampled from all pixels inside body parts. In our joint detection formulation, we use pixels in vicinity of projected joint points (red circles) and randomly sampled non-joint pixels (blue points), as shown in (b).

In this paper, to avoid an arbitrary definition of body parts, we formulate human pose estimation as a *joint detection* problem. Under this formulation, instead of all pixels belonging to the body parts, pixels in the vicinity of the 2D projection points of the 3D skeletal joints are exploited as the training data, as illustrated in Fig. 1. We consider a total of fourteen skeletal joint classes: head, neck, left and right (L/R) shoulders, L/R elbows, L/R wrists, L/R hips, L/R knees, and L/R ankles. Additionally, another class (non-joint class) that does not belong to any other skeletal joints is defined to increase the discriminability in multi-class joint classification and to be exploited for the cascade approach in the next section. The training data for this non-joint class is obtained by randomly sampling pixels in the human region, excluding the training pixels for the predefined skeletal joints.

2. Random Forest Classification

Random forest [19] has been proven to be easy to implement, simple to train, fast to test, and effective in various computer vision problems [24]-[26] as a multi-class classification method. It is an ensemble classifier that consists of T randomized decision trees. This method combines bootstrap aggregating (bagging) [27] and random feature selection [28], [29] to reduce the correlation between trees and make it more robust against noise.

Let us assume that depth image I and pixel $\mathbf{x}$ are given to the

decision tree. We start from the root node and traverse a path for each node until reaching the leaf node by repeatedly evaluating the following weak classification rule:

$$h(\mathbf{x};\theta,\tau)=\left[f(\mathbf{x};\theta)\geq\tau\right], \tag{1}$$

where $[\cdot]$ is the indicator function, τ denotes the threshold parameter, and

$$f(\mathbf{x};\theta)=I\left(\mathbf{x}+\frac{\mathbf{u}}{I(\mathbf{x})}\right)-I\left(\mathbf{x}+\frac{\mathbf{v}}{I(\mathbf{x})}\right) \tag{2}$$

is a simple depth comparison feature. Here, $\theta=(\mathbf{u},\mathbf{v})$ is the feature parameter that consists of two offset vectors. At the leaf node of tree t, we obtain the discrete probability distribution $P_t(c\mid I,\mathbf{x})$ over joint class c, and these distributions for all trees in the forest are combined to provide the final distribution for classification:

$$P(c\mid I,\mathbf{x})=\frac{1}{T}\sum_{t=1}^{T}P_t(c\mid I,\mathbf{x}). \tag{3}$$

Figure 2 illustrates how the specific decision tree produces the output distribution from the input depth map and pixel coordinate.

The algorithm used for training the random forest is described as follows. For each tree, new training images are selected by sampling from the training images uniformly with replacement. We use this new training data to grow the tree until it meets some stopping criteria by recursively applying the following procedure.

Step 1. A set of node split parameters, $\phi=(\theta,\tau)$, (that is, feature parameter θ and threshold parameter τ) is randomly generated.

Step 2. For each proposed split parameter, ϕ, the training examples, $Q=\{(I,\mathbf{x})\}$, are divided into the following left and right subsets:

$$\begin{cases}Q_L(\phi)=\{(I,\mathbf{x})\mid h(\mathbf{x};\phi)=0\}\\ Q_R(\phi)=\{(I,\mathbf{x})\mid h(\mathbf{x};\phi)=1\}.\end{cases} \tag{4}$$

Step 3. For each pair of subsets corresponding to the split parameter, we compute the information gain as follows:

$$G(\phi)=H(Q)-\frac{|Q_L(\phi)|}{|Q|}H(Q_L(\phi))-\frac{|Q_R(\phi)|}{|Q|}H(Q_R(\phi)), \tag{5}$$

where $H(Q)=-\sum_{c\in C}P_Q(c)\ln P_Q(c)$ denotes the Shannon entropy for normalized histogram P_Q of Q.

Step 4. Select the node split parameter with the maximum information gain,

$$\phi^{*}=\arg\max_{\phi}G(\phi), \tag{6}$$

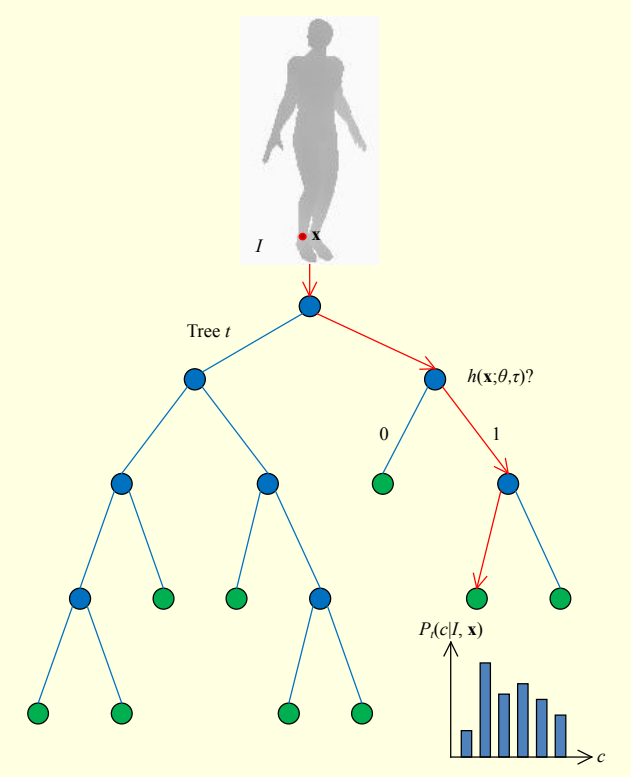

Fig. 2. For input depth map and pixel coordinate, tree is traversed from root node to specific leaf node, producing output probability distribution.

and assign it to the current node as the optimal split parameter.

