Random Tree Walk toward Instantaneous 3D Human Pose Estimation

Ho Yub Jung[1] Soochahn Lee[2] Yong Seok Heo[3] Il Dong Yun[4]

Div. of Comp. & Elect. Sys. Eng.,
Hankuk U. of Foreign Studies
Yongin, Korea, 449-791
jung.ho.yub@gmail.com[1]
yun@hufs.ac.kr[4]

Dept. of Elect. Eng.
Soonchunghyang U.
Asan-si, Korea, 336-745
sclsch@sch.ac.kr[2]

Dept. of Elect. & Comp. Eng.
Ajou University
Suwon, Korea, 336-745
ysheo@ajou.ac.kr[3]

Abstract

The availability of accurate depth cameras have made real-time human pose estimation possible; however, there are still demands for faster algorithms on low power processors. This paper introduces 1000 frames per second pose estimation method on a single core CPU. A large computation gain is achieved by random walk sub-sampling. Instead of training trees for pixel-wise classification, a regression tree is trained to estimate the probability distribution to the direction toward the particular joint, relative to the current position. At test time, the direction for the random walk is randomly chosen from a set of representative directions. The new position is found by a constant step toward the direction, and the distribution for next direction is found at the new position. The continual random walk through 3D space will eventually produce an expectation of step positions, which we estimate as the joint position. A regression tree is built separately for each joint. The number of random walk steps can be assigned for each joint so that the computation time is consistent regardless of the size of body segmentation. The experiments show that even with large computation gain, the accuracy is higher or comparable to the state-of-the-art pose estimation methods.

1. Introduction

The introduction of accurate depth cameras has made human pose estimation a much more feasible problem, facilitating the development of many advanced methods. Among these are methods using generative models, where a pose is estimated by minimizing the distance between a human model and the depth map [14, 29, 9, 10, 27, 28]. This minimization often requires a complex optimization and thus these methods rely heavily on temporal constraints by setting the initial pose as the one from the previous frame. Moreover, these tracking approaches often additionally require an accurate human model beforehand.

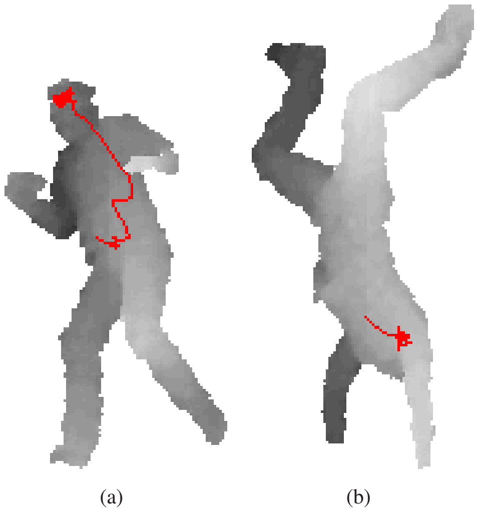


Figure 1. The red lines represents the random tree walks trained to find the head position. The random walk starts from the body center in (a). In (b), the head position is found with fewer steps by starting from the chest, which is much closer than the body center.

In contrast, a discriminative approach aims to directly train the conditional probability of the body part labels or positions. This enables pose estimation even from a single depth image, without any initialization from previous frame and accurate body models [19, 13, 20, 22]. The methods based on a randomized decision and regression forest have shown to be effective and efficient, being able to operate in real-time and on commercial products [16].

Despite their efficiency, there is still demand for even more improvement. The implementation of Shotton et al. relies on parallel computation using either a GPU or a multi-core CPU [19]. These resources are not easily available in current embedded platform computing devices such as smart phones, TVs, and tablets. Mobile devices further require minimal computation to maximize battery life. Also, more importantly, a pose estimation algorithm has to operate simultaneously with multiple applications and games which requires larger portion of computational resources [17]. These trends all support demand for a much more efficient algorithm.

In this paper, we introduce a simple yet powerful discriminative method for pose estimation from a single depth image which can operate over 1000 frames per second (fps) on a single core 3.20 GHz CPU with no additional use of GPU or SIMD operations. Considering the omission of parallel computing, the proposed method is over 100 times more efficient than previous methods.

As with the methods of [19, 13, 20, 5, 21], the proposed method is also based on randomized trees. In previous methods, the joint position is estimated by aggregating pixel-wise tree evaluations. Since the body part sizes are all different, an excessive number of tree evaluations are often made for a larger body part. Similar to supervised descent approach [24], we learn to estimate the relative direction to the joint. Then, at test stage, an initial starting point is moved towards the joint position by random walk in the direction estimated from the trained randomized regression trees. We term this process as *random tree walks* (RTW). We note that, the specific vector that guides the walk is randomly selected among a set of representative vectors in each leaf node of the random tree. By reconstructing the joint position estimation using random walks, we minimize the number of required samples. Fig. 1 shows an example of the proposed RTW process to estimate the head position. We can see the path of the walk as the regression tree guides the step direction at each point.

