# Hough Forest With Optimized Leaves for Global Hand Pose Estimation With Arbitrary Postures

Hui Liang, *Member, IEEE*, Junsong Yuan, *Senior Member, IEEE*, Jun Lee, *Member, IEEE*, Liuhao Ge, *Student Member, IEEE*, and Daniel Thalmann

***Abstract*—Vision-based hand pose estimation is important in human–computer interaction. While many recent works focus on full degree-of-freedom hand pose estimation, robust estimation of global hand pose remains a challenging problem. This paper presents a novel algorithm to optimize the leaf weights in a Hough forest to assist global hand pose estimation with a single depth camera. Different from traditional Hough forest, we propose to learn the vote weights stored at the leaf nodes of a forest in a principled way to minimize average pose prediction error, so that ambiguous votes are largely suppressed during prediction fusion. Experiments show that the proposed method largely improves pose estimation accuracy with optimized leaf weights on both synthesis and real datasets and performs favorably compared to state-of-the-art convolutional neural network-based methods. On real-world depth videos, the proposed method demonstrates improved robustness compared to several other recent hand tracking systems from both industry and academy. Moreover, we utilize the proposed method to build virtual/augmented reality applications to allow users to manipulate and examine virtual objects with bare hands.**

***Index Terms*—Gesture recognition, hand pose estimation, Hough forest.**

## I. Introduction

VISION-BASED hand tracking and pose estimation serves as an effective tool for human–computer interaction (HCI) [1], [2]. Recently, this field has gained considerable progresses with the advent of depth cameras, such as random forests [3], [4], articulated iterative-closest-point (ICP) algorithm [5], and convolutional neural network (CNN)-based methods [6]–[8]. However, these methods mostly emphasize on full degree-of-freedom (DOF) hand pose estimation, but we find that their accuracy of global hand rotation estimation with a single depth camera is still unsatisfactory for HCI applications that require precise rotation control, especially for some ambiguous hand postures. Consider the scenario of virtual object grasping, in which a user needs to pose his/her hand as a fist to perform a grasping motion and rotate the fist to change the viewpoints of a virtual object. As a fist posture has a relatively less discriminative pattern for hand rotation estimation compared to that of a stretched hand, we observe frequent hand rotation estimation failures with some full-DOF solutions, such as [5] and [9]. Fig. 1 illustrates some exemplar results for hand rotation estimation with a recent full-DOF hand tracking framework [9], which is reasonably accurate for a fully stretched hand posture but not reliable with a fist posture. We conjecture that this may be due to different emphasizes between full-DOF hand pose estimation and global hand pose estimation. In the former, the entire hand region is emphasized in order to recover all the finger and palm pose parameters. In the latter, some discriminative local hand parts tend to produce confident hand rotation estimations, while some other parts tend to produce ambiguous estimations. Finally, full DOF articulated pose estimation can benefit if the hand rotation angles are accurately estimated [3], [10].

Therefore, we emphasize on robust estimation of 6-DOF global hand pose for arbitrary postures with a single depth camera. The global hand pose includes 3-D hand rotation and translation, which are sufficient for many HCI applications with a small alphabet of static hand postures despite high dimensionality of full DOF hand pose [11]–[13]. A very similar problem is 3-D global head pose estimation [14], [15] to recover head rotation and translation in input images. However, different from head which is nearly rigid, global hand pose estimation is more challenging due to large hand posture variations.

The Hough forest [16] is utilized to estimate global hand pose using a single depth camera. The success of Hough forest in hand pose analysis is largely attributed to its voting scheme. The basic concept is to fuse the votes cast by a set of voting elements for robust prediction. To be specific, local features, e.g., small image patches, are first sampled from a number of different locations over input image and each of them retrieves a pose vote from a pretrained Hough forest. The final prediction is made by fusing all votes based on certain criteria, e.g., average pooling. Such voting techniques are

Manuscript received June 19, 2016; revised February 18, 2017 and October 4, 2017; accepted November 27, 2017. This work was supported by the Singapore Ministry of Education Academic Research Fund under Grant Tier 2 MOE2015-T2-2-114. This paper was recommended by Associate Editor K.-S. Hwang. *(Corresponding author: Hui Liang.)*

H. Liang was with the Institute for Media Innovation, Nanyang Technological University, Singapore (e-mail: hliang1@e.ntu.edu.sg).

J. Yuan is with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore (e-mail: jsyuan@ntu.edu.sg).

J. Lee was with the Institute for Media Innovation, Nanyang Technological University, Singapore. He is now with the Division of Computer and Information Engineering, Hoseo University, Asan 31499, South Korea.

L. Ge is with the Institute for Media Innovation, Nanyang Technological University, Singapore.

D. Thalmann was with the Institute for Media Innovation, Nanyang Technological University, Singapore. He is now with the École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2779800

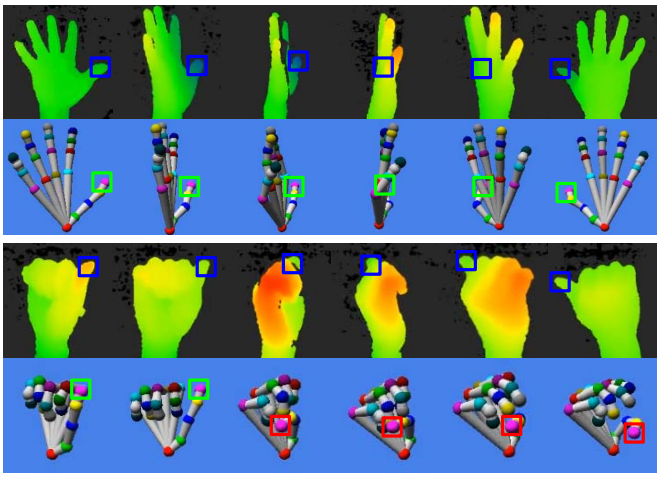

Fig. 1. Rotation estimation with [9] for a hand turning from right to left in depth images. Upper: results with a stretched hand posture that has a discriminative pattern. Lower: results with a fist posture that has ambiguous appearance for rotation estimation. The thumb fingers are enclosed with squares to illustrate hand rotation. □: reference thumb positions in input depth images. □: reference thumb positions for *correct* rotation estimation. □: reference thumb positions for *incorrect* rotation estimation.

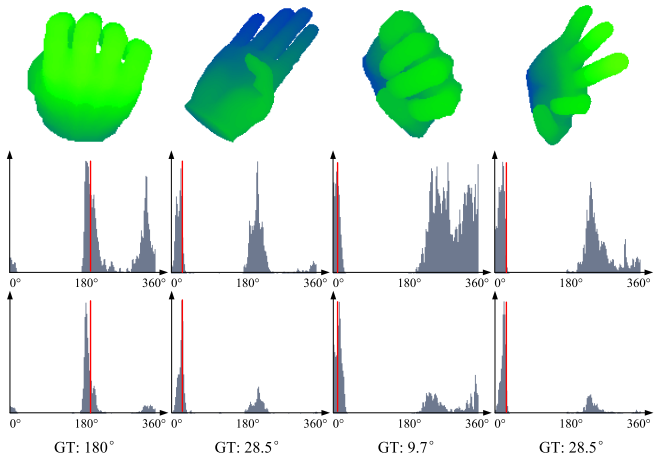


Fig. 2. Illustration of ambiguity encountered in Hough-voting-based yaw rotation prediction. Note the rotation angle is periodic with 360°. First row: query depth images. Second row: per-pixel vote distributions in the interval $[0, 360°]$. Voting pixels are uniformly sampled over the hand region with uniform voting weights. Third row: per-pixel vote distributions with optimized voting weights. Red lines in histograms and texts at the bottom denote ground truth rotation angles.

common for discriminative pose estimation for a wide range of objects, e.g., head [15], body [17], and hand [3], [18]. These methods prove more efficient and robust compared to those that rely on the global image features only. In the proposed method, we also follow this principle to fuse the pose votes from a set of densely sampled pixels to infer the global hand pose.

Since the voting elements may contribute unequally to the fused prediction, one important issue is to determine the strategy to aggregate the Hough votes for hand pose. Take Fig. 2 as an example, in which we want to infer the yaw rotation of the hand via Hough-voting. As many local features can have similar appearances for different hand rotation angles, the vote distributions from these local features often form multiple peaks at ambiguous poses in the histograms. As a result, the fused prediction via average-pooling or mode-seeking cannot provide satisfactory results. The task to predict roll and pitch rotation suffers from similar problems, and thus ambiguous pose votes need to be suppressed. In some previous work this is achieved by adding different weights to these votes during fusion, in which a confident vote is assigned large weight and vice versa, e.g., vote length threshold [17] or leaf-variance-based weights [15]. However, they are not principled solutions.

In contrast, we propose to automatically determine the optimal voting weights during the training phase in global hand pose estimation. These weights are stored in the leaf nodes of the Hough forest and can be retrieved during the testing phase. That is, each leaf node in the Hough forest includes not only a vote for hand pose but also the corresponding weight to reflect the confidence of this vote during fusion, both of which are learned during forest training. To this end, we first follow a way similar to that in [3] and [19] to build the tree structures of Hough forest and to learn the pose vote at the leaf nodes. Then, we propose a novel algorithm to retrain the Hough forest with fixed tree structures in order to optimize the voting weights stored at the leaf node, so that the fused prediction error via Hough voting is minimized for each training image. We show that the weight optimization problem can be formulated as a constrained quadratic programming problem and can be solved efficiently. Fig. 2 illustrates the effectiveness of the proposed method by comparing the vote distribution histograms for yaw rotation prediction from the Hough forest with uniform leaf weights and optimized leaf weights, and the detailed procedure to obtain the histograms can be found in Section IV-A. The vote distribution is more concentrated on the ground truth pose with optimized leaf weights.

The experimental results on a synthesized dataset and a real-world dataset demonstrate that the proposed Hough forest with optimized leaves can estimate hand pose accurately compared to traditional Hough forest [3] and recent CNN-based methods [6], [8]. Particularly, in contrast to the CNN-based methods that generally need powerful GPU cards for training/testing and large amount of storage to store the learned parameters of convolutional layers for feature extraction and fully connected layers for prediction, the proposed method can achieve comparable prediction accuracy and frames-per-second but with only CPU support and a hand-crafted feature [20], and the trained model is very compact for storage. This makes the proposed method flexible in application, such as on mobile platforms. For qualitative evaluations, we have tested the proposed method with different depth cameras, including a SoftKinetic DS325 camera and an Intel Realsense camera using the forest trained on synthesis dataset, which shows it can correctly predict hand pose for very challenging postures, e.g., a fist posture rotating backwards, while the state-of-the art methods [5], [9] cannot work robustly for such cases. Moreover, we utilize the proposed algorithm to build two augmented/virtual reality applications to manipulate virtual objects with bare hands for natural interaction.

The remainder of this paper is organized as follows. In Section II, we provide a literature review of related techniques. In Section III, we present a detailed description and analysis of the proposed hand pose estimation scheme. In Section IV, we show the experimental results, performance comparison, and the HCI application. In Section V, we give our concluding remarks and further work.