Step 5. If the optimal information gain, ϕ^{*}, is smaller than a threshold or the depth of the current node in the tree reaches the maximum value, stop the algorithm and return the tree. Otherwise, recursively repeat Steps 1 through 4 for the left and right nodes, respectively.

3. Mean Shift Mode Detection

Under the assumption of the calibrated depth camera, we can compute the position of the reprojected 3D point corresponding to each pixel in the depth image. The random forest classifier explained in the previous subsection is then applied to all pixels in the human region of the input depth image, producing 3D points with probability distributions over all joint classes. The core idea is that from these 3D points with probabilities, the densest position (that is, the mode point) for each joint class is determined as the 3D joint estimate.

To define the probability density for each joint class, we use the following kernel density estimation:

$$f_c(\hat{\mathbf{x}})\propto\sum_{i=1}^{N}P(c\mid I,\mathbf{x}_i)\cdot k\left(\left\|\frac{\hat{\mathbf{x}}-\hat{\mathbf{x}}_i}{b_c}\right\|^2\right), \tag{7}$$

where $\hat{\mathbf{x}}$ is a 3D world coordinate, $\hat{\mathbf{x}}_i$ is the 3D coordinate

http://dx.doi.org/10.4218/etrij.13.2013.0063

corresponding to the pixel coordinate $\mathbf{x}_i$, N is the number of pixels in the target human region, $k(\cdot)$ is the kernel function, and b_c denotes the kernel bandwidth defined for each joint class. As the kernel function, we use the simple Epanechnikov kernel:

$$k(x)=\begin{cases}1-x, & 0\le x\le 1,\\ 0, & \text{otherwise.}\end{cases} \tag{8}$$

The density estimation in (7) can be thought to be the Parzen density estimate with per-joint probability $P(c\,|\,I,\mathbf{x}_i)$ in (3) as the weight coefficient.

Now, the mode of the per-joint density obtained through (7) is found using the mean shift algorithm [20], which is a well-known, local, and iterative mode seeking method. The mean shift algorithm is basically a gradient ascent algorithm, and its simplest implementation is evolving the trajectories, starting from all data points, and following the gradient direction until convergence. In the case of a mean shift, this gradient vector is easily expressed by computing the weighted mean in the window centered on the current point. Finally, the mode points are obtained by merging the end points of the converging trajectories.

IV. Proposed Method

1. Multi-scale Approach

The original pose estimation method first applies the random forest classifier to all pixels belonging to the human region in the input depth image. If the number of pixels and the maximum depth of the decision tree are denoted by N and D, respectively, the computational complexity of this step is $O(NTD)$. As a result of the first step, the probability density corresponding to each joint class is obtained, and this is used for the subsequent mean shift algorithm. In the second step, we apply the mean shift procedures to all human region pixels, which results in $O(N^2M)$ complexity, where M denotes the number of iterations.

To reduce the overall computation complexity, we first adopt a multi-scale approach. The basic idea is to consider the pixels in the vicinity of the roughly estimated point more densely rather than to equally consider all pixels. For this purpose, we first construct multi-scale depth images $\{I^s\}_{s=0}^{S}$. We then apply the original pose estimation algorithm to the lowest scale image, I^0. In a concrete form, the random forest classifier computes the probability distribution $P(c\,|\,I^0,\mathbf{x}^0)$ for all pixels $\{\mathbf{x}_i^0\}_{i=1}^{N_0}$, where N_0 denotes the number of pixels inside image I^0. Then, the mode point, $\mathbf{m}^0$, is obtained by the mean shift algorithm. The converged mode point is illustrated in Fig. 3(b). We then use the detected mode as the starting point of the mean shift algorithm in the next scale and obtain the refined mode, as shown in Fig. 3(d). This mode refining process at scale s can be formulated by the following update equation:

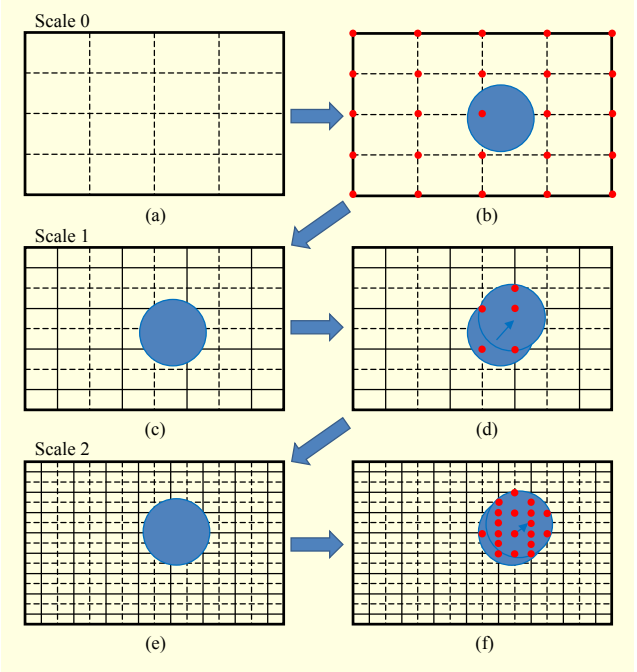

Fig. 3. Based on proposed multi-scale approach, random forest and mean shift computations are densely performed only in neighborhood of roughly estimated point. Note that additional computations of random forest classification occur at red points at each scale.

$$\mathbf{m}_{k+1}^s=\mathbf{m}_k^s+\mathbf{v}^s\left(\mathbf{m}_k^s\right), \tag{9}$$

where $\mathbf{m}_k^s$ denotes the mode point at the k-th iteration and $\mathbf{v}^s$ is the mean shift vector defined by

$$\mathbf{v}^s(\hat{\mathbf{x}})=\frac{\sum_{i=1}^{Ns}P(c\,|\,I^s,\mathbf{x}_i^s)\,g\left(\left\|\frac{\hat{\mathbf{x}}-\hat{\mathbf{x}}_i^s}{b_c}\right\|^2\right)\hat{\mathbf{x}}_i^s}{\sum_{i=1}^{Ns}P(c\,|\,I^s,\mathbf{x}_i^s)\,g\left(\left\|\frac{\hat{\mathbf{x}}-\hat{\mathbf{x}}_i^s}{b_c}\right\|^2\right)}-\hat{\mathbf{x}}. \tag{10}$$

The initial model point $\mathbf{m}_0^s$ is set to be $\mathbf{m}^{s-1}$, the converged mode point at the previous scale. Note that $g(x)$ becomes the uniform function for the Epanechnikov kernel in (8) according to $g(x) = -k'(x)$. Therefore, the computation of the probability, $P(c\,|\,I^s,\mathbf{x}_i^s)$, by the random forest classifier need not be performed for all pixels. It can be done for the points inside the uniform kernel during the mean shift update on demand. After convergence, we can obtain $\mathbf{m}^s$, the mode point at scale s. These procedures are repeated until we arrive at the original image, I^s, in the last scale to generate the final mode point, $\mathbf{m}^s$, as shown in Fig. 3(f).

Using this multi-scale approach, the number of random

forest computations can be significantly reduced. In the next two subsections, we explain how to make the random forest computation itself much faster.

2. Node Cascade Approach

In the pose refinement step, the role of the random forest computation is to calculate the probability value corresponding to the particular skeletal joint for each pixel in the input image. The probability value is obtained by traversing each decision tree and adding the probability value stored in the leaf node for all trees, as in (3). This probability represents the contribution of each 3D point for a per-joint probability density. The basic idea of the cascade approach is to ignore the points with a low contribution. For such low contribution points, we do not need to fully traverse the decision tree, and the random forest computation can be terminated before arriving at the leaf node.

To implement the proposed idea, we recursively define the auxiliary probability vector, $\mathbf{P}_t^n$, for each node n in decision tree t as follows:

$$\mathbf{P}_t^n = \begin{cases} P_t & \text{if } n \text{ is a leaf node,} \\ \max\left(\mathbf{P}_t^l, \mathbf{P}_t^r\right) & \text{otherwise,} \end{cases} \tag{11}$$

where P_t is the probability distribution stored at the leaf node in the original random forest classifier, $\max(\cdot,\cdot)$ represents the element-wise maximum operator, and $\mathbf{P}_t^l$ and $\mathbf{P}_t^r$ denote the auxiliary probability vectors of the two children nodes of node n. Let us assume that during the random forest computation for input depth I and pixel $\mathbf{x}$, we reach node n and $\mathbf{P}_t^n(c) = \rho$. This means that whatever the path chosen from current node n, probability $P_t(c \mid I, \mathbf{x})$ for joint class c does not exceed ρ. Therefore, if ρ is smaller than the predefined small threshold, ρ_{node}, to reduce the computational complexity, our cascade algorithm returns a zero probability without further proceeding to traverse the tree. We call this strategy a node cascade approach because the condition checking for early termination is carried out at the node level. This node cascade approach is illustrated in Fig. 4.