Using the RTW, large computational gain is possible compared to the previous methods [19, 13, 20] because the regression tree is iteratively evaluated per joint, rather than per pixel. Furthermore, the kinematic tree of joints can be leveraged by performing RTWs sequentially, and initializing the subsequent random walk's starting point as the estimated position of a preceding joint. We construct an optimal sequence of joints based on the kinematic tree. The comparison between Fig. 1 (a) and (b) demonstrates the effects of different starting points.

By adjusting the step size in RTWs, the required iterations for each joint can be controlled to balance the tradeoff between accuracy, stability, and efficiency. In our experiments, sufficient accuracy and stability is achieved with only 64 steps. By fixing the number of steps, the computation time is stable regardless of the depth map size or occlusions in joints. With 15 joints overall in the model, the total number of RTW steps is under one thousand. In contrast, the methods of [19] and [13], that also apply decision trees, attempt to evaluate all pixels in the human region using multiple trees.

This paper is organized as follows. We first provide a review of related works in Sec. 2. Then the proposed method using RTW is presented in Sec. 3, where details of constructing the training set, training the regression trees and the pose estimation algorithm are described. Experimental evaluation is presented in Sec. 4, including comparison with the state-of-the-art methods as well as effects of changing number of random walk steps and step sizes. The paper is concluded in Sec. 5.

2. Related Work

The marker-less human pose estimation problem is defined by various input data and conditions. The most challenging is pose estimation from a single color image. Methods that deal with this problem include poselets [3] and the mixture-of-parts model [26]. Pose estimation from video sequences employs additional information such as background subtraction or body part segmentations. The body segmented image is used as features, and hashing technique is used to find examples with similar features and poses [18]. The human silhouette is an efficient descriptor for 3D pose recovery [1]. The segmentation also enables smaller pose search space for body parts [7]. By increasing the view points, more accurate human extraction is possible, and the under-constraint issue can be eased [11, 6]. If efficient stereo matching is possible from multi-cameras, poses are estimated from the depth images [25].

Compared to human segmentation on color images, depth images provide much more invariant information for estimation of pose. Thus, methods using active depth cameras have been quite successful. The common strategy is to use a discriminative model for initialization and a generative model for tracking. The iterated closest point (ICP) algorithm, a greedy method for minimizing the distance between model and depth map, is the basic framework for tracking [14, 23, 15]. The recent tracking algorithm of [10] can operate in 125 fps with sub-sampling. Simultaneous human shape estimation and tracking algorithms were also introduced [8, 28].

For the problem of model construction and pose from a single depth image, following discriminative approaches have been effective. In the work of Shotton et al. [19], multiple decision trees are traversed for each pixel to find the body part labels of the pixel. Once the pixels are classified into body parts, the possible joint positions are found with multiple mean-shifts. The pixel-wise decision forest method was furthered extended to regression forests. Instead of categorizing the pixels, the positions of joints are directly estimated from the regression forests by learning the joint offsets from the pixel positions [13, 20]. Ye et al. initialized the skeletal frame by alignment and database look up, and the final pose was refined by minimizing least least-squares distance [27]. Similarly, Baak et al. estimated the pose from a single frame by nearest neighbour learning on extrema points [2].

Since methods for a single image can be applied to each frame, these methods can be easily applied to video. Parallel implementations of these methods have achieved real-time performance [19, 13] using GPU. A majority of computation resources are committed to pixel-wise body part classification, in order to obtain sufficient number of classified pixels for stable mean-shift localization. Longer CPU time is spent in localizing bigger body parts such as torso. Although smaller body parts like elbows are important to pose estimation, a smaller computation resource is allotted. The offset regression forest can infer hidden or smaller body parts from other larger body part pixels [13]. However, the pixel-wise tree traverse on larger body part is still unavoidable.

Overall, discriminative methods have an advantage over the generative tracking methods which require a fairly comprehensive human model along with accurate pose estimation in the previous frame. However, compared to the state-of-art body tracking [10], methods using randomized decision trees for pixel-wise inference often require heavier computation [19, 13]. Especially when increasing the tree number to improve generalizability and accuracy [4, 13], the computational burden on the multiple trees may force a trade-off between speed and accuracy.