## II. Related Work

There has been a lot of work on vision-based hand pose estimation in literature. Among them, model-based fitting and template-matching are two main categories of methods, in which the optimal pose is inferred in either a generative or discriminative manner, respectively. The model-based fitting methods are usually built upon a generative deformable hand model and seek for the optimal pose by iterative adjustment of pose parameters of the model and compatibility check between model features and input images [21]. In [22], multiple hand silhouettes are extracted from the images captured with several cameras around the hand, where the background is set using blue boards for easy hand segmentation. A voxel model is generated with the multiview data and matched to a 3-D hand model, and the optimal pose is sought to make the hand model surface stay inside the voxels. In [23], the texture and shading of the skin are captured from input images and synthesized in the hand model, and the illumination sources are controlled in real scenario and simulated during hand modeling. A variational formulation is proposed to estimate the full DOF hand pose. In this way, hand pose is recovered quite accurately since matching ambiguity is largely reduced. However, this method is difficult to use in real HCI scenarios. In [24], a Kinect depth camera is adopted to capture the hand image as it can better handle the background clutter and pose ambiguity in monocular color image, the particle swarm optimization algorithm is used to find the optimal pose that best fits the image projection of a 3-D hand model to the input depth image and skin silhouette. With the point clouds generated by the depth camera, the iterative closest points algorithm and its extensions to articulated objects are also commonly used for hand pose estimation [25], [26], which iteratively build point-to-point correspondences between model and input point cloud and seek for the skeleton transform to minimize the distance between the point pairs. In [5], the hand model is fitted to both input 3-D point cloud and 2-D hand silhouette for hand pose estimation. Particularly, optimization of the model-fitting problem is performed with respect to prelearned low dimensional hand pose prior, kinematic prior, and temporal prior altogether to avoid infeasible hand poses and to improve temporal smoothness. In [27], an elaborate hand model is adopted to fit to multiview inputs of eight HD cameras, so that very subtle hand motion can be captured. The fingernails are detected in each view by Hough forest classifiers and used to assist model-fitting for improved accuracy. Similar ideas have been adopted for RGB-D cameras [28], [29], in which fingertips are detected in depth images by SVM classification or morphological analysis and used in combination with other low-level image features for model-fitting stage, which can help to speed up convergence and to avoid local optima. Generally, the initial pose is important to achieve good model-fitting results, which can be obtained from either temporal reference during tracking [5], [24] or template-matching methods [29], [30].

The template-matching methods infer hand pose parameters by directly mapping image features to preindexed templates. Generally, they need to build a large dataset to cover the possible hand postures and index it for fast search. During testing, the input hand pose is recovered by looking for the templates that share the similar features. In [31], the hand edge image is encoded into a score value vector by matching to a predefined set of shape templates, and a multivariate relevance vector machine uses it as the input to retrieve some pose hypotheses. The optimal pose is obtained by a verification stage with the hand model projection. In [12], a two-camera system is presented to capture 6-DOF palm motion and simple gestures like pinching or pointing for both hands. A pair of hand silhouettes is extracted and coded into binary strings for fast query in the database to retrieve the hand pose. In [32], an isometric self-organizing map is used to learn a nonlinear mapping between image features and pose, which reduces the dataset redundancy by grouping templates with similar features and poses together. The hand edges are captured at only depth discontinuities with a multiflash camera and encoded into shape context for matching. In [20], the hand joint parameters are decomposed into many overlapping subsets and the local sensitivity hashing algorithm is adopted to get the partial estimations for the subsets, which are further integrated by an EM-like algorithm to obtain the fused prediction. Since monocular color images lack discriminative power for hand pose recovery, template-matching-based methods usually retrieve a set of ambiguous pose candidates. Traditionally, this issue is solved using multicamera setting [12], [33], temporal constraints [34], or verification with a hand model [31], [35].

The depth cameras largely alleviate ambiguous predictions of template-matching methods as they can capture more detailed 3-D structure of the hand surface to resolve ambiguity. Girshick *et al.* [17] proposed to directly regress for the body joints from the depth images. With a prelearned random regression forest, a set of voting local image patches are randomly sampled from the image and cast their votes for the joint positions, and the votes from all the voting-elements are fused by mean-shift algorithm to give the final predictions. As each voting patch only utilizes partial image observations, the method is more robust than that rely on global image descriptors against imperfect inputs, e.g., inaccurate hand segmentation. Very similar ideas are then used for head and hand pose estimation [3], [15] and prove to be effective. However, in these earlier works [3], [17], the correlations between different body and hand joints are not well utilized during regression. Therefore, many following researches have focused on how to utilize such correlations with the random forest, e.g., hierarchical regression [4], [36] and late fusion with hand joint correlations [37], [38]. But these improvements do not directly benefit our task to estimate the unconstrained 6-DOF hand motion.

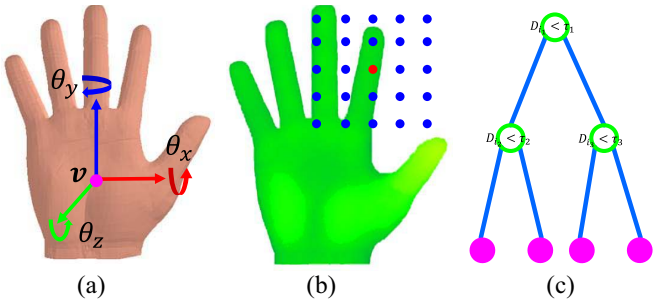


Fig. 3. (a) Global hand pose to be predicted. This example has zero rotation angles for all three axes. (b) Depth context descriptor [20]. The red circle denotes the current pixel and the blue circles denote the context points. (c) Random decision tree in Hough forest.

Recently, the CNN-based methods are gaining popularity in the field of hand pose estimation [6]–[8], [39], which include convolutional layers for image feature extraction and fully connected layers for prediction. During training, the parameters of both convolutional layers and fully connected layers are jointly learned to minimize prediction error, so that feature extraction is optimized for the hand pose estimation task. With such capabilities, these methods have demonstrated superior performance for full-DOF hand pose estimation. However, they generally need GPU support during runtime and relatively large memory space to store the network parameters. This may pose an issue on emerging mobile platforms, such as Oculus Rift and Microsoft Hololens, which are limited in computational power and storage.

## III. Hough Forest With Optimized Leaves

As discussed in Section I, our goal is to estimate global hand pose $\boldsymbol{\Phi}$, including 3-D hand rotation and translation from depth images, which are defined as Euler angles of pitch, yaw, and roll rotations of the hand and the 3-D hand position, i.e., $\boldsymbol{\Phi} = (\boldsymbol{\theta}, \boldsymbol{v})$, where $\boldsymbol{\theta} = (\theta_x, \theta_y, \theta_z)$ is the global rotation angles, and $\boldsymbol{v} = (x_c, y_c, z_c)$ is the position of an anatomical stable point on the hand such as the palm center, as illustrated in Fig. 3(a).

The Hough forest [16] is utilized to predict $\boldsymbol{\Phi}$ from a single input frame $I$. It is an ensemble of $T$ random decision trees, each of which is trained independently with a bootstrap training set. Following the previous work on Hough-voting-based pose estimation [3], [18], the Hough forest is learned to map the local features of a set of Hough voting pixels to the probabilistic votes for the hand pose. During testing, the per-pixel votes from all voting pixels are fused to predict $\boldsymbol{\Phi}$, which proves quite robust against noisy inputs. We adopt the depth context descriptor [20] as local pixel feature for regression with the Hough forest. For a pixel $\boldsymbol{p}$, its depth context descriptor $\mathcal{D}$ is defined as the set of depth differences between $\boldsymbol{p}$ and a number of context points

$$\mathcal{D} = \left\{ I\left(\boldsymbol{p} + \frac{\boldsymbol{u}_i}{I(\boldsymbol{p})}\right) - I(\boldsymbol{p}) \middle| i = 1, \ldots, B \right\} \tag{1}$$

where $I(\boldsymbol{p})$ is the depth value at pixel $\boldsymbol{p}$, $\boldsymbol{u}_i$ is the offset between the context point and the current pixel and $\boldsymbol{p} + \boldsymbol{u}_i / I(\boldsymbol{p})$ is the depth-normalized coordinate of the context point to ensure depth invariance of the descriptor. Fig. 3(b) shows an example of $\mathcal{D}$. This kind of features relying on the depth difference is depth-invariant, fast to compute and has demonstrated their good performance for both pose estimation and gesture recognition [3], [18], [20]. To ensure $\mathcal{D}$ can capture the fine 3-D structure of the neighborhood, its dimensionality is around several hundreds. Each tree in the Hough forest contains a set of nonleaf nodes and leaf nodes, which are denoted as green and pink circles in Fig. 3(c), respectively. The root node is at the top level of each tree. Each nonleaf node contains a split function and has two children nodes. The split function is defined as a Boolean function to determine which child branch a pixel should reach based on the feature value of its depth context descriptor during testing, i.e., a pixel reaches the left branch if the split function value is *true* and vice versa. Each leaf node stores a single relative vote for hand pose and the associated voting weight, which are also learned ruing training. During testing, if a pixel reaches a leaf node, its pose vote and weight are assigned to this pixel for Hough voting. Let the vote and weight be $(\bar{\boldsymbol{\theta}}, \bar{\Delta}, \boldsymbol{w})$, where $\bar{\boldsymbol{\theta}}$ is the prediction of hand rotation angles, $\bar{\Delta}$ is the 3-D offset between a pixel and the predicted palm center, and $\boldsymbol{w}$ is a 6-D vector to represent the voting weight of each dimension of hand pose. Their detailed description can be found in Section III-A.

Fig. 4 illustrates the Hough-voting-based hand pose prediction pipeline for a query image $I_t$ using a pretrained Hough forest. First, a set of $N_s$ voting pixels $\{p_i\}$ are uniformly sampled over the hand region in $I_t$. Second, these pixels cast their pose votes independently with the Hough forest. To this end, each pixel $p_i$ is sent to the root node of each tree, and determines to reach whether the left or right branch based on its depth descriptor value at the nonleaf nodes, and thus gradually branches down each tree in the forest until a leaf node is reached. Therefore, each voting pixel retrieves totally $T$ votes from the leaf nodes it reaches in all the trees. Let the votes be $\{\boldsymbol{\Phi}_{ij}, \boldsymbol{w}_{ij}\}_{j=1}^{T}$, where $\boldsymbol{\Phi}_{ij}$ is the vote for the hand pose and $\boldsymbol{w}_{ij}$ is the voting weight. Here, $\boldsymbol{\Phi}_{ij}$ is converted from the relative pose vote $(\bar{\boldsymbol{\theta}}_{ij}, \bar{\Delta}_{ij})$ retrieved from the forest by setting $\boldsymbol{v}_{ij} = \bar{\Delta}_{ij} + \boldsymbol{v}_i$, where $\boldsymbol{v}_i$ is the 3-D position of the pixel. Finally, the fused pose prediction is obtained via Hough voting as follows.

