Hand Segmentation for Hand-Object Interaction from Depth map

Byeongkeun Kang, Student Member, IEEE, Kar-Han Tan, Senior Member, IEEE, Hung-Shuo Tai, Daniel Tretter, and Truong Q. Nguyen, Fellow, IEEE

Abstract—Hand-object interaction is important for many applications such as augmented reality, medical application, and human-robot interaction. Hand segmentation is a necessary pre-process to estimate hand pose and to recognize hand gesture or object in interaction. However, current hand segmentation method for hand-object interaction is based on color information which is not robust to objects with skin color, skin pigment difference, and light condition variations. Therefore, we propose the first hand segmentation method for hand-object interaction using depth map. This is challenging because of the small depth difference between hand and object during interaction. The proposed method includes two-stage randomized decision forest (RDF) with validation process, bilateral filtering, decision adjustment, and post-processing. We demonstrate the effectiveness of the proposed method by testing for five objects. The proposed method achieves the average \( F_1 \) score of 0.8826 using different model for each object and 0.8645 using a global model for entire objects. Also, the method takes only about 10 ms to process each frame. We believe that this is the state-of-the-art hand segmentation algorithm using depth map for hand-object interaction.

Index Terms—Hand segmentation, human-computer interaction, randomized decision forest

I. INTRODUCTION

Currently, hand is the most frequently used body part when one communicates with other people or electronics systems and devices. We use hand in navigating devices to touch screens, type keyboards, control mouses, and so on. Beyond this type of human-machine communications and interactions, there have been a lot of demands on more natural and convenient interaction technologies avoiding usages of keyboards and mouses for decades. Recently, with the expansion of virtual reality (VR), augmented reality (AR), robotics, and user interfaces in automobile, the development of new interaction technologies has become unavoidable since these applications require more natural interaction methods rather than input devices. For these applications, many researches have been conducted such as gesture recognition and hand pose estimation.

However, most technologies focus on understanding interactions which do not involve touching or handling any real world object although understanding interactions with objects is important in many applications. For example, to have better experience in augmented reality, we should be able to interact with real world object using our hands. Even in virtual reality, we might want to hold real world tools or sensors for haptic sensing while we are playing games or interacting in virtual world. Even though understanding interactions with real world objects is important, only limited researches have been conducted. We believe that this is because hand segmentation is much more difficult in hand-object interaction. After hand segmentation, similar approaches can be applied with the consideration of occlusions for the case involving real objects. For example, in hand pose estimation, we are mostly interested in the pixels on hand region and ignore the other pixels. Consequently, after segmentation, similar approaches can be applied using only the pixels on hand region while considering occlusions. Thus, hand segmentation is an important pre-processing step for hand-object interaction.

A. Related work

Hand segmentation has been studied for many extensions such as hand pose estimation, tracking, and gesture / sign recognition. In color image based methods, skin color based method has been popular for the segmentation of hand, face, and other body part. Jones et al. labeled 1 billion pixels and trained histogram and Gaussian mixture model [1]. They reported that histogram model is superior in accuracy and computational cost. Recently, Khan et al. explored six color spaces (IHLS, HSI, RGB, normalized RGB, YCbCr, and CIELAB) and nine skin modeling approaches (AdaBoost, Bayesian network, J48, multilayer perceptron, naïve Bayesian, random forest, RBF network, support vector machine, and histogram model) [2]. They showed that cylindrical color spaces (IHLS and HSI) and tree-based classifiers (random forest and J48) outperform others. Specifically for hand segmentation, Li et al. proposed pixel-level hand detection system for egocentric RGB camera using a sparse 50 dimensional combination of color, texture, and gradient histogram features [3], [4]. However, it was still challenging to deal with extreme conditions such as complete saturation, very dark scenes, and high contrast shadows. Also, it was hard to separate hand and arm. For more analysis and comparison of skin color based segmentation methods, we refer readers to the papers [5], [6]. Alternatively, Wang et al. used a color glove for hand tracking and also for segmentation by finding fully saturated colors [7]. This method is simple, but inconvenient and unnatural because of the requirement of wearing a glove.
For hand-object interaction, skin color based method has also been popular. Oikonomidis et al. proposed a hand tracking system to interact with an object [8]. Romero et al. estimated hand poses with and without grasped object by searching a large database (100,000 entries) based on a nearest neighbor [9], [10]. Both methods segmented hand by thresholding skin color in HSV space [11]. Wang et al. proposed a method to capture physically realistic hand manipulation data from multiple video streams to adapt the captured motion data to the interaction with new objects [12]. Hand segmentation is processed using a learned probabilistic model where the model is constructed from the color histogram of the first frame. The histogram is modeled using super-Gaussian mixture model based on [13]. Tzionas et al. proposed 3D object reconstruction system using hand motion from hand-object interaction [14]. They applied skin color based segmentation using Gaussian mixture model from [1]. However, skin color based segmentation has limitations in interacting with objects in skin color, segmenting from other body parts such as arm or face, skin pigment difference, and light condition variations.