3. Random Tree Walk

This paper follows the problem addressed by Shotton [19] and Girshick [13]; the pose estimation from a single depth image using randomized decision trees for discriminative training. The proposed random tree walk (RTW) can be described as follows. A regression tree is trained for each human joint in the skeleton. Here, we train the *direction*, not the offset, to that particular joint. Also, the directions are stored as clusters at each leaf node, so that several representative directions, along with corresponding probability weights, comprise the output of tree. When estimating the pose, we start a random walk process from some initial points. At each step in the walk, the regression tree is traversed to a leaf node where a set of directions can be obtained corresponding to the current point. The step direction is randomly chosen among this probability set. The walks are made for a fixed number of steps and the body joint position is estimated as the expectation of the walk. We note that the process is equivalent to a memory-less random walk in 3D space.

For all localization problems, a correct position can be found if the direction toward the position is known in any point on the body. Ideally, the direction to all parts should be trained from all possible positions in the whole body. This will ensure the correct joint position to be found even when starting from a random point as in Fig. 1. However, this is difficult to train and heavily redundant. Rather, a kinematic tree can be used to reduce the size of regions required for training and to provide a nearby initialization point for the RTW such that all steps are approximately in-line with the skeletal frame. Details of the method are provided in the following subsections, starting with training sample collection.

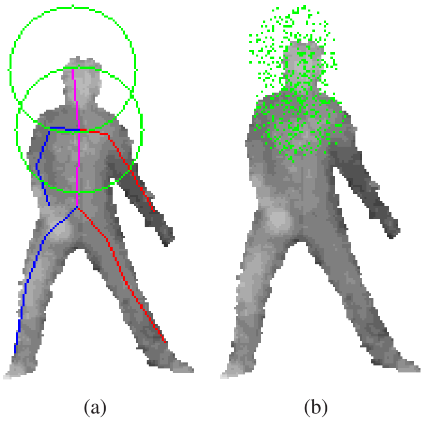


Figure 2. The offset positions are randomly sampled from head and chest joints. (a) illustrates offset sample range spheres in green. In (b), the green dots represents offset samples.

3.1. Training Set Collection and Preparation

The input data is a single depth image I comprising scalar depth measurements of a single segmented human body. For a set of body depth images $\{I^1, I^2, ...\}$, there are corresponding ground truth skeletal joint positions $\{P^1, P^2, ...\}$. Each ground truth skeletal pose is represented as 15 body joint positions $P = (p_1, p_2, ..., p_{15})$ with each $p_j = (\tilde{x}, \tilde{y}, \tilde{z})$, $j = 1, ..., 15$ representing the 3D coordinate of the j^{th} joint.

We train a regression tree that represents the direction toward a particular joint p_j from nearby random point. We thus collect samples of random points from training images with ground truth joints. For each depth I^i and $p_j \in P^i$, offset points $q = (\tilde{x} + x_o, \tilde{y} + y_o, \tilde{z} + z_o)$ are sampled with offsets x_o, y_o, and z_o in each axis having uniform distribution between $[-dist_{max}, dist_{max}]$. We implement rejection sampling technique to constrain the distance between q and p_j.

The unit direction vector to the joint from an offset point is found by $\widehat{u} = (p_j - q)/\|p_j - q\|$. A training sample S of regression tree consists of body depth image I, random offset point q, and the unit direction vector $\widehat{u}$ toward the true joint position.

$$S = (I, q, \widehat{u}). \tag{1}$$

3.2. Training Regression Tree

During training of the regression tree, the training samples are separated into different partitions using depth image feature and the offset position (I, q). The goal is to find partitioning binary tree that minimizes the sum of squared difference of $\widehat{u}$. The objective function is defined as follows.

$$E^{reg}(\mathbf{Q}) = \sum_{Q_s \subset \mathbf{Q}} \sum_{\widehat{u} \in Q_s} \|\widehat{u} - \bar{u}_s\|^2, \tag{2}$$

$$\bar{u}_s = \frac{1}{|Q_s|} \sum_{\widehat{u} \in Q_s} \widehat{u}, \tag{3}$$

where $\mathbf{Q}$ is a set of partition Q_s of training samples, and it contains all training samples. $\bar{u}_s$ is the average of unit direction in the training sample partition Q_s.

The training samples are recursively partitioned into the left and right child nodes denoted as $Q_l(\phi)$ and $Q_r(\phi)$, respectively. Here, $Q_l(\phi) \cup Q_r(\phi)$ is the training sample set at the parent node and $Q_l(\phi) \cap Q_r(\phi) = \emptyset$.At each parent node, a split parameter ϕ is randomly selected to minimize the variance at left and right partitions.

$$\phi^* = argmin_{\phi} \sum_{\widehat{u} \in Q_l(\phi)} \|\widehat{u} - \bar{u}\|^2 + \sum_{\widehat{u} \in Q_r(\phi)} \|\widehat{u} - \bar{u}\|^2, \tag{4}$$

$$\text{where } |Q_l(\phi)| > N_{min}, \text{ and } |Q_r(\phi)| > N_{min}, \tag{5}$$

where N_{min} minimum cardinality constraint on the sizes of child nodes. If the minimum size criteria cannot be met, no further split is considered.