To perform Hough-voting, a score function $P(\boldsymbol{\Phi}|I)$ is obtained by aggregating the individual votes from all the voting pixels. Since $\boldsymbol{\Phi}$ is unconstrained in 6-D space, its different dimensions are thus uncorrelated. Therefore, we can take $P(\boldsymbol{\Phi}|I) = \prod_{\phi \in \boldsymbol{\Phi}} P(\phi|I)$, where $\phi$ is one dimension of $\boldsymbol{\Phi}$. This allows us to solve for each dimension of the six parameters in $\boldsymbol{\Phi}$ independently during inference. Similar to [3], we use the weighted Parzen density estimator to evaluate $P(\phi|I)$

$$P(\phi|I) = \sum_i P(\phi|p_i) = \sum_{i,j} w_{ij} P(\phi|\phi_{ij}) \tag{2}$$

where $\phi_{ij}$ is one dimension of the pixel vote $\boldsymbol{\Phi}_{ij}$. Based on this expression, each retrieved pose vote contributes to the fused prediction via linear weighting. For hand translation parameters $\phi \in \boldsymbol{v}$ we adopt the Gaussian kernel for $P(\phi|\phi_{ij})$ with an

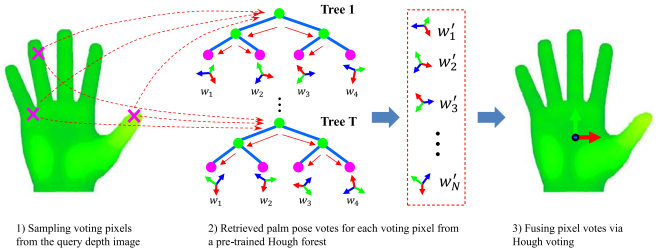


Fig. 4. Hand pose estimation procedure with a trained Hough forest. Here only hand rotation is illustrated for clarity, which is represented by the rotated 3-D coordinate system. $w$ is the voting weight of the leaf node.

isotropic variance $\delta_v$ in all three dimensions

$$P(\phi|\phi_{ij}) = \frac{1}{\delta_v\sqrt{2\pi}} \exp\left(-\frac{\|\phi - \phi_{ij}\|^2}{\delta_v^2}\right). \tag{3}$$

For hand rotation parameters $\phi \in \boldsymbol{\theta}$ we cannot assume a Gaussian distribution for $P(\phi|\phi_{ij})$ since the angle is in non-Euclidean space. Instead, we utilize an 1-D wrapped Gaussian kernel [40] to model $P(\phi|\phi_{ij})$ for these rotation parameters within the range $[0, 2\pi]$

$$P(\phi|\phi_{ij}) = \sum_{z\in\mathbb{Z}} \mathcal{N}(\phi - 2z\pi; \phi_{ij}, \delta_\theta^2) \tag{4}$$

which is basically infinite wrappings of linear Gaussian within $[0, 2\pi]$. In practice it is shown that the summation over $z \in [-2, 2]$ approximates the infinite summation in (4) well enough [41]. Combining (2)–(4) we see that $P(\phi|I)$ is still sum of Gaussians for both the translation and rotation parameters. Therefore, for single-frame hand pose, we can seek the optimal $\phi^*$ by maximizing $P(\phi|I)$ for each dimension of $\boldsymbol{\Phi}$ independently. Note that $P(\phi|I)$ takes the form of sum of Gaussians, this can be efficiently solved by the Mean-shift algorithm [42].

### A. Tree Structure Learning

To train the Hough forest, we collect a set of depth images $\{I_m\}_{m=1}^M$, and annotate each of them with the ground truth rotation angles and center positions of hand, i.e., $\boldsymbol{\Phi}_m = (\boldsymbol{\theta}_m, \boldsymbol{v}_m)$. From each image $I_m$ we uniformly sample a fixed number of $N$ training pixels $\{p_{i,m}\}_{i=1}^N$ and associate them with their depth context descriptor $\boldsymbol{\mathcal{D}}_{i,m}$. Besides, each training pixel is annotated with the ground truth hand rotation $\boldsymbol{\theta}_{i,m} = \boldsymbol{\theta}_m$, the offset $\boldsymbol{\Delta}_{i,m}$ between the 3-D position $\boldsymbol{v}_{i,m}$ of the pixel and the ground truth palm center position $\boldsymbol{v}_m$. The forest is trained to learn a nonlinear mapping between local pixel features and hand pose parameters.

During training, each tree of the forest is built with a bootstrap subset of the annotated data. Starting from the root node, the training samples are split into two subsets recursively to reduce the prediction errors at the child nodes. At the nonleaf nodes, a set of candidate split functions $\{\psi\}$ are randomly generated as the proposals for node splitting, which takes the following form:

$$\mathcal{D}_i \le \tau \tag{5}$$

where $\mathcal{D}_i$ is a randomly selected dimension of $\boldsymbol{\mathcal{D}}$ by sampling the feature dimension index $i \in [1, \ldots, B]$, and $\tau$ is also a random threshold value to determine whether to branch to the left or right children. The optimal split function is selected to maximize a gain measure $G(\psi)$ based on hand pose distribution in the training samples that reach the node

$$\begin{aligned}\psi^* &= \arg\max_{\psi} G(\psi) \\ &= \arg\max_{\psi}\left[H(A) - \sum_{s\in\{l,r\}} \frac{|A_s(\psi)|}{|A|} H(A_s(\psi))\right]\end{aligned} \tag{6}$$

where $A$ is the set of samples reaching the current node and $A_l$ and $A_r$ are the two subsets of $A$ split by $\psi$. The function $H(A)$ is defined as the variance of the hand pose among the samples in $A$ to measure the pose uncertainty. Given the optimal split function $\psi^*$ is learned, the pixels for which $\psi^*$ takes Boolean value *true* are split into the left child branch and vice versa. Since we assume that the global hand pose is unconstrained, the different dimensions of $\boldsymbol{\Phi}$ are regarded as independent. Its variance is calculated via $H = \delta_\Phi^2 = \delta_\Delta^2 + \delta_{\theta_x}^2 + \delta_{\theta_y}^2 + \delta_{\theta_z}^2$. Here, $\delta_\Delta^2$ is the variance of the 3-D sample offsets

$$\delta_\Delta^2 = \frac{1}{|A|}\sum_{j\in A} \|\Delta_j - \bar{\Delta}\|^2 \tag{7}$$

where $\Delta_j$ is the offset associated with a sample and $\bar{\Delta}$ is the mean offset in $A$. $\delta_\theta^2$ is the variance of a rotation angle. Since the angle follows circular distribution, the circular variance $\delta_\theta^2$ is defined by [40]:

$$\delta_\theta^2 = 1 - \sqrt{\left[\frac{\sum_{j\in A}\cos\theta_j}{|A|}\right]^2 + \left[\frac{\sum_{j\in A}\sin\theta_j}{|A|}\right]^2}. \tag{8}$$

At the start, each tree is initialized with an empty root node at first, and the training samples at the root node are split into the left and right branches based on the optimal split function $\psi^*$. The samples reaching each branch are then used to construct a new tree node by either continuing the splitting procedure or ending up splitting to obtain a leaf node. This is done by checking whether certain stopping criteria are met, e.g., the pose of the samples are pure enough, or the maximum depth is reached. For each leaf node, its pose vote is represented by the mean pose in the sample set $A$ reaching it

$$\begin{aligned}\bar{\Delta} &= \frac{1}{|A|}\sum_{j\in A}\boldsymbol{\Delta}_j \\ \bar{\theta} &= \text{atan2}\left[\frac{1}{|A|}\sum_{j\in A}\sin\theta_j, \frac{1}{|A|}\sum_{j\in A}\cos\theta_j\right].\end{aligned} \tag{9}$$

Since the discriminative power of the leaf nodes are generally unequal for hand pose estimation, we use a weighting coefficient vector $\boldsymbol{w}$ to represent their importance during Hough-voting. The next section addresses the problem to seek for the optimal weighting coefficients for all the leaf nodes.

### B. Leaf Weight Optimization

According to Section III-A, the Hough forest is learned to minimize the prediction error for the sampled training pixels and focuses more on per-pixel prediction performance.

However, during the testing stage, the goal is to fuse the pose votes from the Hough-voting pixels to get more robust prediction for the query image, which lays more stress on per-frame prediction performance. Especially, these voting pixels do not contribute equally to the per-frame prediction and it is necessary to weigh their votes according to their confidence during fusion. However, the problem is not well studied in literature. In [17], a vote length threshold scheme is adopted in the problem of human pose estimation, in which a vote for certain joint position is assigned zero weight if its norm exceeds certain threshold. In [15], a variance-based weighting scheme is used for head pose estimation with Hough forest, and a leaf node in the forest is assigned a zero weight if the pose variance of training samples reaching that node exceeds certain threshold. However, they are not principled solutions, e.g., the threshold value is found by testing many different thresholds values on validation datasets to pick up the one that produces the best prediction. Besides, those votes that are filtered out by threshold may still contain useful information for prediction, e.g., contribution to fused prediction in a relatively small weight.

Based on (2), the contribution of each per-pixel pose vote to the fused prediction is represented as a linear weighting coefficient. Since the per-pixel votes are obtained at the leaf nodes of the Hough forest which are reached by each voting pixel, the weighting coefficient largely relies on the discriminative power of these leaves. This suggests a way to gap the per-pixel training objective and the per-frame testing objective. To this end, we derive a per-frame pose prediction for each training image based on the sampled training pixels used during tree-structure learning, and propose to further minimize the per-frame prediction error on the training dataset by adjusting the linear weighting coefficient $\boldsymbol{w}$ of the leaf nodes with fixed prelearned tree structures.

Fig. 5 illustrates the pipeline to learn the voting weights of the leaves. Here, we consider the case of a single tree and a single pose dimension $\phi \in \boldsymbol{\Phi}$. Let $V = \{\bar{\phi}_k, w_k\}_{k=1}^{K}$ be the set of pose votes stored at all the leaf nodes and the associated voting weights to be optimized, where $K$ is the number of leaf nodes. Basically, our goal to optimize the weights of the leaf nodes is achieved by minimizing the average per-frame prediction error on the entire training dataset

$$\begin{aligned} W^* &= \arg\min_{W} \sum_{m=1}^{M} E\Big(\tilde{\phi}_m, \phi_m\Big) + \lambda \|W\|^2 \\ &\text{s.t.} \quad W \geq 0 \end{aligned} \tag{10}$$

where $W = [w_1, \ldots, w_K]^T$ is the concatenated vector of the leaf weights, $\tilde{\phi}_m$ is the per-frame prediction, $\phi_m$ is the ground truth pose, and $E(\tilde{\phi}_m, \phi_m)$ is the prediction error on $I_m$. $W \geq 0$ enforces all the leaf weights to be non-negative. $\|W\|^2$ is a regularization term and $\lambda$ is a positive constant to weigh its significance. For hand translation parameters $\phi \in \boldsymbol{v}$, the error metric $E$ is defined as the squared difference between $\tilde{\phi}_m$ and $\phi_m$. For hand rotation parameters $\phi \in \boldsymbol{\theta}$, though, the error metric cannot be defined in this way since the angle is in non-Euclidean space. Thus, we define $E$ as the squared distance between the two points corresponding to the angles $\tilde{\phi}_m$ and $\phi_m$ on a unit circle to avoid introducing discontinuity in the error metric

$$E = \begin{cases} \left\| \tilde{\phi}_m - \phi_m \right\|^2 & \phi \in \boldsymbol{v} \\ \left\| \cos\tilde{\phi}_m - \cos\phi_m \right\|^2 + \left\| \sin\tilde{\phi}_m - \sin\phi_m \right\|^2 & \phi \in \boldsymbol{\theta}. \end{cases} \tag{11}$$