For depth map based method, popular methods involve using a black wrist band or using randomized decision forest (RDF). The method using a black wrist band assumes that a hand is the closest part [15]–[17]. This segmentation method is very simple and effective. However, it is inconvenient and unnatural since it requires a user to wear a black wrist band and also the hand has to be the closest part. Most importantly, since this method processes segmentation by finding the connected component from the closest point, the object in interaction will be included as hand region. In RDF-based method, Tompson et al. collected dataset by painting a hand with bright red to label the hand on color image. Then they trained a RDF classifier using the collected dataset to perform per-pixel hand-background segmentation [18]. Sharp et al. first estimated a rough hand position using motion extrapolation, a hand detector, or the hand position from Kinect skeletal tracker [19]. Then within 3D search radius around the rough estimate, RDF was applied to classify pixels to hand or background. The RDF was trained on 100,000 synthetic images of hand and arm. For both methods, RDF was trained to segment an isolated hand which does not touch or handle any object.

To our knowledge, we propose the first hand segmentation method for hand-object interaction using depth map. This is challenging because of the small depth difference between hand and object while interacting with the object. However, it does not have any limitation in the color of object, skin color difference, light condition changes, and segmentation from other body parts.

The rest of the paper is organized as follows. In Section II, dataset collection and process is presented. Next, two-stage RDF algorithm and post-processes are proposed in Section III and Section IV, respectively. Then experimental results are presented in Section V. Finally, we conclude with a summary in Section VI.

Fig. 1. Example of collected data. (a) depth map; (b) ground truth segmentation; (c) color image.

Fig. 2. Objects in dataset. The dimensions for each object in cm are (a) top radius: 4.3, bottom radius: 3, height: 10.5; (b) top radius: 4.3, bottom radius: 2.9, height: 9.5; (c) radius: 3.1; (d) width: 10, height: 16.4, depth: 5.3; (e) width: 7.8, height: 14.7, depth: 2.5.

II. DATASET

A. Dataset

Dataset is collected using both a depth sensor and a color camera in the Intel RealSense camera system in the Sprout by HP. The resolution of depth map is 640×480, and that of color image is 1280×720. The dataset includes both touching/handling an object and articulating hand alone so that the trained model can be used for both cases. During data collection, the subject wears a blue color glove to estimate ground truth segmentation of hand using the corresponding color image of each depth map. Except for this purpose, color images are not used in any other processing step. Figure 1 shows an example of collected data. Figure 2 shows the five objects which are considered including two cups, one ball, and two boxes. For each object, 5,000 images are collected.
B. Ground truth hand segment

Ground truth hand segment is required for supervised learning. It is acquired by capturing the corresponding color image of depth map and by wearing a blue color glove. Each pixel on color image is classified to hand region using both HSV representation and YCrCb representation of color image. Among the classified pixels, the largest blob is selected as hand blob. Then the hand blob is projected on depth map. The largest blob on depth map is selected as ground truth hand blob.