The feature used for partition is defined, similar to that in [19], as:

$$f_\theta(I, x) = d_I\left(x + \frac{t_1}{d_I(x)}\right) - d_I\left(x + \frac{t_2}{d_I(x)}\right), \tag{6}$$

where parameters $\theta = (t_1, t_2)$ are offsets to the current pixel coordinate position x. $d_I(x)$ is the depth at x. The division by the depth $d_I(x)$ acts as transformation from world space coordinate to pixel coordinate. As with previous random forest approaches [19], $d_I(x)$ outside of human segmentation is given a large positive constant value.

Given θ together with threshold τ, $\phi = (\theta, \tau)$ partitions samples in parent node into left and right subsets Q_l and Q_r as:

$$\begin{aligned} Q_l(\phi) &= \{S_i \mid f_\theta(I, x) < \tau\}. & (7) \\ Q_r(\phi) &= Q \setminus Q_l(\phi). & (8) \end{aligned}$$

The split parameters ϕ are randomly chosen from uniform distribution. For each selection of ϕ, the partitions $Q_l(\phi)$ and $Q_r(\phi)$ are evaluated with equation (4), and the best ϕ^* is saved as the final split parameters of the node. The split parameters are found down the tree until the minimum sample number criteria (5) cannot be met, else until the maximum number of leaf size is reached. In this paper, the maximum leaf size is fixed at 32768. The leaf size of 32768 is equivalent to level 13 full tree in memory usage.

3.3. K-Mean Clusters at Leaf Nodes

In order to provide a way to escape local minima, we provide randomness to the direction selection. In the same spirit as Markov Chain Monte Carlo (MCMC) optimization approaches [12], we rely on the stochastic relaxation in choosing the unit direction. The training samples at the leaf nodes Q_s are further clustered using K-means algorithm. A leaf node in a regression tree represents a set of training samples $Q_s = \{S_1, S_2, S_3, ...\}$. The goal is to construct representative vectors $\bar{u}_k$ that minimize the variance V of unit direction vectors $\widehat{u}$ that is defined by

$$V = \sum_{k=1}^{K} \sum_{\widehat{u} \in C_k} \|\widehat{u} - \bar{u}_k\|^2, \tag{9}$$

$$\bar{u}_k = \frac{1}{|C_k|} \sum_{\widehat{u} \in C_k} \widehat{u}, \tag{10}$$

where C_k is the k^{th} cluster of Q_s. The clusters are found using typical K-means algorithm, using random initialization. Then, the average unit directions $\bar{u}_k$ are normalized as $\widehat{u}_k = \bar{u}_k / \|\bar{u}_k\|$. Traversing to the end of tree will produce a set of unit direction vectors and proportional size of clusters. Leaf set $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = \{(\frac{|C_1|}{|Q_s|}, \widehat{u}_1), ..., (\frac{|C_K|}{|Q_s|}, \widehat{u}_K), \}. \tag{11}$$

$$Q_s = \bigcup_{k=1}^{K} C_k. \tag{12}$$

The direction of the step is chosen randomly from one of the unit vectors $\widehat{u}_k$ in equation (11).

3.4. The Random Tree Walk Algorithm

Alg. 1 summarizes the proposed RTW algorithm for localizing a joint position from series of constant steps. The RTW begins with depth map I, starting position q_0, regression tree T_{reg}, number of steps N_s, and step size $dist_s$. The goal is to find the expectation of the random walk $\bar{q}$, which provides position for a single joint.

I is provided by the Kinect camera along with human segmentation. T_{reg} is trained for a specific joint. N_s and $dist_s$ are adjustable parameters which can provide tradeoff between precision and computation time. The efficient starting position q_0 will be varied upon the joint being localized.

```
Data: Depth map $I$, starting point $q_0$, regression tree
      $T_{reg}$, number of steps $N_s$, and step distance
      $dist_s$.
Result: The average joint position $\bar{q}$.
Initialization. $m = 0$, $q_{sum} = (0, 0, 0)$.
while $m < N_s$ do
    Find the leaf node $\mathcal{L}$ of $T_{reg}$ using $(I, q_m)$.
    $\mathcal{L} = \{(\frac{|C_1|}{|Q_s|}, \widehat{u}_1), ..., (\frac{|C_K|}{|Q_s|}, \widehat{u}_K)\}$
    Randomly select $k^{th}$ cluster with probability
    $Prob(k) = \frac{|C_k|}{|Q_s|}$.
    Step into new position using $k^{th}$ direction vector.
    $q_{m+1} = q_m + \widehat{u}_k \cdot dist_s$.
    Update joint position sum $q_{sum}$ += $q_{m+1}$.
    Update $m$ += 1.
end
Compute the joint position by averaging step
positions. $\bar{q} = q_{sum}/N_s$.
```