Now, we provide the derivation of per-frame predictions as a function of leaf weight vector $W$ on the training images. According to Section III-A, we uniformly sample a fixed number of $N$ pixels from each training image $I_m$ to construct the forest. For each image used in tree structure learning, we keep track of the destination leaf nodes of the training pixels sampled from it, and define its per-frame prediction as the weighted sum of the votes stored at the reached leaf nodes. As shown in Fig. 5, after the tree structure is learned, each of the $N$ pixels reaches one leaf node, and it is possible that multiple pixels from one image reach the same leaf node. This can be represented by a vector $\boldsymbol{q}_m = [q_{m,1}, \ldots, q_{m,K}]^T$, in which each element $q_{m,k}$ is the number of pixels that are sampled from the training image $I_m$ and reach the $k$th leaf node. $\boldsymbol{q}_m$ can be normalized so that its elements sum to one to simplify the following derivation. Let $\boldsymbol{\phi} = [\bar{\phi}_1, \ldots, \bar{\phi}_K]^T$ be the concatenation of all the leaf votes from $V$. For each training image $I_m$, its prediction $\tilde{\phi}_m$ can be expressed as

$$\tilde{\phi}_m = \begin{cases} (\boldsymbol{q}_m \circ \boldsymbol{\phi})^T W & \phi \in \boldsymbol{v} \\ \text{atan2}\big[(\boldsymbol{q}_m \circ \sin\boldsymbol{\phi})^T W, (\boldsymbol{q}_m \circ \cos\boldsymbol{\phi})^T W\big] & \phi \in \boldsymbol{\theta} \end{cases} \tag{12}$$

where $\circ$ denotes the Hadamard product and $\sin\boldsymbol{\phi}$ and $\cos\boldsymbol{\phi}$ calculate the element-wise sine and cosine values for the rotation angles in $\boldsymbol{\phi}$. By combining (11) and (12) we show the optimization problem (10) can be converted to a standard bound-constrained least squares problem.

Here, we define three auxiliary matrices $Q, A \in \mathbb{R}^{M\times K}$, and $B \in \mathbb{R}^M$ to express $E$ as a quadratic form of $W$. $Q = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_M]^T$ records the hits on the leaf nodes of the forest for all the training images. $A = [\boldsymbol{\phi}, \ldots, \boldsymbol{\phi}]^T$ stacks the vector $\boldsymbol{\phi}$ for $M$ repetitive rows. $B = [\phi_1, \ldots, \phi_M]^T$ stores the ground truth pose of the training images. First, consider the translation parameters $\phi \in \boldsymbol{v}$. Let $A_Q = Q \circ A$. The error term $\sum_{m=1}^{M} E(\tilde{\phi}_m, \phi_m)$ in (10) can be rewritten as $\|A_Q W - B\|^2$. Thus, we have

$$\begin{aligned} W^* &= \arg\min_{W} W^T A_Q^T A_Q W - 2B^T A_Q W + \lambda \|W\|^2 \\ &= \arg\min_{W} W^T \Big(A_Q^T A_Q + \lambda I\Big) W - 2B^T A_Q W \\ &\text{s.t.} \quad W \geq 0. \end{aligned} \tag{13}$$

We can see the task to find $W^*$ is now a standard constrained quadratic programming problem. Similarly, the derivation holds for the rotation parameters $\phi \in \boldsymbol{\theta}$, except that the matrices $Q$, $A$, and $B$ need to include both the cosine and sine parts based on the error metric for rotation angles in (11). Specifically, $Q' = [Q^T, Q^T]^T$ stacks the matrix $Q$ for twice; $A' = [\cos A^T, \sin A^T]^T$ stacks both the element-wise cosine and sine values of $A$. $B' = [\cos B^T, \sin B^T]^T$ stores the

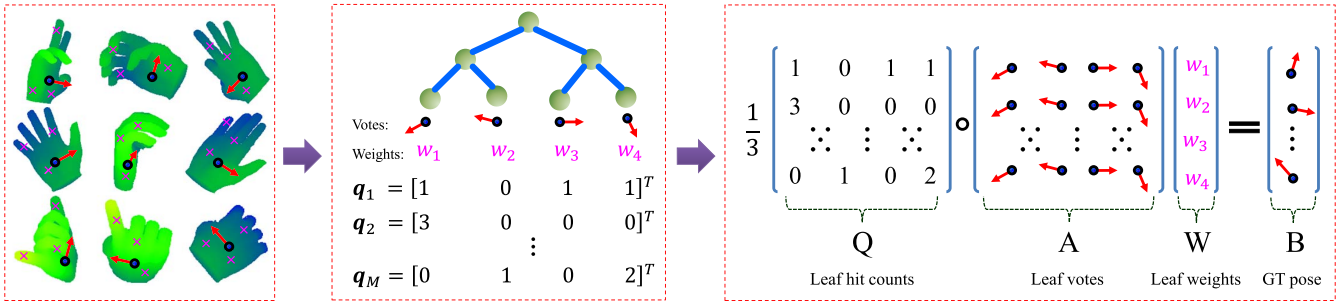


Fig. 5. Example to illustrate leaf weight optimization for a single rotation parameter. The red arrows denote the rotation angles. In this simple example three pixels are sampled from each training image, which are denoted as pink crosses. $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_M$ records the hit counts of the sampled training voting pixels on the leaf nodes for each training image. The objective of leaf weight learning is to minimize the sum of per-frame prediction error on the training dataset.

element-wise cosine and sine values for the ground truth rotation angles. Let $A'_Q = Q' \circ A'$ and substitute $A'_Q$ and $B'$ into (13), and the problem to find $W^*$ for $\phi \in \boldsymbol{\theta}$ is also converted to a constrained quadratic programming problem.

In general, the numbers of the training images and the leaf nodes in the regression forest are very huge, and the size of the matrix $A_Q$ is very huge, i.e., at the magnitude of $10^4 \times 10^4$. However, since the number of training pixels per image is relatively small, i.e., several hundreds, and usually many of the pixels from the same image also go to the same leaf node, $A_Q$ is actually very sparse. Therefore, we utilize the interior-point-convex quadric programming algorithm [43] to solve (13), which usually takes less than one hour to converge for a single dimension $\phi \in \boldsymbol{\Phi}$ in a single tree.

## IV. Experimental Results

This section presents the experiments of global hand pose estimation on a synthesized dataset, a real-world dataset and real-time video sequences. The synthesized dataset is obtained by driving a 3-D hand model [20] with various finger articulation and hand rotation parameters to generate synthesize hand depth images with computer graphic methods, which thus contain accurate global hand pose annotations. In addition, the proposed method is also evaluated on the real-world hand pose dataset [36] to demonstrate its capability to handle noisy real depth data. Finally, we use the SoftKinetic DS325 and Intel Realsense depth camera to capture real-world hand depth videos and compare the proposed method to state-of-the-art hand tracking solutions from industry and academy.

In all the tests in this section, the resolution of the images is 320 × 240. The forests consist of three trees, which follows previous work on Hough forest-based hand pose estimation [18]. To train each tree of the Hough forest, 200 pixels are uniformly sampled from each training image and 10 000 candidate split functions are generated. During testing, 1000 pixels are uniformly sampled from the hand region for spatial voting. All evaluated methods were coded in C++/OpenCV except that the leaf weight optimization algorithm in Section III-B was implemented in MATLAB. Training is performed on a workstation with Intel Xeon E5620 2.4 GHz CPU and 32 GB RAM. The online tests are performed on a PC with Intel i5 3.2 GHz CPU without parallelization.

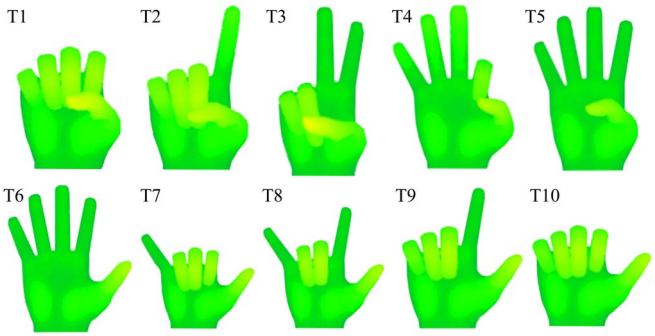


Fig. 6. Basic hand posture templates to generate the synthesis dataset.

### A. Quantitative Evaluation on Synthesis Dataset

In this experiment, we test the performance of the proposed Hough forest on a synthesis dataset. The 3-D hand model [20] to generate this dataset consists of a skeleton system and a skin surface mesh to simulate the real hand. The skeleton is modeled as kinematic chains of bones in a tree structure with the root at the wrist. The skin mesh consists of 7000 triangles. The palm center is set as the central point on the middle Metacarpal bone on the model skeleton. Give a hypothesized hand pose, the skeleton is updated via forward kinematics, and skin mesh is then updated via skeleton subspace deformation. The hand poses for data synthesis contain variations of both global hand rotation and local finger articulation. The global hand rotation is confined within $(-20°, 60°)$ around the $x$-axis, $(-180°, 30°)$ around $y$-axis and $(-60°, 60°)$ around the $z$-axis, which have covered most of the natural hand rotation ranges. It is uniformly discretized into 864 viewpoints. The definition of rotation axis can be referred to Fig. 3. The local finger articulation consists of ten basic hand posture templates, as illustrated in Fig. 6. For each of the ten basic templates, the fingers are also allowed to move in small ranges, which give in total 60 static hand postures. Overall, the number of images in this dataset is around 50 k, and each of them is annotated with ground truth global hand pose parameters.

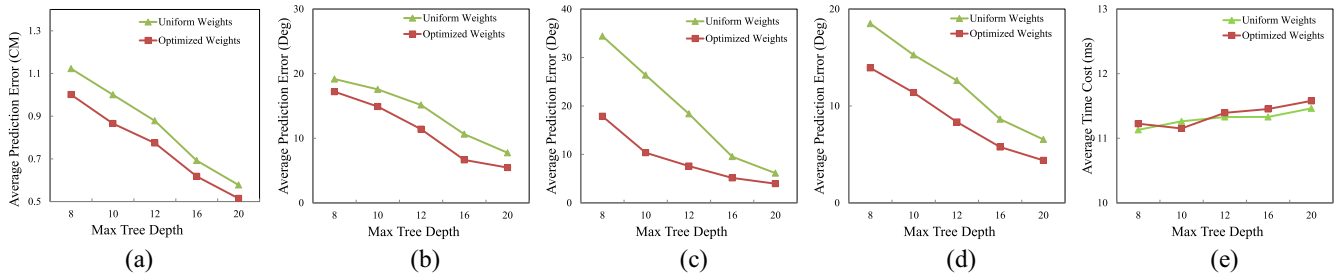


Fig. 7. Hand pose estimation results on the synthesis dataset for different values of maximum tree depth. (a) Translation $\boldsymbol{\nu}$ prediction. (b) Pitch angle $\theta_x$ prediction. (c) Yaw angle $\theta_y$ prediction. (d) Roll angle $\theta_z$ prediction. (e) Avg. per-frame time cost.