III. RANDOMIZED DECISION FORESTS (RDF)

The framework of both general RDF and two-stage RDF is explained in this section. The basic structure of RDF is based on [20] which used RDF for human pose estimation from single depth map. We first explain our detailed strategy for training and testing of general RDF. Then we propose two-stage RDF to achieve better classification and to improve computational efficiency. The training, validation, and testing process for two-stage RDF are explained.

A. RDF

1) Training: To train RDF, we need to determine the size of the model and training data. We decide to train 10 trees for each experimental case and to limit the maximum depth of tree to 20. For training data, we randomly sample 1,000 pixels from hand region and another 1,000 pixels from non-hand region on each depth map. Training data is sampled only from the pixels with non-zero depth. In testing, we assume all the pixels with depth zero as background.

At each node in a tree, we train offset parameter vector $\theta$ in feature computation and threshold $t$ to compare with computed feature value. The offset parameter vector $\theta \in \mathbb{R}^4$ consists of two separate offsets $u \in \mathbb{R}^2$ and $v \in \mathbb{R}^2$. The two offsets are used to compute scalar feature value for each sampled data $x$ using the following equation:

$$f_\theta(I, x) = d_I(x + \frac{u}{d_I(x)} - d_I(x + \frac{v}{d_I(x)}))$$

where $d_I(x)$ is the depth at pixel $x$ on image $I$. If the depth at pixel is less than 10mm, the depth value is replaced to the maximum depth on the corresponding depth map. By comparing the computed feature $f_\theta(I, x)$ to threshold $t$, the sampled data $x$ is classified to left child or right child.

To train offset parameter vector $\theta$ and threshold $t$, we generate 100 vectors for offset candidates and 50 scalar values for threshold candidates. The offset candidates are randomly generated from linear distribution. For half of offset candidates, one offset (either $u$ or $v$) is fixed as $(0, 0)$. The threshold candidates are linearly distributed with a fixed step size. The range of offset candidates and that of threshold candidates are $[-0.4m, 0.4m]$ and $[-0.2m, 0.2m]$, respectively.

For each offset candidate, a computed feature vector is generated using the feature computation equation, where each component in the feature vector corresponds to each sampled pixel. Since 100 offset candidates are considered, 100 computed feature vectors are generated. Then using each computed feature vector, distribution $p_d$ is computed for each threshold candidate where distribution $p_d$ consists of four probabilities $p_d(c, h)$. $c$ and $h$ are child parameter and class parameter, respectively. 50 distributions are generated for each computed feature vector. Thus, in total, 5,000 possible distributions are generated.

Using the computed distributions at each node, offset parameter vector and threshold are determined as the parameters with the maximum information gain. The information gain $g$ is computed as follow:

$$g = \sum_{c \in \{l, r\}} \sum_{h \in \{0, 1\}} \frac{n(c)}{n(d) + n(r)} p_d(c, h) \log \frac{p_d(c, h)}{p_d(l, h)}$$

where $c$ is the parameter for left child $l$ or right child $r$, $h$ is the class parameter for hand 1 or non-hand 0, and $n(\cdot)$ is the number of samples.

The training is repeated at each node until it meets termination condition. The termination condition is based on (1) the maximum depth of tree, (2) distribution, and (3) the portion of remaining pixels. For the distribution, if the probability of a class at a child is larger than the pre-defined probability, the corresponding child becomes a leaf node. The pre-defined probability is set to 0.95 for RDF. For the portion of remaining pixels, if the number of remaining pixels for a child is less than 0.01%, the corresponding child also becomes a leaf node. For leaf node, the distribution is stored for classification in testing.