Algorithm 1: The Random Tree Walk Algorithm

The novelty of the proposed algorithm comes from the construction of step direction distribution from trained regression tree. The leaf node is found deterministically, but the direction of the step is chosen randomly from K-mean clustered unit vectors at the leaf. The k^{th} sampling probability distribution is determined from the proportional size of k^{th}-mean cluster. The expectation of sequential step positions $\bar{q}$ is estimated as the joint position.

3.5. Kinematic Tree Pose Estimation

Since RTW expectation can operate independently for each joint, all joint positions can be found in parallel. However, a sequential estimation will provide better starting point q_0 for each RTW. A starting point closer to the targeted joint position is desired because random tree walk will reach the joint position faster. With known skeletal topology, the firstly estimated joint position can provide the starting point for next adjacent joint.

The 15 joints in skeletal frame are as follows: head, chest, belly, L/R hip, L/R shoulders, L/R elbows, L/R wrists, L/R knees and L/R ankles. The placement of the each joint is illustrated in Fig. 3 (a). The joint positions are determined sequentially according to the typical skeletal topology. The process is illustrated in Fig. 3. First, the RTW for belly position starts from the body center. Then, the expectation of RTW $\bar{p}$ is used as the q_0 of next adjacent joint. An example of the sequential RTW is shown in Fig. 3 (b). Note that the computation for each limb may be processed in parallel, if further computational time reduction is desired.

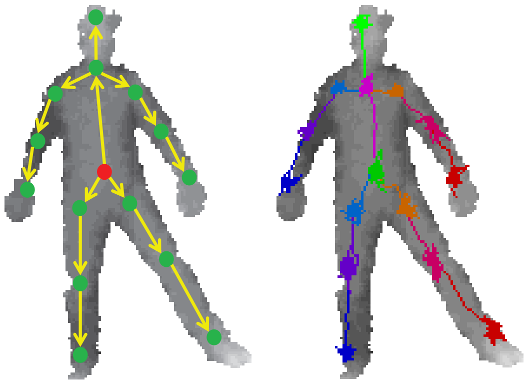


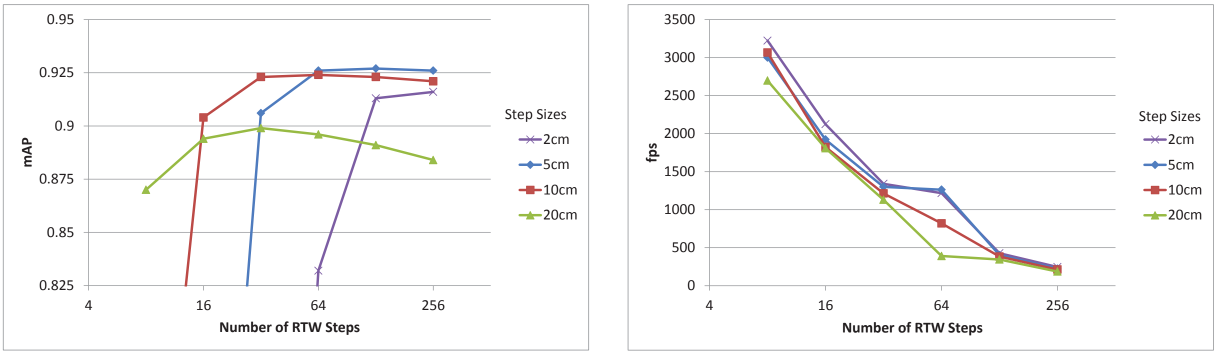
Figure 3. The adjacent joint positions can be used as the starting positions for new RTW. (a) illustrates the kinematic tree implemented along with RTW. First, the random walk toward belly positions starts from body center. The belly positions (red dot in (a)) become starting point for hips and chest, and so forth. (b) shows the RTW path examples.

4. Experiments

The RTW approach shows robustness to publicly available SMMC-10 set [9] and EVAL set [10]. The SMMC-10 set contains 28 action sequences performed by a single individual. More recent EVAL set has 3 models performing 8 sequences each. The EVAL set is more suitable for our approach because it provides multiple models for train and test set separation. Three sets of trees are constructed for each model. The sequences for each model is evaluated using trees trained from other 2 models as in a typical leave-one-out training scheme. The majority of evaluation results are obtained from the EVAL set. The SMMC-10 set contains only a single individual. Thus we constructed our own training set for the SMMC-10 test set with manually labelled skeleton positions.