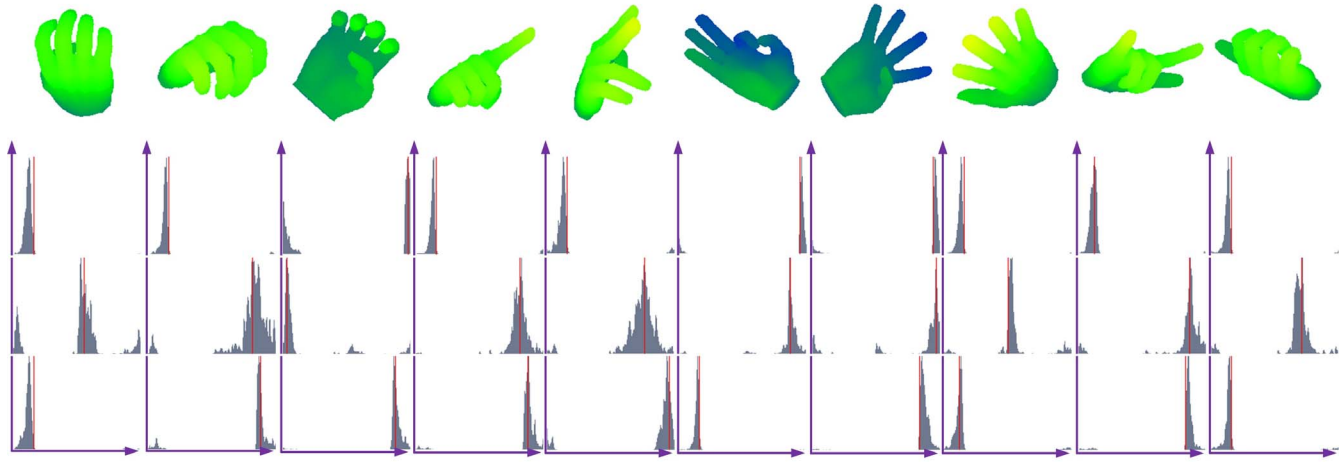

Fig. 8. Illustration of vote distribution histograms obtained with optimized leaf weights within $[0, 360°]$ for the three rotation angles, where the ground truth angles are illustrated with red lines. First row: input depth images. Second row: $\theta_x$ prediction. Third row: $\theta_y$ prediction. Fourth row: $\theta_z$ prediction.

The experimental results are based on single-frame evaluation as the synthesis images are generated independently. To evaluate the prediction accuracy, we perform fivefold cross validation, and in each fold 80% of all the images are used for forest training and the rest 20% for testing. For 3-D hand translation, the prediction accuracy is evaluated in terms of the average distance between the prediction and the ground truth. For 3-D hand rotation angles we follow the conventions [44] to define the prediction error between prediction $\tilde{\phi}$ and ground truth $\phi$ as their absolute difference:

$$D(\tilde{\phi}, \phi) = \left| (\tilde{\phi} - \phi) \mod \pm 180° \right|. \tag{14}$$

Similarly, the prediction performance is evaluated in terms of the average angle error between the prediction and the ground truth. To perform the experiment, we use the training data to learn the tree structures for several different values of maximum tree depth ranging within [8, 20] to better understand the performance of our leaf weight optimization algorithm with respect to different numbers of leaf nodes. Table I provides the average number of leaf nodes per-tree and the corresponding times costs for Hough forest training and leaf weight optimization during the training stage.

In Fig. 7, we compare hand pose estimation results obtained by optimized and uniform leaf weights, which show the prediction errors for 3-D translation and rotation angles and the average per-frame time costs for different tree depths of the forest. Overall, the proposed leaf weight optimization algorithm largely improves the prediction accuracy for both 3-D hand translation and rotation compared to uniform leaf weights. For hand translation, the proposed leaf weight optimization scheme reduces the prediction error by 11.6% on average. For hand rotation, even with relatively low tree depth, e.g., 8, the prediction errors with optimized leaf weights are only 17.2°, 17.9°, and 13.9° for the three rotation angles, which is even better than the performance of the depth-10 forest with uniform leaf weights. Particularly, for the $\theta_y$ rotation which ranges between $(-180°, 30°)$ and has the largest appearance ambiguity, the forest with optimized leaf weights reduces the prediction error by 49.8% on average. In Fig. 7(e), we also present the average per-frame time costs for hand pose prediction, which shows that the proposed method does not require extra time costs compared to the forest with uniform leaf weights. The time costs do not change much with different tree depths, and are below 12 ms on average, which

TABLE I
AVERAGE NUMBER OF LEAF NODES PER-TREE AND THE CORRESPONDING TRAINING TIME COSTS WITH RESPECT TO DIFFERENT TREE DEPTHS. FT: FOREST TRAINING. LWO: LEAF WEIGHT OPTIMIZATION. H: HOUR. M: MINUTE. S: SECOND

| Tree Depth | 8 | 10 | 12 | 16 | 20 |
|---|---|---|---|---|---|
| Leaf Node Number | 254 | 924 | 2526 | 10422 | 20951 |
| FT Time Cost | 3h05m | 4h03m | 4h31m | 5h43m | 7h27m |
| LWO Time Cost | 51s | 2m40s | 10m38s | 1h34m | 12h30m |

TABLE II
AVERAGE PREDICTION ERRORS FOR HAND TRANSLATION (CM) AND ROTATION (DEGREE) FOR EACH BASIC HAND TEMPLATE IN SYNTHESIZED DATASET

| | Uniform Leaf Weights | | | | Optimized Leaf Weights | | | |
|---|---|---|---|---|---|---|---|---|
| | $\boldsymbol{v}$ (cm) | $\theta_x(\circ)$ | $\theta_y(\circ)$ | $\theta_z(\circ)$ | $\boldsymbol{v}$ (cm) | $\theta_x(\circ)$ | $\theta_y(\circ)$ | $\theta_z(\circ)$ |
| T1 | 0.61 | 8.76 | 7.91 | 8.91 | 0.61 | 4.68 | 5.85 | 5.64 |
| T2 | 0.48 | 6.24 | 6.67 | 8.73 | 0.45 | 4.47 | 5.90 | 4.12 |
| T3 | 0.51 | 5.78 | 5.47 | 7.37 | 0.45 | 4.09 | 4.83 | 4.08 |
| T4 | 0.75 | 7.14 | 5.38 | 7.50 | 0.61 | 4.16 | 5.10 | 5.12 |
| T5 | 0.95 | 5.49 | 7.94 | 7.55 | 0.73 | 4.00 | 5.36 | 4.12 |
| T6 | 0.86 | 5.84 | 5.06 | 7.73 | 0.68 | 4.06 | 5.52 | 4.19 |
| T7 | 0.39 | 7.78 | 6.74 | 8.11 | 0.40 | 4.42 | 5.21 | 4.39 |
| T8 | 0.36 | 5.80 | 5.24 | 6.96 | 0.35 | 4.09 | 4.48 | 3.72 |
| T9 | 0.36 | 5.74 | 5.47 | 7.11 | 0.36 | 4.31 | 4.53 | 3.75 |
| T10 | 0.39 | 7.64 | 6.28 | 8.04 | 0.40 | 4.44 | 5.17 | 4.63 |
| Avg. | 0.57 | 6.62 | 6.21 | 7.80 | 0.51 | 4.27 | 5.20 | 4.38 |
| Std Dev | 0.22 | 1.13 | 1.08 | 0.65 | 0.14 | 0.23 | 0.49 | 0.60 |

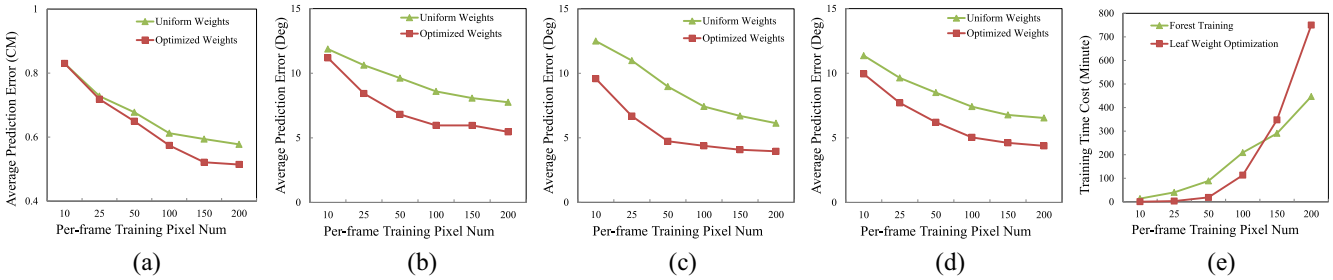


Fig. 9. Hyper-parameter tests for the number of per-frame training pixels $N$. (a) Translation $\boldsymbol{v}$ prediction. (b) Pitch angle $\theta_x$ prediction. (c) Yaw angle $\theta_y$ prediction. (d) Roll angle $\theta_z$ prediction. (e) Training time cost.

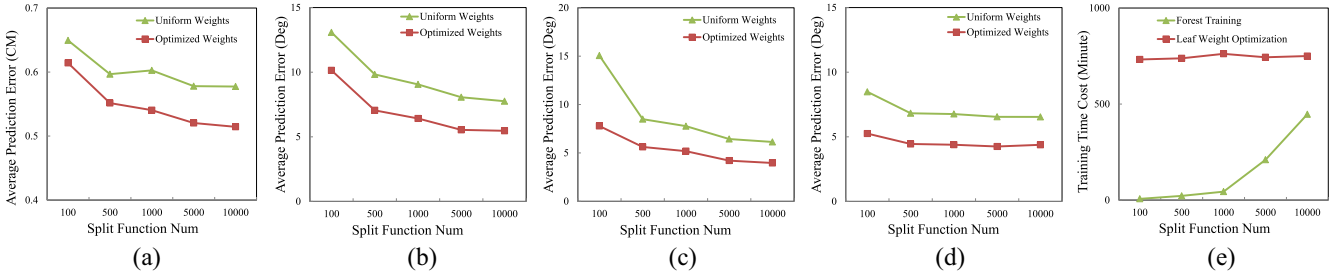


Fig. 10. Hyper-parameter tests for the number of candidate split functions. (a) Translation $\boldsymbol{v}$ prediction. (b) Pitch angle $\theta_x$ prediction. (c) Yaw angle $\theta_y$ prediction. (d) Roll angle $\theta_z$ prediction. (e) Training time cost.

is sufficient for most real-time applications. Fig. 8 illustrates the vote distribution histograms with optimized leaf weights, in which ground truth rotation angles are shown in red lines. The vote distribution histograms are obtained by first discretizing angle range $0°, 360°$ into 180 bins and then counting the weighted number of voting pixels that falls into each bin of the histograms. That is, if the angle vote of a voting pixel falls into a certain bin of the histogram, the bin value is then increased by the voting weight of the pixel. This process is repeated for all the voting pixels to obtain the final histogram. As the rotation angle is periodic with 360°, we round the negative angle values into $[0, 360°]$ by adding 360° to them to simplify drawing the histograms. We can see that the distributions with optimized weights generally concentrate on ground truths.

As ten basic hand posture templates are used to synthesize this dataset, in Table II, we provide the pose prediction results for each of them separately for tree depth of 20. The average and standard deviation are also calculated for prediction errors of the ten templates to evaluate the capability of consistent prediction for different hand shapes. As shown in Table II, the proposed leaf weight optimization method not only reduces pose prediction error but also produces consistent predictions for different postures. Again, it is worth noting that the proposed method can work for hand pose estimation with arbitrary postures and is not limited to these ten templates.