3. Tree Cascade Approach

In this subsection, we extend the previous cascade idea to the tree level. The basic idea is to define a new classifier in charge of checking the early termination similarly to the auxiliary node and place it in front of the random forest. To reduce the additional burden to the overall computational complexity, this classifier should have a simple structure and be fast to test. For this reason, we again use the randomized decision tree, called the rejection tree, which classifies the input pixel as being either"joint" or "non-joint." We can learn this tree with the same training data for the random forest in section III.1 by combining all skeletal joint classes into one joint class. Let $P_0(c \mid I, \mathbf{x})$ denote the output probability of our single rejection tree. Our tree cascade approach continues the successive computation of random forest only if $P_0(c = '\mathit{joint}' \mid I, \mathbf{x})$ is larger than the small threshold, ρ_{tree}. For many pixels apart from the skeletal joint points, this probability should be small enough to ignore these pixels for further processing. This tree cascade approach is shown in Fig. 5.

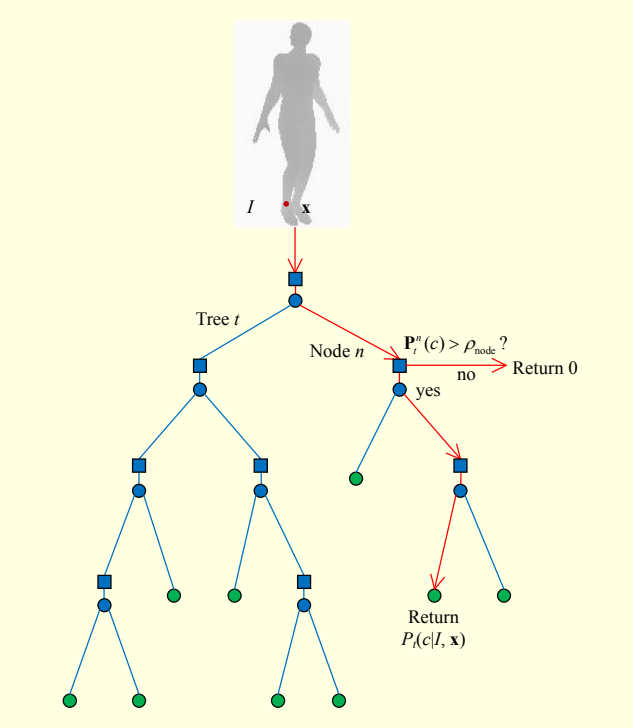

Fig. 4. In our node cascade approach, auxiliary node (blue square) with probability vector $\mathbf{P}_t^n$ is added to each non-leaf node (blue circle). In this auxiliary node, we test whether maximum attainable probability is larger than probability threshold ρ_{node}.

Note that there are other ways of defining the rejection tree. For example, we can construct several decision trees, each of which classifies the input pixel into the particular joint class against all other classes. This specific joint classifier should be faster to pre-test than the single universal joint classifier. However, this scheme requires many rejection trees with the same number of skeletal joints, resulting in too heavy a memory burden.

4. Numerical Analysis

Let us take a close look at the complexity analysis of the proposed method. Random forest computations and mean shift procedures are performed at all grid points of the image at the

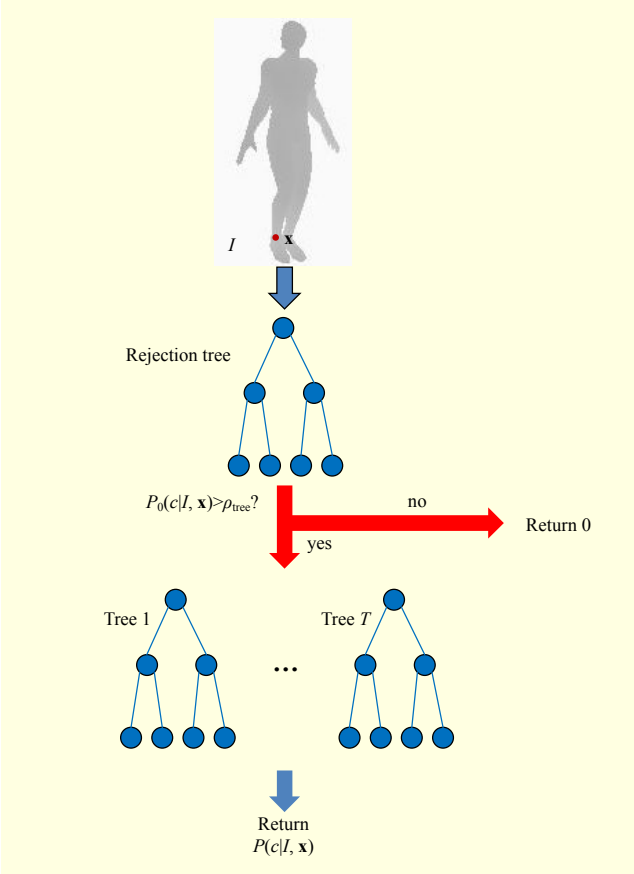

Fig. 5. In our tree cascade approach, input pixel is tested on rejection tree before it is fed to random forest. This rejection tree computes probability that input pixel belongs to any joint class and quickly determines whether to discard the current pixel for further processing by comparing probability with threshold ρ_{tree}.

lowest scale. This results in $O\left(N_s TD + N_s^2 M\right)$ complexity, where N_s denotes the number of points in the lowest scale image. From the next scale, the random forest must be computed only for the points inside the kernel of the mean shift. In addition, this mean shift iteration can only be applied to the detected mode point rather than all points. Because the algorithm is already converged at the lowest scale, the length of the trajectory and the number of iterations in the mean shift refinement stage should be small. Therefore, the additional complexity can be written as $O\left(N_k TD + N_k\right)$, where N_k denotes the number of points inside the kernel window in the original image. Note that N_s exponentially decreases as the multi-scale level increases. Therefore, the overall complexity is dominated by the mean shift refinement process, $O\left(N_k TD + N_k\right)$.