A conventional precision measure for pose estimation algorithms is the 10 cm rule [9, 19, 13]. If the estimated joint position is within 10 cm of the ground truth, it is considered correctly estimated. The precision of the estimated twelve joints on each frame are averaged together to find mean average precision (mAP). The joints included in the evaluations are head, chest, L/R shoulders, L/R elbows, L/R wrists, L/R knees, and L/R ankles. The evaluation joints are chosen based on the availability of ground truth.

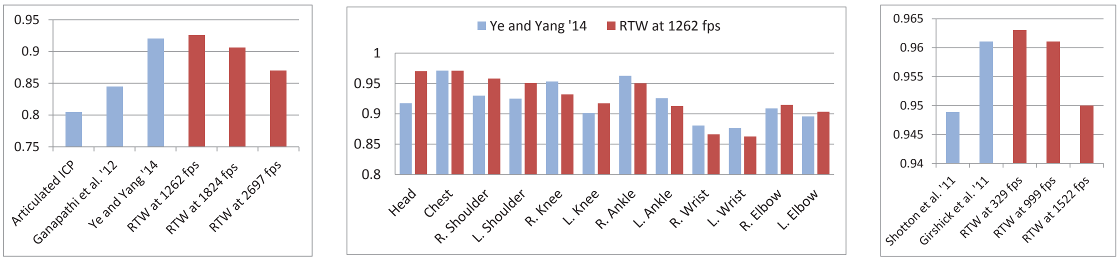
First, the properties of random walk sub-sampling method are evaluated in terms of step size and step number. The trade-off between computation time and precision can be derived from the number of random walks and step distance. The computation and precision of the proposed method is compared with previous methods based on both tracking and discriminative approaches [10, 28, 19, 13].



(a) Mean Average Precision (mAP) of the proposed RTW method on EVAL set [10] with varying step numbers and sizes

(b) Frames per seconds (fps) of the proposed RTW method on EVAL set [10] with varying step numbers and sizes

Figure 4. (a) shows the number of RTW steps versus the mAP for the EVAL sequence. Each line represents different step sizes. The number of RTW steps versus average fps operation is plotted in (b).



(a) mAP for EVAL set

(b) mAP of each joint for EVAL set

(c) mAP on SMMC-10 set

Figure 5. The proposed approach is compared with the recent algorithms using EVAL [10] set in (a) and (b). (a) shows our 3 results in red with different fps. RTW approach performs slightly higher than Ye and Yang [28] at 1262 fps. Even at 2687 fps, RTW shows higher precision than Ganapathi et al. [10]. The precisions in each joint are presented in (b). (c) compares results for SMMC-10 test sequence. Shotton et al. [19], Girshick et al. [13] and our approach are methods for pose estimation from a single image.

4.1. RTW Step Size and Number

Intuitively, larger number of random walk steps will result in more accurate or otherwise more stable pose estimation. Since the directions of steps are chosen randomly, there is always a chance for a wrong direction. A large number of steps will hopefully average out the errors. At the same time, unnecessary computation time should be avoided. If a further increase in the number of steps does not result in significant increase of precision, the saturation number of steps has been reached.

Different step sizes have different saturation numbers. If each step is large, the targeted joint position will be reached quickly, provided that the random direction selections are mostly correct. If the step size is too large, however, the random walks may keep stepping over the correct joint position. The step sizes were varied among 2 cm, 5 cm, 10 cm, and 20 cm in the evaluation as presented in Fig. 4. The mAP for different step size and number of steps are tested for the EVAL set. The number of steps are varied from 8, 16, 32, 64, 128, and 256.

Overall, the precision starts to saturate early for larger step sizes. The saturation number for the step sizes of 2 cm, 5 cm, 10 cm and 20 cm are 128, 64, 32, and 16 respectively. The highest precision 0.927 is achieved with 5 cm step size and 128 steps. Similar precision of 0.926 is achieved for 64 and 256 steps with the same step size. The maximum mAP achieved for different RTW steps are found in Table 1.

The number of steps are plotted against the fps in Fig. 4 (b). The fps was computed by averaging the computation time over all 24 sequences, 8 sequences per 3 models. At 8 RTW steps per joint, the proposed algorithm can output 2697 fps with 0.870 mAP which is higher than the recent implementation by Ganapathi et al. [10]. RTW of eight steps are taken each for 15 joints, translating into 120 (8 × 15) traversals of level 13 trees.