We have also investigated influence of certain hyper-parameters on system performance, including the numbers of per-frame training pixels $N$ and candidate split functions. The results are illustrated in Figs. 9 and 10. For each test, the

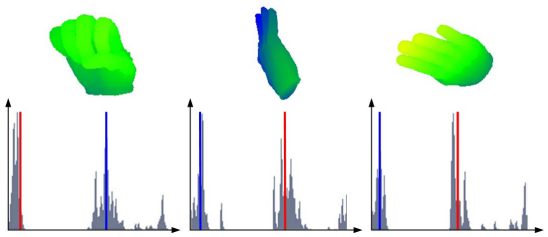

Fig. 11. Failure cases for yaw rotation prediction. Red lines indicate ground truth and blue lines indicate predicted pose.

uninvestigated parameters take the same values as in the beginning of Section IV. Meanwhile, despite that the proposed leaf weight optimization algorithm improves hand pose prediction accuracy considerably, there are still cases in which the hand appearance poses ambiguity for prediction, as in Fig. 11. In these examples the hand looks very similar from several different perspectives, e.g., frontal and back views.

### *B. Quantitative Evaluation on Real Dataset*

The experiments on real-world data is performed on MSRA Hand Gesture database [36], which includes totally 76 k annotated depth images from nine different subjects. There are some other datasets such as NYU and ICVL hand pose datasets [4], [6], but the hand rotation range in their testing images is relatively small compared to [36]. Thus, they are more suitable for full-DOF hand pose estimation rather than 3-D hand rotation estimation. In contrast, the MSRA dataset contains hand images in larger hand rotation range, as illustrated in Fig. 12. As the annotations in MSRA dataset are the 3-D positions of 21 hand joints in each image as in left of Fig. 13, we need to convert them to 3-D hand translation and rotation for evaluation. To this end, we define the 3-D hand translation as the wrist position, i.e., joint 1 in Fig. 13. To calculate the 3-D rotation angles, we define the rotated $y$-axis of hand to be along the vector pointing from wrist to the middle metacapophalangeal joint (joint 10) as in right of Fig. 13. Let this normalized vector be $\vec{\boldsymbol{y}}$. We then calculate three normalized vectors pointing from hand wrist to the middle metacapophalangeal joints of pinky (joint 18), ring (joint 14), and index (joint 6) fingers, i.e., $\vec{\boldsymbol{\alpha}}_1$, $\vec{\boldsymbol{\alpha}}_2$, and $\vec{\boldsymbol{\alpha}}_3$. The rotated $z$-axis of hand is taken to be the average of the cross products $\vec{\boldsymbol{y}} \times \vec{\boldsymbol{\alpha}}_1$, $\vec{\boldsymbol{y}} \times \vec{\boldsymbol{\alpha}}_2$, and $\vec{\boldsymbol{\alpha}}_3 \times \vec{\boldsymbol{y}}$ to reduce the influence of annotation noise. The rotated $x$-axis is obtained by applying orthogonality, and the 3-D Euler rotation angles are calculated based on the rotated hand coordinate system.

We further implemented the single-view CNN-based method [6] with the Torch7 [45] framework for comparison. The method in [6] first adopts CNN to regress for the 2-D positions of hand joints, and then applies a generative model-fitting phase to refines hand pose prediction from CNN. For fair comparison, we only implemented their CNN regression framework and compare their results to the proposed method. Specifically, we convert the estimated 2-D hand joint positions to 3-D positions by taking the depth of a joint to be the value at the predicted 2-D position in the depth

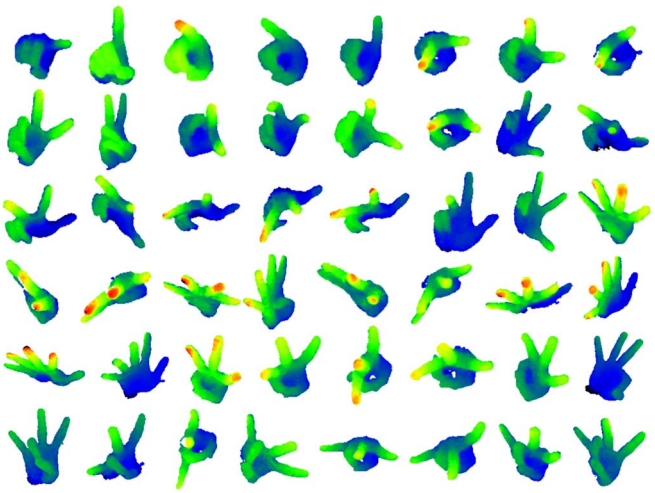

Fig. 12. Exemplar images in MSRA Hand Gesture database [36].

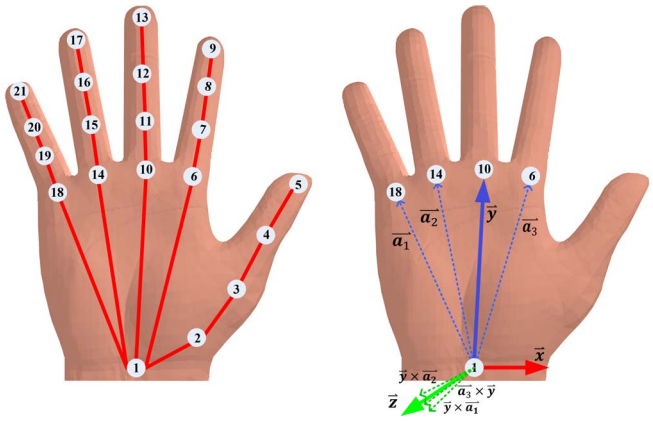


Fig. 13. Left: annotated hand joint positions in MSRA Hand Gesture database [36]. Right: conversion of joint annotation to Euler angles of 3-D hand rotation.

images. The 3-D hand joint position can be obtained by camera back-projection based on 2-D hand joint position and depth value. With the 3-D positions of all hand joints, the 3-D hand translation and rotation angles are calculated via the process in right of Fig. 13. In addition, we also compare the proposed method to the state-of-the-art multiview CNNs algorithm [8], which directly predicts 3-D positions of hand joints. Similarly, its prediction can be converted to global hand pose following Fig. 13. Both methods [6], [8] are tested on a GPU workstation with two Nvidia Tesla K20 GPUs.

Table III presents the pose prediction errors, per-frame time costs, and trained model sizes obtained by Hough forests with both uniform and optimized leaf weights and the CNN-based method [6], [8]. The results are obtained by training Hough forest and CNN on eight of the subjects in MSRA Hand Gesture database [36] and testing on the remaining one. It again shows that the optimized leaf weights reduce hand pose prediction error, particularly for hand rotation angles. Besides, the proposed method largely outperforms single-view CNN method [6], and has comparable accuracy with the multiview CNN method [8]. It is worth noting that our

TABLE III
HAND POSE PREDICTION ERROR ON REAL-DATASET USING THE FORESTS WITH UNIFORM AND OPTIMIZED LEAF WEIGHTS AND THE SINGLE-VIEW CNN METHOD [6] AND MULTIVIEW CNNS METHOD [8]

| | $\boldsymbol{v}$ (cm) | $\theta_x(\circ)$ | $\theta_y(\circ)$ | $\theta_z(\circ)$ | Avg. Angle Error $(\circ)$ | Time Cost | Model Size |
|---|---|---|---|---|---|---|---|
| HF + Uniform weights | 1.47 | 11.99 | 9.23 | 8.05 | 9.76 | 11.3ms | 16MByte |
| HF + Optimized weights | 1.42 | **8.63** | **7.64** | 7.23 | **7.83** | 11.2ms | 16MByte |
| Single-View CNN [6] | 1.93 | 22.35 | 19.18 | 18.64 | 20.06 | 7.2ms | 378MByte |
| Multi-View CNNs [8] | **1.34** | 9.14 | 7.69 | **6.75** | 7.86 | 14.1ms | 1.1GByte |

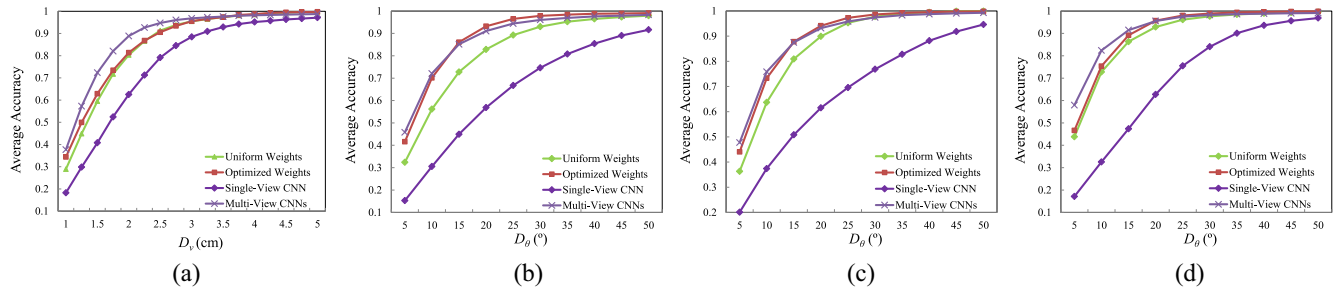


Fig. 14. Hand pose prediction accuracy with respect to different tolerance $D_v$ and $D_\theta$ on MSRA Hand Gesture database [36]. (a) Translation $\boldsymbol{v}$ prediction. (b) Pitch angle $\theta_x$ prediction. (c) Yaw angle $\theta_y$ prediction. (d) Roll angle $\theta_z$ prediction.

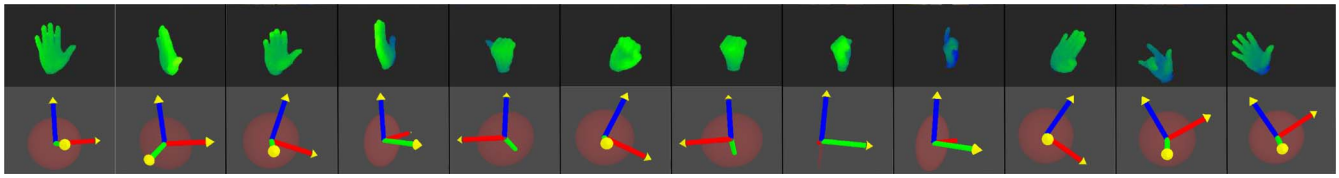

Fig. 15. Real-time hand pose estimation of the proposed algorithm with a DS325 camera. Row 1: input depth images. Row 2: tracked hand pose.