Now, let us focus on the random forest computations in the mean shift refinement process and consider how much our cascade approaches reduce the complexity, $O\left(N_k TD\right)$. First, using the node cascade approach, for some pixels inside the mean shift kernel, the random forest computation stops before it reaches the leaf node. This results in the same effect in that depth D of the decision tree becomes D', which is less than D. Therefore, the reduced complexity can be denoted by $O\left(N_k TD'\right)$. Second, using the tree cascade approach, the number of pixels fed to the random forest classifier is decreased to N_k'. However, before that, the rejection tree should be applied to all pixels inside the mean shift kernel. Therefore, the total complexity can be written as $O\left(N_k D_r + N_k' TD'\right)$, where D_r denotes the maximum depth of the rejection tree. We empirically study the computational complexity of the cascade approach. For this reason, the number of weak classifications in (1) is investigated for our learned human pose estimation algorithm. In the case of the original method without a cascade approach, the average number per image is $O(N_k TD)$ = 141001. When the node cascade approach is exploited, this decreases to $O(N_k TD')$ = 82075. In addition, both the node and tree cascade approaches further reduce it to $O(N_k D_r + N_k' TD')$ = 4636 + 64660, proving the effectiveness of our cascade approaches.

V. Experiment Results

1. Datasets

To construct the dataset, we first use Kinect to capture several depth videos in which the actors perform different gestures. For this purpose, Microsoft Kinect for Windows SDK is used, which enables us to utilize the skeletal tracking functionality. This provides the 3D position estimates of the fourteen skeletal joints defined in section III.1. However, this of course contains incorrect results, which can be classified into three classes: (1) a successful case with acceptable errors, (2) a failed case with completely erroneous results, and (3) a failed case with severe occlusions. To generate a refined dataset, we develop a data post-processing tool, with which the second and third types of cases are manually modified. In the second type of case, the position of an incorrect estimate is interactively changed by a human user. In addition, the results from the third type of case are discarded. The processed dataset serves as the training data for the random forest classifier and as the ground truth data for an evaluation of the proposed method. The final dataset for our experiments consists of 10,000 depth images of 320×240 pixels and their corresponding skeletal joint data. From these images, two-fifths of them are selected to construct the test data. Note that actors in the training data are not the same as the ones in the test data, so the generalization accuracy can be measured. Figure 6 shows several example images from

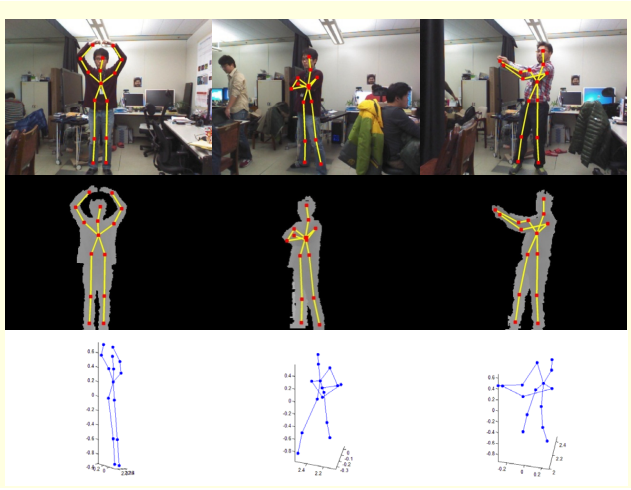

Fig. 6. Example images in our dataset. First row shows color images with their corresponding overlaid skeletons. Second row shows depth images with their corresponding overlaid skeletons. Third row shows skeleton images rendered from various viewpoints.

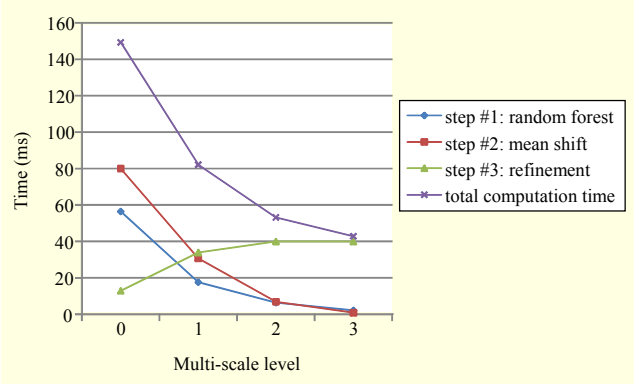

Fig. 7. Computation time (ms) per frame over varying multi-scale levels.

the dataset.