2) Testing: Probability map is computed from depth map using trained RDF. Each pixel on depth map is classified to left child or right child until it reaches a leaf node. When it reaches a leaf node, the corresponding probability is read from stored distribution. This process is repeated for each tree in the forest. After processing with entire trees, each pixel on computed probability map represents average probability of hand class. The pixels with depth zero on depth map are set to probability zero on probability map without processing RDF. In feature computation, if offset pixel is out of image or has depth less than 10mm, the depth value is replaced as the maximum depth on the corresponding depth map.

If classification includes bilateral filtering, final classification is processed after applying bilateral filtering on probability map as shown in IV-A. Otherwise, each pixel on probability map is directly classified to hand class or non-hand class based on the probability at each pixel.

B. Two-stage RDF

Two-stage RDF is proposed to achieve better segmentation and to improve computational efficiency. The first RDF detects rough hand region, and the second RDF segments hand details. The second RDF is trained and tested using the hand region from the first RDF so that the second RDF focuses on the segmentation of more meaningful region such as the separation of hand and object. Validation process is also proposed to improve segmentation and processing time.

1) Training: The first RDF is trained using the same training method of general RDF. To train the second RDF, we first process segmentation using the first RDF and find the bounding box of hand region (see Figure 3). Then 2,000 pixels
are randomly sampled from the pixels inside the bounding box (1,000 pixels from hand class and 1,000 pixels from non-hand class). Using the sampled pixels, the second RDF is trained with the same strategy in III-A1 except the termination condition based on distribution. For the second RDF, the pre-defined probability is set to 0.85 for all objects except object 3. For object 3, the probability is chosen as 0.95 (for details, see Section V). Although training samples are only from the bounding box, entire depth map is used for feature computation.

2) Validation: Since each tree is trained separately, trained forest is validated by computing $F_1$ score using tree sets. We first compute the score using entire forest, then compute the score using the trees excluding one tree for all possible cases. If any case of leaving one tree improves $F_1$ score more than 0.001, the tree is excluded from the forest. This process is repeated until excluding any tree does not improve the score more than 0.001. We found that in most cases, excluding a few trees actually helps achieving better $F_1$ score. Also, it is obvious that validation process reduces processing time. This validation process is only used for the second RDF in two-stage RDF. For the first RDF, entire trees are used without validation. Table I shows the validation process for the second RDF with threshold adjustment, bilateral filtering, and post-processing using validation dataset. The selected case is highlighted in bold type. For three objects, excluding one tree is selected, and for two objects, excluding two trees is chosen. The average improvement of $F_1$ score using validation dataset is 0.00322. The average decrement in the number of nodes is 300 (see Table III).

3) Testing: The first RDF is applied on depth map to find bounding box as shown in Figure 3. Then the second RDF is processed on the bounding box to compute probability map. So, the probabilities on probability map is only from the stored distribution of the second RDF.

### IV. Filtering and post-processing

This section explains processes to improve segmentation after computing probability map from depth map. The processes include modified bilateral filtering, classification decision adjustment, and post-processing. The parameters for these processes are determined using validation dataset and using entire trees before validation process.

#### A. Modified bilateral filter

Modified bilateral filter is applied on probability map to smooth probability $p$ using pixels with close distance and similar depth value. Since RDF computes the probability for each pixel independently, the filter helps to achieve more stable result.

Unlike generic bilateral filtering whose weights are based on input image (in this case, probability map), our filter weights are based on a separate image, depth map [21]. The filtering is defined as follows:

$$
\tilde{p}(x) = \frac{1}{w} \sum_{x_i \in \Omega} g_r(||d_I(x_i) - d_I(x)||)g_s(||x_i - x||)p(x_i).
$$

where $p(x)$ is probability at pixel $x$, $\tilde{p}(x)$ is filtered probability at pixel $x$, $\Omega$ is pixels within filter radius and depth difference, and $w$ is normalization term.