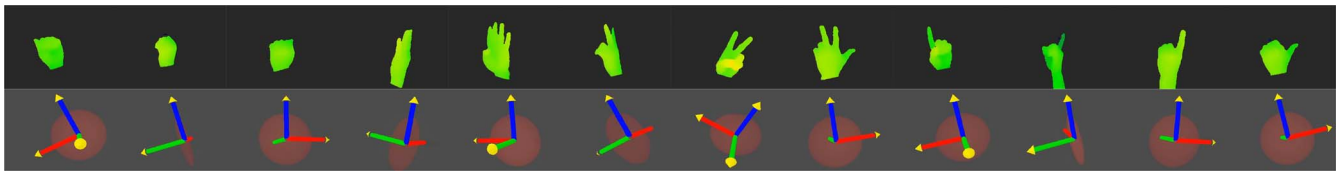

Fig. 16. Real-time hand pose estimation of the proposed algorithm with an Intel Realsense camera. Row 1: input depth images. Row 2: tracked hand pose.

method requires only CPU in run-time, while both CNN methods require GPU support. Without any parallelization, the proposed Hough forest can achieve similar time-performance to CNN methods that heavily rely on GPU for acceleration. Moreover, as CNN needs a large number of network parameters to store the convolutional layers for feature extraction and the full-connected layers for prediction, the resulting model size is generally quite large compared to Hough forest. As shown in Table III, our implementation of the single-view CNN [6] with their network structure requires around 378 MB storage and the multiview CNNs algorithm [8] requires 1.1 GB storage. In contrast, the proposed Hough forest only occupies 16 MB storage. Thus, it is more flexible for different computation platforms, such as Microsoft Hololens and Oculus Rift. Fig. 14 provides the overall prediction accuracy with respect to different values of tolerance $D_v$ and $D_\theta$, i.e., the percentage of predictions with errors less than $D_v$ for translation and $D_\theta$ for rotation. The results are consistent to Table III.

### C. Qualitative Comparison With State-of-the-Arts

In this section, we utilize the forests trained on the synthesis data in Section IV-A and qualitatively test them on real-world depth image sequences from a SoftKinetic DS325 camera and an Intel Realsense camera. The hand region is segmented in the depth images with the SDKs provided by these two cameras. For each dimension of 6-D global hand pose, a Kalman filter is utilized to smooth its prediction independently in successive frames both in Sections IV-C and IV-D. We illustrate the predicted global hand pose in terms of transformed 3-D coordinate, and the results obtained with the DS325 camera and Intel Realsense camera on several exemplar frames are provided in Figs. 15 and 16. Note that the colors of the coordinate axes and the definition of zero rotation are in line with that in Fig. 3. The positive direction of each axis is indicated with the yellow cap. The results show that the predicted hand pose obtained by the proposed method is visually very consistent to real pose.

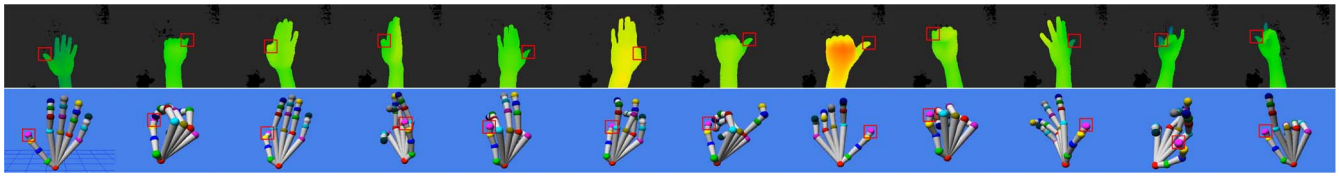

Fig. 17. Hand pose estimation results of [9] with Intel Realsense camera input. First row: input depth images. Second row: estimated hand pose. The red rectangles denote the positions of thumb fingertip in both the input and reconstructed hand model for reference, and the hand rotation prediction is wrong when the thumb positions are on different sides of the hand.

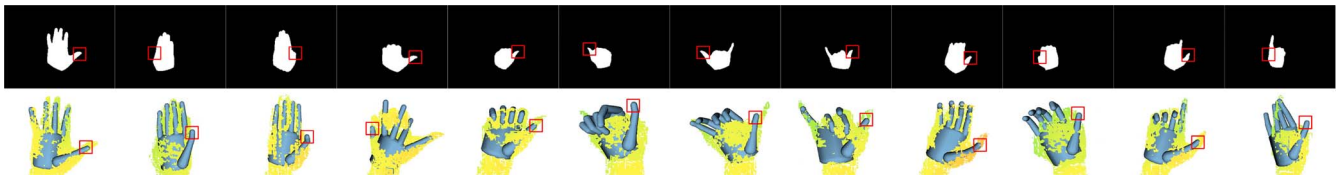

Fig. 18. Hand pose estimation results of [5] with DS325 camera input. First row: input depth images (the released runnable file only illustrates the binary hand mask during running). Second row: estimated hand pose. The red rectangles denote the positions of thumb fingertip in both the input and reconstructed hand model for reference, and the hand rotation prediction is wrong when the thumb positions are on different sides of the hand.

For qualitative comparison, we test the state-of-the-art Intel Hand Tracking Library [9] from industry and the Articulated-ICP method [5] from academia, and the user performs similar postures as that in Figs. 15 and 16. Both methods provide full-DOF hand tracking functionality. Since 6-DOF global hand pose is an intrinsic subset of full-DOF hand pose parameters, we focus on their 3-D hand translation and rotation prediction performance in this test. The former is used with the Intel Realsense camera and the latter is used with the DS325 camera due to the requirement of the SDKs. Figs. 17 and 18 present some exemplar frames of the results obtained with [5] and [9]. Overall, for postures that have a distinctive pattern, such as fully stretched hand postures, these two SDKs perform relatively well. However, for ambiguous postures, such as fist posture or when all fingers are side-by-side, they usually cannot predict the correct hand pose well and can get confused between the back and frontal viewpoints. In contrast, the proposed algorithm with optimized leaf weights can predict hand pose much more robustly against rapidly changing hand postures and rotation angles. Even for postures like a rotating fist, the hand pose can still be predicted accurately. The detailed video results for the proposed algorithm with optimized leaf weights on Intel/DS325 cameras and that of [5] and [9] can be found on our website.[1]

### D. Applications

This section presents two virtual/augmented reality applications based on the proposed hand pose estimation algorithm, which allow users to manipulate virtual objects with bare hands. For command inputs we implement the shape classification forest in [46] to recognize user's hand gestures. As in [46], during training, a number of pixels are sampled from each training image and associated with the corresponding depth context features, and each pixel takes the same gesture label to the image that they are from. The forest is trained to minimize the entropy of gesture label distributions for intermediate node splitting. At the leaf node, we store a histogram to represent the probabilistic gesture class distribution. During testing, a number of $N_g$ gesture voting pixels retrieve a set of gesture votes $\{H_{ij}\}_{j=1}^{T}$, where $H_{ij}$ is the gesture distribution histogram retrieved by pixel $i$ from tree $j$ and $H_{ij}(l)$ represents the probability that the gesture prediction is $l$. The optimal hand gesture $l^*$ is predicted by

$$l^* = \arg\max_{l} \frac{1}{N_g \times T} \sum_{i,j} H_{ij}(l). \tag{15}$$

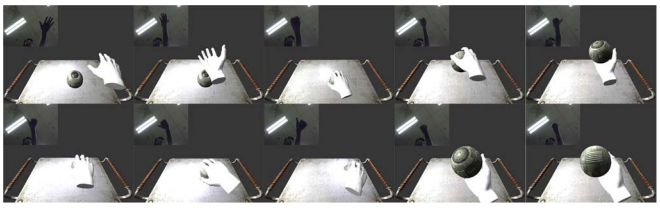

Fig. 19. Virtual object manipulation via global hand pose estimation and gesture recognition. The up-left corner of each frame shows the input image.

The first application is virtual object manipulation in virtual reality, in which a user can grasp a virtual ball and examine it from different perspectives. The shape classification forest recognizes open and close hand gestures to allow the grasp action. We adopt the grasping control pipeline in [47] to map the discrete open/close hand states to continuous grasping motion. During runtime, the system checks the states of grasping when the 6-D hand pose and gesture are recovered. In case that a close hand gesture is detected, i.e., grasping motion, it performs collision detection between the virtual hand and the virtual ball. If collision occurs, the ball moves and rotates following the 6-D hand motion. The object is released if an open hand gesture is detected after a successful grasping. Fig. 19 illustrates the scenario and a grasping example.

The second application is an augmented reality system so that a virtual teapot is visually put upon the real hand in RGB images and three hand gestures are used to change the teapot color, as shown in Fig. 20. This application is publicly demoed

[1] https://sites.google.com/site/seraphlh

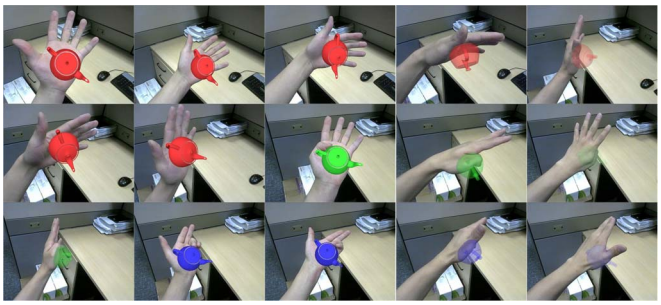

Fig. 20. Teapot inspection and color selection via global hand pose estimation and gesture recognition.

in real-time at ACM MM15 [48]. The SoftKinetic DS325 camera is used to capture both RGB and depth images of user's hand and the global hand motion and gesture information are predicted from depth images. The depth and color sensors of the DS325 camera are calibrated in advance to transform the global hand pose recovered from the depth to the coordinate system centered at the color sensor. The virtual teapot is then rendered according to the transformed hand pose and gesture, and projected onto the image plane with OpenGL, which is then overlaid on the RGB image. To get realistic visual feedback, we define a visibility term for the teapot based on hand rotation angles to reflect hand-object occlusion, which is implemented via controlling the transparency effect in OpenGL with the roll angle of hand. That is, the teapot is fully opaque when the palm is facing the camera, and becomes gradually transparent when it rotates backwards. The whole video for both applications can be found on our website.[1]

## V. Conclusion

In this paper, we present a Hough forest-based algorithm to estimate global hand motion with arbitrary postures. To improve per-frame pose prediction in the Hough-voting scheme, we propose to optimize the voting weights of leaf nodes so that the average error between the ground truth hand pose and the fused predictions of the training images is minimized. The task is formulated as a constrained quadratic programming problem and can be solved efficiently. The method is tested on both a synthesis dataset and a real-world dataset containing large viewpoint variations and various finger motions, and improves prediction accuracy considerably compared to rivals. Comparison to state-of-the-art hand tracking systems also demonstrates the effectiveness of the proposed algorithm. Based on these techniques, we develop two real-time HCI applications to allow users to manipulate virtual objects with bare hands. The proposed algorithm still has large potential to exploit in the areas of visualization and HCI, and we plan to integrate it with real-world applications, such as virtual mechanical part manipulation in computer-aided design.

## References


[1] M. A. A. Aziz, J. Niu, X. Zhao, and X. Li, "Efficient and robust learning for sustainable and reacquisition-enabled hand tracking," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 945–958, Apr. 2016.

[2] S. Poularakis and I. Katsavounidis, "Low-complexity hand gesture recognition system for continuous streams of digits and letters," *IEEE Trans. Cybern.*, vol. 46, no. 9, pp. 2094–2108, Sep. 2016.

[3] C. Xu and L. Cheng, "Efficient hand pose estimation from a single depth image," in *Proc. Int. Conf. Comput. Vis.*, Sydney, NSW, Australia, 2013, pp. 3456–3462.

[4] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim, "Latent regression forest: Structured estimation of 3D articulated hand posture," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 3786–3793.

[5] A. Tagliasacchi *et al.*, "Robust articulated-ICP for real-time hand tracking," in *Proc. Eurograph. Symp. Geometry Process.*, Graz, Austria, 2015, pp. 101–114.

[6] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Trans. Graph.*, vol. 33, no. 5, p. 169, 2014.