2. Implementation Details

Before training, all dataset images are pre-processed to contain only segmented human regions using a standard background subtraction technique. To construct the training example pixels, the fourteen skeletal joint pixels and ten randomly sampled non-joint pixels are first identified for each training image. We take 25 pixels in the vicinity of each identified pixel, resulting in a total of 600 training example pixels per image. These training examples are used to train both the random forest classifier and the rejection tree, as described in subsection IV.3. The forest consists of $T = 3$ trees, of which the maximum depth is set as $D = 20$. In the case of the rejection tree, we use $D_r = 10$ as the maximum depth. To optimize each node in the tree, we generate candidate feature θ 1,000 times and candidate threshold τ 100 times per feature. From these candidates, the optimal node split parameter is found in a brute force manner. We use the same kernel bandwidth parameter, $b_c = 0.1$ m, for all joint classes in the mean shift mode detection step. All experiments are performed on a standard PC with a 3.20 GHz Intel CPU and 32 GB of RAM. The proposed algorithm is implemented as a MATLAB code, with the core functions written in C. We should emphasize that the parallel computing functionalities are not used in our implementation.

3. Performance Analysis

In this paper, we analyze the performance of the proposed method in two aspects: computation time and prediction accuracy. To empirically study the computational complexity, we divide our pose estimation process into three steps: (1) computation of the random forest, (2) computation of the mean shift at the smallest scale, and (3) pose refinement at the largest scale. The computing times of the three steps are separately measured in milliseconds (ms). To measure the joint-prediction accuracy, we adopt the mean average precision (mAP) over all joints. This can be obtained by calculating the average precision per joint and then computing their average. Herein, we set the true positive distance threshold to d_{tp}= 60 mm. We conduct the experiments of the pose estimation testing by sequentially applying three parts of the proposed method: (1) multi-scale approach, (2) node cascade approach, and (3) tree cascade approach.

Figure 7 and Table 1 show the results with varying multi-scale level parameters. The zero multi-scale level represents the original method by Shotton and others [7] without our proposed speed-up techniques. Note that the computation time (149 ms/frame) of the original method is different from the result (50 fps = 20 ms/frame on an 8-core CPU) in [7] because the parallel computing functionality is not exploited in our experiments. We observe that the total computation time is dramatically decreased as the multi-scale level is increased. This is because the complexity of Steps 1 and 2 is heavily reduced. On the other hand, Step 3 becomes dominant, as expected from the analysis in subsection IV.4. Our goal is to reduce the computing time with little sacrifice in accuracy. Therefore, we select multi-scale level 2, resulting in 53 ms/frame and 0.7618 mAP, which is almost 99 percent of the accuracy of the original method.

To test the node cascade approach, we fix the multi-scale level to 2, and conduct the experiments under varying node cascade thresholds. The experiment results of this node cascade approach are illustrated in Fig. 8 and Table 2. The results show that under a wide range of parameters (0.1 to 0.9), the proposed method can reduce the computation time by about 10 ms,

Table 1. Computation time (ms) per frame and mAP over varying multi-scale level. Last column shows ratio of mAP to mAP without our speed-up technique.

Multi-scale level	Resolution	Step #1: random forest	Step #2: mean shift	Step #3: refinement	Total computation time (per frame)	mAP	mAP (%)
0	(320×240)	56.472	80.047	12.85	149.369	0.7707	100%
1	(160×120)	17.633	30.683	33.898	82.214	0.7697	100%
2	(80×60)	6.435	6.847	39.914	53.196	0.7618	99%
3	(40×30)	2.131	0.823	39.93	42.884	0.6925	90%

Table 2. Computation time (ms) per frame and mAP over varying node cascade thresholds. Multi-scale level is fixed by 2. Last column shows ratio of mAP to mAP without our speed-up technique.

Node cascade thresholds	Step #1: random forest	Step #2: mean shift	Step #3: refinement	Total computation time (per frame)	mAP	mAP (%)
0	6.435	6.847	39.914	53.196	0.7618	99%
0.1	5.843	6.411	27.042	39.296	0.7471	97%
0.2	5.702	6.251	26.869	38.822	0.7476	97%
0.3	5.675	6.096	26.888	38.659	0.7480	97%
0.4	5.68	5.927	26.953	38.56	0.7473	97%
0.5	5.639	5.87	26.807	38.316	0.7474	97%
0.6	5.537	5.352	26.722	37.611	0.7453	97%
0.7	5.487	5.081	26.564	37.132	0.7456	97%
0.8	5.4	4.871	26.438	36.709	0.7460	97%
0.9	5.377	4.695	26.198	36.27	0.7430	96%
1	4.93	3.933	21.382	30.245	0.6657	86%

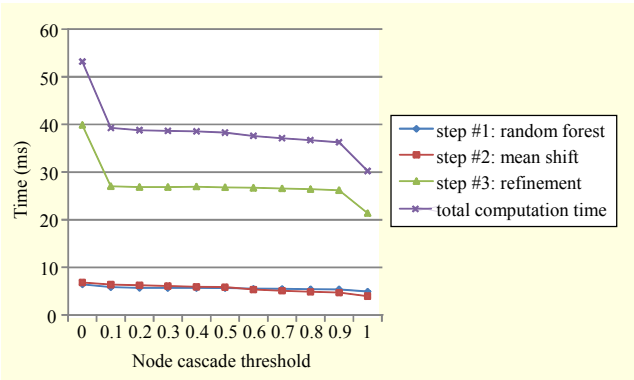

Fig. 8. Computation time (ms) per frame over varying node cascade thresholds. Multi-scale level is fixed at 2.