$$
w = \sum_{x_i \in \Omega} g_r(||d_I(x_i) - d_I(x)||)g_s(||x_i - x||).
$$

g_r(\cdot) and g_s(\cdot) are Gaussian functions for depth difference $r$ and distance $s$ from pixel, respectively.

$$
g_r(r) = \exp\left(-\frac{r^2}{2\sigma_r^2}\right), \quad g_s(s) = \exp\left(-\frac{s^2}{2\sigma_s^2}\right).
$$

After some experiments using validation dataset, we decide to use the same filter parameters for all the cases because of simplicity and computational complexity. For some cases using general RDF, larger filter radius improves segmentation performance, but also increases computational costs. The determined parameters are as follows. The radius is 5 pixels, and the maximum depth difference to be considered is 400 mm. Both standard deviations ($\sigma_r$ and $\sigma_s$) are 100.

#### B. Classification decision adjustment

After filtering probability map from RDF, a parameter for classification decision has to be determined. Although the most general parameter is 0.5 for probability map, we found that it is not the best parameter for our segmentation. To determine the parameter, the possible parameters are tested with the step size of 0.01 using validation dataset. Fig. 4(a) shows $F_1$ score for each object using RDF depending on threshold. It shows that the trend of $F_1$ score is similar between different cases. So, we decide to use the same parameter for classification of entire objects for generality. Fig. 4(b) and (c) show the average $F_1$ score, precision, and recall of validation dataset depending on threshold using RDF and two-stage RDF, respectively. For the experiment of two-stage RDF, the threshold for the first RDF is fixed to 0.5 whose recall is 0.9916. From this experiment, we set the parameter for RDF to 0.87 and that for the second RDF to 0.74.

![Fig. 3. Bounding box from the first RDF in two-stage RDF.](Image)

### Table I

**Validation process for the second RDF in two-stage RDF. The validated case is highlighted in bold type.**

| # of excl. trees | Object 1 | Object 2 | Object 3 | Object 4 | Object 5 |
|------------------|---------|---------|---------|---------|---------|
| 0                | 0.8775  | 0.8716  | 0.9188  | 0.8692  | 0.8515  |
| 1                | 0.8885  | 0.8750  | 0.9206  | 0.8720  | 0.8550  |
| 2                | 0.8886  | 0.8752  | 0.9223  | 0.8725  | 0.8568  |
| 3                | -       | -       | 0.9230  | -       | 0.8575  |
Fig. 4. Scores of validation dataset depending on threshold. The scores are computed using entire trees and without any filtering and post-processing. (a) $F_1$ score for each object using RDF; (b) average score using RDF; (c) average score using two-stage RDF. The maximum $F_1$ score of RDF is at the threshold 0.87 and that of the second RDF is at 0.74.

### TABLE II

| Method            | Score | |   | Method | Score | |   | Method | Score | |   |
|-------------------|-------|---|---|-------|-------|---|---|-------|-------|---|---|
|                  |       | |   |       |       | |   |       |       | |   |
| RDF               |       | |   |       |       | |   |       |       | |   |
| 0.50              | -     | Yes | - | -     | -     | - | - | 0.5952 | 0.9916 | - | 0.7434 |
| 0.87              | -     | Yes | - | -     | -     | - | - | 0.7069 | 0.9417 | - | 0.8333 |
|                   | -     | Yes | - | -     | 0.7656 | 0.9338 | - | 0.8409 |
|                   | Yes   | Yes | - | -     | 0.7445 | 0.9300 | - | 0.8264 |
|                   | Yes   | Yes | - | -     | 0.7707 | 0.9253 | - | 0.8404 |
| Two-stage RDF     |       | |   |       |       | |   |       |       | |   |
| 0.50, 0.50        | -     | Yes | - | -     | 0.7069 | 0.9417 | - | 0.8333 |
|                   | -     | Yes | - | -     | 0.7312 | 0.9811 | - | 0.8536 |
|                   | -     | Yes | - | -     | 0.7596 | 0.9758 | - | 0.8536 |
| 0.50, 0.74        | -     | Yes | - | -     | 0.8245 | 0.9163 | - | 0.8776 |
|                   | -     | Yes | - | -     | 0.8290 | 0.9237 | - | 0.8755 |
|                   | Yes   | Yes | - | -     | 0.8417 | 0.9216 | - | 0.8796 |
|                   | Yes   | Yes | - | -     | 0.8295 | 0.9294 | - | 0.8762 |
|                   | Yes   | Yes | - | -     | 0.8424 | 0.9277 | - | 0.8826 |