[7] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands deep in deep learning for hand pose estimation," in *Proc. Comput. Vis. Win. Workshop*, 2015, pp. 21–30.

[8] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Robust 3D hand pose estimation in single depth images: From single-view CNN to multi-view CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 3593–3601.

[9] *The Intel Skeletal Hand Tracking Library*. Accessed: Oct. 15, 2015. [Online]. Available: http://www.intel.com

[10] M. Sun, P. Kohli, and J. Shotton, "Conditional regression forests for human pose estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, 2012, pp. 3394–3401.

[11] P. Song, W. B. Goh, W. Hutama, C.-W. Fu, and X. Liu, "A handle bar metaphor for virtual object manipulation with mid-air interaction," in *Proc. ACM Conf. Human Factors Comput. Syst.*, Austin, TX, USA, 2012, pp. 1297–1306.

[12] R. Y. Wang, S. Paris, and J. Popović, "6D hands: Markerless hand-tracking for computer aided design," in *Proc. ACM Symp. User Interface Softw. Technol.*, Santa Barbara, CA, USA, 2011, pp. 549–558.

[13] M. Schlattmann, F. Kahlesz, R. Sarlette, and R. Klein, "Markerless 4 gestures 6 DOF real-time visual tracking of the human hand with automatic initialization," *Comput. Graphics Forum*, vol. 26, no. 3, pp. 467–476, 2007.

[14] Q. Ji, "3D face pose estimation and tracking from a monocular camera," *Image Vis. Comput.*, vol. 20, no. 7, pp. 499–511, 2002.

[15] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. V. Gool, "Random forests for real time 3D face analysis," *Int. J. Comput. Vis.*, vol. 101, no. 3, pp. 437–458, Feb. 2013.

[16] A. Yao, J. Gall, and L. V. Gool, "A hough transform-based voting framework for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, 2010, pp. 2061–2068.

[17] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 415–422.

[18] D. Tang, T.-H. Yu, and T.-K. Kim, "Real-time articulated hand pose estimation using semi-supervised transductive regression forests," in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 3224–3231.

[19] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[20] H. Liang, J. Yuan, and D. Thalmann, "Parsing the hand in depth images," *IEEE Trans. Multimedia*, vol. 16, no. 5, pp. 1241–1253, Aug. 2014.

[21] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013.

[22] E. Ueda, Y. Matsumoto, M. Imai, and T. Ogasawara, "Hand pose estimation using multi-viewpoint silhouette images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2001, pp. 1989–1996.

[23] M. de La Gorce, D. J. Fleet, and N. Paragios, "Model-based 3D hand pose estimation from monocular video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1793–1805, Sep. 2011.

[24] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3D tracking of hand articulations using kinect," in *Proc. Brit. Mach. Vis. Conf.*, Dundee, U.K., 2011, pp. 1–11.

[25] S. Pellegrini, K. Schindler, and D. Nardi, "A generalisation of the ICP algorithm for articulated bodies," in *Proc. Brit. Mach. Vis. Conf.*, Leeds, U.K., 2008, pp. 1–10.

[26] M. Schröder, J. Maycock, H. Ritter, and M. Botsch, "Real-time hand tracking using synergistic inverse kinematics," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, 2014, pp. 5447–5454.

[27] L. Ballan, A. Taneja, J. Gall, L. V. Gool, and M. Pollefeys, "Motion capture of hands in action using discriminative salient points," in *Proc. Eur. Conf. Comput. Vis.*, Florence, Italy, 2012, pp. 640–653.
[28] S. Sridhar, A. Oulasvirta, and C. Theobalt, "Interactive marker-less articulated hand motion tracking using RGB and depth data," in *Proc. Int. Conf. Comput. Vis.*, Sydney, NSW, Australia, 2013, pp. 2456–2463.
[29] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 1106–1113.
[30] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Trans. Graph.*, vol. 31, no. 6, p. 188, 2012.
[31] A. Thayananthan, R. Navaratnam, B. Stenger, P. H. S. Torr, and R. Cipolla, "Pose estimation and tracking using multivariate regression," *Pattern Recognit. Lett.*, vol. 29, no. 9, pp. 1302–1310, 2008.
[32] H. Guan, R. S. Feris, and M. Turk, "The isometric self-organizing map for 3D hand pose estimation," in *Proc. Int. Conf. Autom. Face Gesture Recognit.*, Southampton, U.K., 2006, pp. 263–268.
[33] H. Guan, J. S. Chang, L. Chen, R. S. Feris, and M. Turk, "Multi-view appearance-based 3D hand pose estimation," in *Proc. Comput. Vis. Pattern Recognit. Workshop*, New York, NY, USA, 2006, p. 154.
[34] J. Romero, H. Kjellström, and D. Kragic, "Monocular real-time 3D articulated hand pose estimation," in *Proc. Int. Conf. Humanoid Robots*, Paris, France, 2009, pp. 87–92.
[35] M. Potamias and V. Athitsos, "Nearest neighbor search methods for handshape recognition," in *Proc. Int. Conf. Pervasive Tech. Related Assistive Environ.*, Athens, Greece, 2008, Art. no. 30.
[36] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 824–832.
[37] H. Liang, J. Yuan, and D. Thalmann, "Resolving ambiguous hand pose predictions by exploiting part correlations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 7, pp. 1125–1139, Jul. 2015.
[38] F. Kirac, Y. E. Kara, and L. Akarun, "Hierarchically constrained 3D hand pose estimation using regression forests from single frame depth data," *Pattern Recognit. Lett.*, vol. 50, pp. 91–100, Dec. 2013.
[39] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei, "Model-based deep hand pose estimation," in *Proc. Int. Joint Conf. Artif. Intell.*, New York, NY, USA, 2016, pp. 2421–2427.
[40] N. I. Fisher, *Statistical Analysis of Circular Data*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[41] C. Herdtweck and C. Curio, "Monocular car viewpoint estimation with circular regression forests," in *Proc. IEEE Intell. Veh. Symp.*, Gold Coast, QLD, Australia, 2013, pp. 403–410.
[42] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.
[43] N. Gould and P. L. Toint, "Preprocessing for quadratic programming," *Math. Program. B*, vol. 100, no. 1, pp. 95–132, 2004.
[44] A. Agarwal and B. Triggs, "Recovering 3D human pose from monocular images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 1, pp. 44–58, Jan. 2006.
[45] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A MATLAB-like environment for machine learning," in *Proc. BigLearn NIPS Workshop*, 2011, Art. no. EPFL-CONF-192376.
[46] C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun, "Hand pose estimation and hand shape classification using multi-layered randomized decision forests," in *Proc. Eur. Conf. Comput. Vis.*, Florence, Italy, 2012, pp. 852–863.
[47] J. Lee, N. Magnenat-Thalmann, and D. Thalmann, "Shared object manipulation," in *Context Aware Human–Robot and Human-Agent Interaction*. Cham, Switzerland: Springer, 2016, pp. 191–207. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-19947-4_9
[48] H. Liang, J. Yuan, D. Thalmann, and N. M. Thalmann, "Ar in hand: Egocentric palm pose tracking and gesture recognition for augmented reality applications," in *Proc. 23rd Annu. ACM Conf. Multimedia Conf.*, Brisbane, QLD, Australia, 2015, pp. 743–744.

**Hui Liang** (M'13) received the B.Eng. degree in electronics and information engineering and the M.Eng. degree in communication and information system from the Huazhong University of Science and Technology, Wuhan, China, in 2008 and 2011, respectively, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2016.

He was a Research Associate with the Institute for Media Innovation and a Research Fellow with the Rapid-Rich Object Search Lab, Nanyang Technological University. He is currently a Research Scientist with the Institute of High Performance Computing, A*STAR, Singapore. His current research interests include computer vision, machine learning, and human–computer interaction.

**Junsong Yuan** (M'08–SM'14) received the graduation degree from the Special Class for the Gifted Young, Huazhong University of Science and Technology, Wuhan, China, in 2002, the M.Eng. degree from the National University of Singapore, Singapore, and the Ph.D. degree from Northwestern University, Evanston, IL, USA.

He is currently an Associate Professor with the School of Electrical and Electronics Engineering, Nanyang Technological University (NTU), Singapore. His current research interests include computer vision, video analytics, gesture and action analysis, and large-scale visual search and mining.

Dr. Yuan was a recipient of the Best Paper Award from International Conference on Advanced Robotics (ICAR17), the 2016 Best Paper Award from the IEEE TRANSACTIONS ON MULTIMEDIA, the Doctoral Spotlight Award from IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09), the Nanyang Assistant Professorship from NTU, and Outstanding EECS Ph.D. Thesis Award from Northwestern University. He is currently a Senior Area Editor of the *Journal of Visual Communications and Image Representations*, an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and served as a Guest Editor for the *International Journal of Computer Vision*. He is the Program Co-Chair of ICME18 and the Area Chair of ACM Multimedia Conference, the IEEE Conference on Computer Vision and Pattern Recognition, the IEEE International Conference on Image Processing, the International Conference on Pattern Recognition, and the Asian Conference on Computer Vision.

**Jun Lee** (M'12) received the B.S. and M.S. degrees in computer science and engineering and the Ph.D. degree in advanced technology fusion from Konkuk University, Seoul, South Korea, in 2004, 2006, and 2012, respectively.

He was a Post-Doctoral Researcher with the Center for Robotics Research, Korea Institute of Science and Technology, Seoul, South Korea. He is an Assistant Professor with the Division of Computer and Information Engineering, Hoseo University, Asan, South Korea, in 2017. He was a Research Fellow with the Institute for Media Innovation, Nanyang Technological University, Singapore, being involved in various research projects on immersive virtual environments and 3-D telepresence systems. His current research interests include consistency management, grasping virtual object and manipulation, shared object manipulation, and increasing sense of presence in the virtual environments.

**Liuhao Ge** (S'16) is currently pursuing the Ph.D. degree from the Institute for Media Innovation, Interdisciplinary Graduate School, Nanyang Technological University, Singapore.

His current research interests include computer vision, multimedia, and machine learning.

**Daniel Thalmann** received the Ph.D. degree in computer science from the University of Geneva, Geneva, Switzerland, in 1977 and the Honorary Doctorate degree from University Paul-Sabatier, Toulouse, France, in 2003.

He is currently a Honorary Professor with the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, where he is the Director of Research Development with MIRALab Sarl. He has been the Founder of the Virtual Reality Lab, EPFL. From 2009 to 2017, he was a Visiting Professor with the Institute for Media Innovation, Nanyang Technological University, Singapore. He has published over 600 papers in graphics, animation, and virtual reality. He is one of the most highly cited scientists in Computer Graphics. Pioneer in research on virtual humans, his current research interests include social robots, crowd simulation, and virtual reality.

Prof. Thalmann was a recipient of the Eurographics Distinguished Career Award in 2010, the 2012 Canadian Human Computer Communications Society Achievement Award, and the CGI 2015 Career Achievement. He is the Co-Editor-in-Chief of the *Journal of Computer Animation and Virtual Worlds* and a member of the Editorial Board of 12 other journals. He was the Program Chair and the Co-Chair of several conferences, including IEEE Virtual Reality Conference, the ACM Symposium on Virtual Reality Software and Technology, and the ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry. More details on http://en.wikipedia.org/wiki/Daniel_Thalmann.