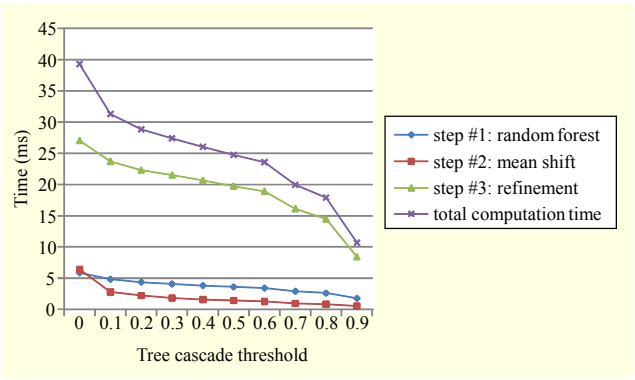

Fig. 9. Computation time (ms) per frame over varying tree cascade thresholds. Multi-scale level and node cascade threshold are fixed by 2 and 0.1, respectively.

simultaneously maintaining the accuracy of the original method. Note the high robustness of our algorithm against the node cascade thresholds. This is because the joint classes are almost separated from each other at the leaf node of the trained randomized tree, which means that the probabilities are close to either zero or one. We set the node cascade threshold to ρ_{node}=0.1, which results in 39 ms/frame and 0.7471 mAP, which is about 97% of the original accuracy.

In the tree cascade test, the multi-scale level and node cascade threshold are fixed to 2 and 0.1, respectively. The results under varying tree cascade thresholds are presented in

Table 3. Computation time (ms) per frame over varying tree cascade thresholds. Multi-scale level and node cascade threshold are fixed by 2 and 0.1, respectively. Last column shows ratio of mAP to mAP without our speed-up technique.

Tree cascade thresholds	Step #1: random forest	Step #2: mean shift	Step #3: refinement	Total computation time (per frame)	mAP	mAP (%)
0	5.843	6.411	27.042	39.296	0.7471	97%
0.1	4.835	2.768	23.703	31.306	0.7420	96%
0.2	4.339	2.206	22.302	28.847	0.7391	96%
0.3	4.082	1.796	21.522	27.4	0.7355	95%
0.4	3.805	1.547	20.684	26.036	0.7297	95%
0.5	3.62	1.404	19.73	24.754	0.7140	93%
0.6	3.414	1.246	18.909	23.569	0.6998	91%
0.7	2.911	0.943	16.116	19.97	0.6192	80%
0.8	2.62	0.801	14.496	17.917	0.5674	74%
0.9	1.77	0.509	8.407	10.686	0.3071	40%

Fig. 9 and Table 3. The proposed method works very well. As a bigger threshold is used, more points far away from the skeletal joints are ignored for the subsequent processing, considerably reducing the computation time. However, compared to the node cascade approach, the degradation of the prediction accuracy is relatively large. We find that a reasonable performance of 29 ms/frame and 0.7391 mAP can be achieved by setting ρ_{tree}=0.2. This is about 96% of the original accuracy, proving that the proposed method works in real time with competent accuracy compared to the original random-forest-based human pose estimation method.

VI. Conclusion

In this paper, we presented several approaches to speed up the existing random-forest-based 3D human pose estimation method. Our approaches are based on two conceptually simple ideas: (1) the multi-scale approach and (2) the cascade approach. In the experiments with a real dataset, our proposed method was proven to be effective in reducing the computational complexity and to run in real time (30 fps) on a single-core CPU machine. This technique should be helpful in applying a real-time pose estimation capability to machines with low computing power, such as smart mobile devices. Our future work will employ a bottom-up agglomerative clustering method instead of the relatively time-consuming mean shift algorithm as a mode detection approach to achieve further gain in computational complexity.

References

[1] A. Baak et al., "A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera," *Int. Conf. Comput. Vis.*, 2011, pp. 1092-1099.

[2] V. Ganapathi, "Real Time Motion Capture Using a Single Time-of-Flight Camera," *Comput. Vis. Pattern Recog.*, 2010, pp. 755-762.

[3] R. Girshick et al., "Efficient Regression of General-Activity Human Poses from Depth Images," *Int. Conf. Comput. Vis.*, 2011, pp. 415-422.

[4] D. Grest, J. Woetzel, and R. Koch, "Nonlinear Body Pose Estimation from Depth Images," *DAGM*, 2005, pp. 285-292.

[5] S. Knoop, S. Vacek, and R. Dillamnn, "Sensor Fusion for 3D Human Body Tracking with an Articulated 3D Body Model," *IEEE Int. Conf. Robotics Autom.*, May 2006, pp. 1686-1691.

[6] C. Plagemann et al., "Real-Time Identification and Localization of Body Parts from Depth Images," *IEEE Int. Conf. Robotics Autom.*, May 2010, pp. 3108-3113.