### C. Post-processing

After classification, the largest blob is detected from classification result and is considered as hand. We also restrict the maximum size of blob to 20cm on both vertical and horizontal axis. This process improves the segmentation result of both RDF and two-stage RDF. However, for two-stage RDF, since the improvement of $F_1$ score is less than 0.01, the limit of maximum size can be released for generality.

### V. Experimental Results

We analyze the results of the dataset in Section II using proposed models and methods in Section III and Section IV. Previous works have shown that RDF-based segmentation and classification work well for different users although those works do not include object interactions [18]–[20]. So, we focus on demonstrating RDF-based model can be used for hand-object interaction by exploring multiple objects. We first analyze the result using different model for each object in Section V-A and then show the result using a global model for entire objects in Section V-B. In the former experiment, we demonstrate the effectiveness of the proposed method by comparing the result from generic RDF. In the latter experiment, we extend the effectiveness of the proposed method by generalizing the model for entire objects.

In all experiments, dataset is separated to 50%, 25%, and 25% for training, validation, and testing, respectively. The parameters in models and methods are determined using training dataset and validation dataset. For quantitative analysis of segmentation performance, we measure $F_1$ score, precision, and recall. Processing time is measured using two different systems. System 1 has Intel i7-4790K CPU with 4.00GHz, 15.9GB RAM, and NVIDIA GeForce GTX TITAN. System 2 has Intel i7-3770 CPU with 3.40GHz, 16.0GB RAM, and NVIDIA GeForce GTX 770. Mainly, RDF and bilateral filtering are processed on GPU, and post-processing is processed on CPU. The measured processing time also includes reading images from hard drive.

### A. Different model for each object

This subsection shows the result using different model for each object. Each model is trained using the training dataset of the corresponding object. Overall quantitative result of test dataset is shown in Table II. RDF achieves $F_1$ score from 0.7434 using only RDF to 0.8409 using decision adjustment and post-processing. Comparing the two results, precision is
improved from 0.5952 to 0.7656, and recall is decreased from 0.9916 to 0.9338. Decision adjustment and post-processing improve $F_1$ score 0.0799 and 0.0688, respectively. Bilateral filtering improves the method without post-processing, but is not helpful with post-processing. We found that for RDF model, bilateral filter with larger kernel size improves $F_1$ score further. However, because of computational complexity and generality, the same small kernel is used with two-stage RDF.

Two-stage RDF achieves $F_1$ score from 0.8371 using only two-stage RDF to 0.8826 using threshold adjustment, bilateral filtering, post-processing, and validation process. Precision is improved from 0.7312 to 0.8424, and recall is decreased from 0.9811 to 0.9277. Decision adjustment and post-processing improve $F_1$ score 0.0335 and 0.0165, respectively. Bilateral filtering and validation process also improve from the method with other processes.