4.2. Comparison with Existing Methods

The recent state-of-the-art pose tracking algorithm implementation by Ye and Yang showed 0.921 mAP on EVAL

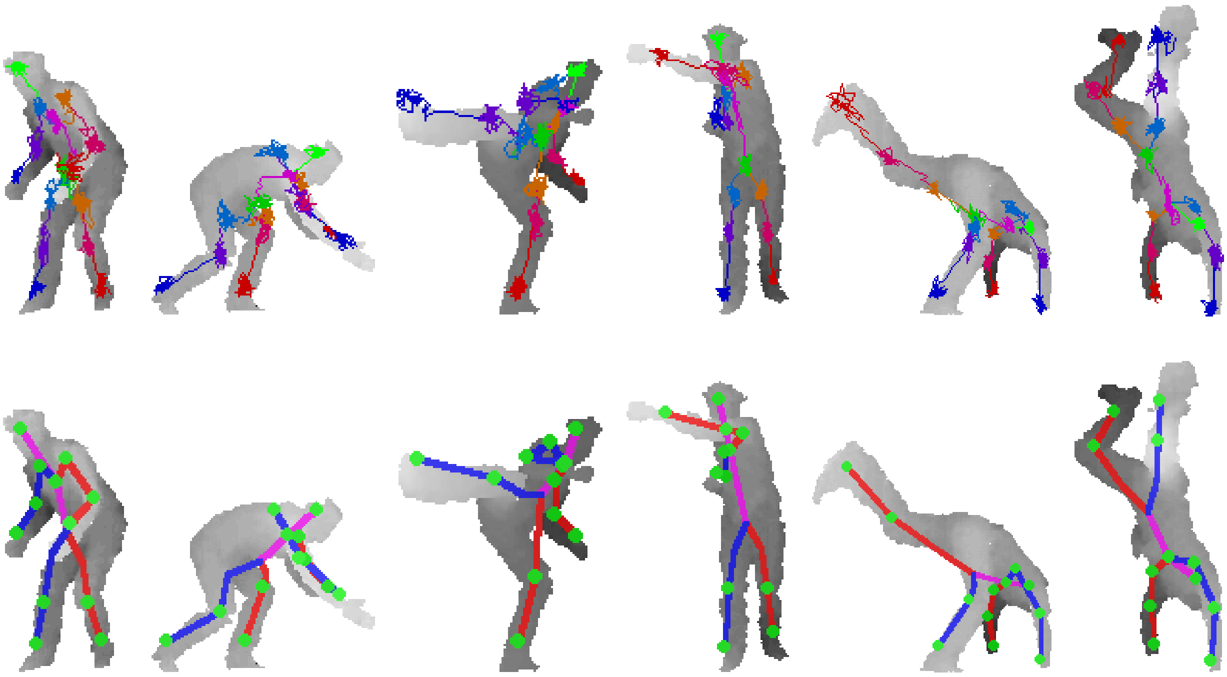


Figure 6. Example results of the RTW from EVAL test data. 64 RTW steps are taken for each joint. The RTW paths are drawn at the top row, the expectations of RTW steps are used to find joint positions in bottom row. The pose estimation from a single image takes less than 1 millisecond.

steps	8	16	32	64	128	256
mAP	0.870	0.904	0.923	0.926	0.927	0.926
fps	2697	1824	1213	1262	402	224
size	20 cm	10 cm	10 cm	5 cm	5 cm	5 cm

Table 1. EVAL set's highest precisions are presented for different RTW step numbers, followed by the corresponding fps and step sizes. 64 RTW steps with 5 cm size is clocked faster than 32 RTW steps with 10 cm size. This may be due to CPU's cache management; smaller steps make it more likely to find new feature pixels in cache.

sequences [28]. Their implementation achieved greater than 30 fps operation using GPU. The pose tracking method by Ganapathi et al. does not rely on GPU to achieve 125 fps operation, but the precision is significantly lower [10]. In Fig. 5 (a), the proposed RTW is compared with these two methods [10, 28]. RTW is able to obtain slightly higher precision than Ye and Yang's state-of-the-art tracking algorithm. In addition, the computation is more than 40 times faster than their GPU assisted implementation [28].

Another advantage of RTW over previous tracking algorithms comes from the ability to estimate pose from a single depth image. In our approach, the pose of previous frame is not used as either the initial pose or a-priori knowledge for the estimation of pose in current frame. In that regard, the proper comparison should be made with the single depth image pose estimation methods of [19, 13]. However, since their evaluation results for the EVAL set are unavailable, the older SMMC-10 test sequences are used in the comparison test. See Fig. 5 (c). Since SMMC-10 set mostly contains relatively easy poses, the results of each method are all fairly high, above 0.94 mAP.

Similar to the EVAL set, the proposed RTW obtains slightly higher precision for the SMMC-10 set, but the computation gain is significant. The previous methods' the 8 core CPU implementations run at 50 and 200 fps [19, 13]. In comparison, the proposed RTW run at approximately 1000 fps using a single core CPU, awhile obtaining equal or higher mAP.