[7] J. Shotton et al., "Real-Time Human Pose Recognition in Parts from Single Depth Images," *Comput. Vis. Pattern Recog.*, June 2011, pp. 1297-1304.

[8] J. Shotton et al., "Efficient Human Pose Estimation from Single Depth Images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, 2012, pp. 2821-2840.

[9] M. Siddiqui and G. Medioni, "Human Pose Estimation from a Single View Point, Real-Time Range Sensor," *CVCG CVPR*, San Francisco, CA, USA, June 2010.

[10] M. Sun, P. Kohli, and J. Shotton, "Conditional Regression Forests for Human Pose Estimation," *Comput. Vis. Pattern Recog.*, 2012, pp. 3394-3401.

[11] J. Taylor et al., "The Vitruvian Manifold: Inferring Dense Correspondences for One-Shot Human Pose Estimation," *Comput. Vis. Pattern Recog.*, 2012, pp. 103-110.

[12] M. Ye and X. Wang, "Accurate 3D Pose Estimation from a

Single Depth Image," *Int. Conf. Comput. Vis.*, 2011, pp. 731-738.
[13] Y. Zhu, B. Dariush, and K. Fujimura, "Controlled Human Pose Estimation from Depth Image Streams," *Comput. Vis. Pattern Recog.*, Anchorage, AK, USA, June 2008.
[14] Y. Zhu and K. Fujimura, "Constrained Optimization for Human Pose Estimation from Depth Sequences," *Asian Conf. Comput. Vis.*, Nov. 2007, pp. 408-418.
[15] J. Ziegler, K. Nickel, and R. Stiefelhagen, "Tracking of the Articulated Upper Body on Multi-view Stereo Image Sequences," *Comput. Vis. Pattern Recog.*, 2006, pp. 774-781.
[16] A. Hernández-Vela et al., "Graph Cuts Optimization for Multi-limb Human Segmentation in Depth Maps," *Comput. Vis. Pattern Recog.*, June 2012, pp. 726-732.
[17] D. Jo et al., "Tracking and Interaction Based on Hybrid Sensing for Virtual Environments," *ETRI J.*, vol. 35, no. 2, Apr. 2013, pp. 356-359.
[18] Microsoft Corp, "Kinect for XBOX 360," Redmond, WA, USA.
[19] L. Breiman, "Random Forests," *Mach. Learning*, vol. 45, 2001, pp. 5-32.
[20] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, 2002, pp. 603-619.
[21] S. Escalera, "Human Behavior Analysis from Depth Maps," *AMDO*, *LNCS*, vol. 7378, 2012, pp. 282-292.
[22] T.B. Moeslund, A. Hilton, and V. Krüger, "A Survey of Advances in Vision-Based Human Motion Capture and Analysis," *Comput. Vis. Image Understanding*, vol. 104, 2006, pp. 90-126.
[23] R. Poppe, "Vision-Based Human Motion Analysis: An Overview," *Comput. Vis. Image Understanding*, vol. 108, 2007, pp. 4-18.
[24] V. Lepetit, P. Lagger, and P. Fua, "Randomized Trees for Real-Time Keypoint Recognition," *Comput. Vis. Pattern Recog.*, 2005, pp. 775-781.
[25] F. Moosmann, B. Triggs, and F. Jurie, "Fast Discriminative Visual Codebooks Using Randomized Clustering Forests," *Neural Inf. Process. Syst.*, 2006, pp. 985-992.
[26] J. Shotton, M. Johnson, and R. Cipolla, "Semantic Texton Forests for Image Categorization and Segmentation," *Comput. Vis. Pattern Recog.*, Anchorage, AK, USA, June 2008.
[27] L. Breiman, "Bagging Predictors," *Mach. Learning*, vol. 24, 1996, pp. 123-140.
[28] Y. Amit and D. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Comput.*, vol. 9, 1997, pp. 1545-1588.
[29] T.K. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, 1998, pp. 832-844.

Ju Yong Chang received his B.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, Rep. of Korea, in 2001 and 2008, respectively. From 2008 to 2009, he was a postdoctoral researcher at Mitsubishi Electric Research Lab. (MERL), Cambridge, MA, USA, working on the vision-based robot bin picking system. He was with the Samsung Electronics Co. Ltd. at Suwon in the Rep. of Korea as a senior engineer from 2009 to 2011, where he worked on developing 3D image processing algorithms for 3D TV. In April 2011, he joined the School of Electrical Engineering and Computer Science at Seoul National University as a contract professor. From March 2012, he has been with the Creative Content Research Laboratory at ETRI, Daejeon, Rep. of Korea, as a senior researcher. His current research interest is solving various computer vision problems, such as human pose estimation and action recognition.

Seung Woo Nam received his B.S. and M.S. degrees in electronics engineering from Kyungpook National University, Daegu, Rep. of Korea, in 1996 and 1998, respectively. He is a senior researcher with the Creative Content Research Laboratory at ETRI, Daejeon, Rep. of Korea. His research interests are 3D video processing, human-computer interaction, and computer graphics.