We also analyze models and results for each object. Table III shows the number of nodes in each forest. Although the number of nodes changes at each training, we want to share the size of our trained model for reference. In average, the number of nodes in RDF and the first RDF is 215.6, and that in the second RDF is 1600 after validation process. The number of nodes is the same in RDF and the first RDF because the RDF model is used for the first RDF. Although the number of nodes in the second RDF is much larger than that in the first RDF, the required computation is not directly proportional to the increment of the number of nodes. This is because the second RDF is only processed in the region from the first RDF model is used for the first RDF. Although the number of nodes in the second RDF is much larger than that in the first RDF, the required computation is not directly proportional to the increment of the number of nodes. This is because the second RDF is only processed in the region from the first RDF (see Figure 3 and Table V). After validation process, as the decrement of the number of trees in forest, the number of nodes is also decreased. So, the validation process improves both segmentation performance and processing time.

Table IV shows $F_1$ score, precision, and recall for each object using two-stage RDF with decision adjustment, bilateral filtering, post-processing, and validation process. The model for object 3 achieves the best $F_1$ score, and that for object 5 achieves the lowest $F_1$ score. The standard deviation of $F_1$ score for each object is 0.0229.

For object 3, we first trained the second RDF using the same termination condition with others. But the trained model has only 189 nodes which are too small, and does not achieve good result on validation dataset. We believe that since the object is too small, training is terminated before the model learns the separation of hand and object. So, we decide to train the second RDF using higher required distribution for termination. The trained model has much larger number of nodes and achieves 0.0434 improvement on $F_1$ score. Although the number of nodes is increased significantly, processing time is increased only about 0.1$ms$. The comparison is shown on Table V and Figure 5.

Table VI shows average processing time in $ms$ using different model for each object. It shows that using two-stage RDF only increases processing time about 1~1.5$ms$, but improves segmentation result significantly. This is because the second RDF is only applied on the bounding box from the first RDF. Bilateral filtering and post-processing takes about 2.5~4$ms$ and 0.6$ms$, respectively. As expected, validation process reduces processing time while improving segmentation result. Table VII shows the measured processing time for each object using two-stage RDF with validation process, filtering, and post-processing (different model for each object).

We also analyze models and results for each object. Table III shows the number of nodes in each forest. Although the number of nodes changes at each training, we want to share the size of our trained model for reference. In average, the number of nodes in RDF and the first RDF is 215.6, and that in the second RDF is 1600 after validation process. The number of nodes is the same in RDF and the first RDF because the RDF model is used for the first RDF. Although the number of nodes in the second RDF is much larger than that in the first RDF, the required computation is not directly proportional to the increment of the number of nodes. This is because the second RDF is only processed in the region from the first RDF (see Figure 3 and Table V). After validation process, as the decrement of the number of trees in forest, the number of nodes is also decreased. So, the validation process improves both segmentation performance and processing time.

Table IV shows $F_1$ score, precision, and recall for each object using two-stage RDF with decision adjustment, bilateral filtering, post-processing, and validation process. The model for object 3 achieves the best $F_1$ score, and that for object 5 achieves the lowest $F_1$ score. The standard deviation of $F_1$ score for each object is 0.0229.

For object 3, we first trained the second RDF using the same termination condition with others. But the trained model has only 189 nodes which are too small, and does not achieve good result on validation dataset. We believe that since the object is too small, training is terminated before the model learns the separation of hand and object. So, we decide to train the second RDF using higher required distribution for termination. The trained model has much larger number of nodes and achieves 0.0434 improvement on $F_1$ score. Although the number of nodes is increased significantly, processing time is increased only about 0.1$ms$. The comparison is shown on Table V and Figure 5.

Table VI shows average processing time in $ms$ using different model for each object. It shows that using two-stage RDF only increases processing time about 1~1.5$ms$, but improves segmentation result significantly. This is because the second RDF is only applied on the bounding box from the first RDF. Bilateral filtering and post-processing takes about 2.5~4$ms$ and 0.6$ms$, respectively. As expected, validation process reduces processing time while improving segmentation result. Table VII shows the measured processing time for each object using two-stage RDF with validation process, filtering, and post-processing (different model for each object).