Finally, few qualitative examples from EVAL test sequences are shown in Fig. 6. In the figure, 64 RTW steps are taken with 5 cm step size. Both the RTW paths and RTW pose expectations are shown. Note that few very hard poses like hand-stand, crouching, and cross-punch are efficiently and accurately estimated with RTW.

5. Conclusion

In this paper, RTW approach for 3D pose estimation problem is proposed, which gives a large computation gain without decrease in accuracy. The proposed approach moves away from pixel-wise classification, and applies a supervised gradient descent and MCMC like random sampler in the form of Markov random walks. By initializing the starting point to the adjacent joint according to kinematic tree, we demonstrate a robust and super-real-time pose estimation algorithm.

Acknowledgments

This work was supported by Hankuk University of Foreign Studies Research Fund of 2015. Also, this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2013R1A1A2A10004550).

References

[1] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2006.

[2] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel, and C. Theobalt. A data-driven ap- proach for real-time full body pose reconstruction from a depth camera. *Proc. Int'l Conf. Computer Vision*, 2011.

[3] L. Bourdev and J. Malik. Body part detectors trained using 3d human pose annotations. *Proc. Int'l Conf. Computer Vision*, 2009.

[4] L. Breiman. Random forest. *Machine Learning*, 45:5–32, 2014.

[5] T. F. Cootes, M. C. Ionita, C. Lindner, and P. Sauer. Robust and accurate shape model fitting using random forest regression voting. In *Computer Vision–ECCV 2012*, pages 278–291. Springer, 2012.

[6] A. Elhayek, C. Stoll, N. Hasler, K. I. Kim, H.-P. Seidel, and C. Theobalt. Spatio-temporal motion tracking with unsynchronized cameras. *Proc. Conf. Computer Vision and Pattern Recognition*, 2012.

[7] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. *Proc. Conf. Computer Vision and Pattern Recognition*, 2008.

[8] J. Gall, C. Stoll, E. de Auiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. *Proc. Conf. Computer Vision and Pattern Recognition*, 2009.

[9] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. *Proc. Conf. Computer Vision and Pattern Recognition*, 2010.

[10] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-time human pose tracking from range data. *Proc. European Conf. Computer Vision*, 2012.

[11] D. M. Gavrila and L. Davis. 3-d model-based tracking of humans in action: a multi-view approach. *Proc. Conf. Computer Vision and Pattern Recognition*, 1996.

[12] S. Geman and D. Geman. Stochastic relaxation, gibbs distribution, and the bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[13] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. *Proc. Int'l Conf. Computer Vision*, 2011.

[14] D. Grest, V. Kruger, and R. Koch. Single view motion tracking by depth and silhouette information. *Scandinavian Conference on Image Analysis*, 2007.

[15] T. Helten, A. Baak, G. Bharaj, M. Muller, H.-P. Seidel, and C. Theobalt. Personalization and evaluation of a real-time depth-based full body tracker. *3DV*, 2013.

[16] Microsoft. Kinectsdk. http://www.microsoft.com/en-us/kinectforwindows/, 2014.

[17] J. Oberg, K. Eguro, R. Bittner, and A. Forin. Random decision tree body part recognition using fpgas. *Int'l Conf. on Field Programmable Logic and Application*, 2012.

[18] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. *Proc. Int'l Conf. Computer Vision*, 2003.

[19] 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. *Proc. Conf. Computer Vision and Pattern Recognition*, 2011.

[20] M. Sun, P. Kohli, and J. Shotton. Conditional regression forests for human pose estimation. *Proc. Conf. Computer Vision and Pattern Recognition*, 2012.

[21] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3786–3793. IEEE, 2014.

[22] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. *Proc. Conf. Computer Vision and Pattern Recognition*, 2012.

[23] X. Wei, P. Zhang, and J. Chai. Accurate realtime full-body motion capture using a single depth camera. *SIGGRAPH ASIA*, 2012.

[24] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 532–539. IEEE, 2013.

[25] H.-D. Yang and S.-W. Lee. Reconstruction of 3d human body pose from stereo image sequences based on top-down learning. *Pattern Recognition*, 2007.

[26] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. *Proc. Conf. Computer Vision and Pattern Recognition*, 2011.

[27] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys. Accurate 3d pose estimation from a single depth image. *Proc. Int'l Conf. Computer Vision*, 2011.

[28] M. Ye and R. Yang. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. *Proc. Conf. Computer Vision and Pattern Recognition*, 2014.

[29] Y. Zhu, B. Dariush, and K. Fujimura. Controlled human pose estimation from depth image streams. *Proc. CVPR Workshop on TOF Computer Vision*, 2008.