B. Global model for all objects

This subsection shows the result using a global model for all objects. It is important to use a global model for all objects for generality. We explore two different approaches for a global...
model. The first method is training a global model using training dataset of entire objects, which we call re-training. The second method is using the combination of trained model from each object, which we call combination. For the former method, 10 trees are trained where the number of nodes in RDF is 74. For the latter method, 50 trees are used from the models in previous subsection where the number of nodes in RDF is 1078. The segmentation result using RDF from each approach is compared in Table VIII. It shows that the latter method achieves better result compare to the former method. Also, the latter method is convenient because it reuses the models from each object. Therefore, we decide to use the combination of trained models for the second RDF without comparing two approaches. However, we decide to use re-trained model for the first RDF since the performance is similar in the case with threshold 0.50 and without any post-processing. Also, re-trained model is used to reduce processing time since the number of nodes in the first RDF has more impact on processing time than that in the second RDF. To demonstrate, the processing time of each RDF is measured. The re-trained model takes about 4 ms to process each frame while the combination model takes about 12 ms in system 1.

Table IX shows quantitative segmentation result using a global two-stage RDF. The first RDF is trained using training dataset of entire objects. The second RDF is the combination of trained model for each object. The global model with all processes achieves 0.8645 in $F_1$ score, 0.8236 in precision, and 0.9111 in recall. It is 0.0181, 0.0188, and 0.0166 lower in $F_1$ score, precision, and recall compare to the result using
**TABLE IX**
Comparison of segmentation result and processing time using global two-stage RDF in test dataset. The first RDF is trained model using training dataset of entire objects. The second RDF is the combination of trained model for each object.

| Method                | Score | Processing time (ms) |
|-----------------------|-------|----------------------|
|                        | Precision | Recall | $F_1$ score | System 1 | System 2 |
|                        | score 0.50, 0.50 | Bilateral filter | Post-process | Validation | 0.6995 | 0.9811 | 0.8160 | 7.610 | 8.560 |
|                        | - | - | - | 0.7297 | 0.9713 | 0.8327 | 8.065 | 8.974 |
|                        | 0.50, 0.74 | - | - | - | 0.8128 | 0.9073 | 0.8567 | 7.610 | 8.560 |
|                        | - | - | - | 0.8254 | 0.9055 | 0.8630 | 8.065 | 8.974 |
|                        | Yes | - | - | 0.8149 | 0.9065 | 0.8575 | 10.082 | 12.140 |
|                        | Yes | Yes | - | 0.8258 | 0.9059 | 0.8633 | 10.575 | 12.819 |
|                        | Yes | - | Yes | 0.8124 | 0.9118 | 0.8585 | 9.878 | 12.192 |
|                        | Yes | Yes | Yes | 0.8236 | 0.9111 | 0.8645 | 10.577 | 12.766 |

**TABLE X**
Comparison of segmentation result for each object using global two-stage RDF with validation process, filtering, and post-processing.

| Object | 1 | 2 | 3 | 4 | 5 | Avg. |
|--------|---|---|---|---|---|-----|
| Precision | 0.8213 | 0.8275 | 0.8277 | 0.8259 | 0.7706 | 0.8236 |
| Recall | 0.9342 | 0.9055 | 0.9015 | 0.8798 | 0.9342 | 0.9111 |
| $F_1$ score | 0.8741 | 0.8649 | 0.8869 | 0.8520 | 0.8445 | 0.8645 |

Figure 6 compares segmentation results of the best method using RDF from different model for each object, that using two-stage RDF from different model for each object, and that using two-stage RDF from global model. Each column from left to right shows depth map, ground truth, and results as the same order as mentioned. It first shows that two-stage RDF achieves much better segmentation performance than RDF. Two-stage RDF segments hand quite clearly even though the depth difference between hand and object is very small. It also shows that global two-stage RDF achieves slightly lower segmentation performance than two-stage RDF from different model for each object. However, it still demonstrates the effectiveness of the global two-stage model by achieving good result.

**VI. CONCLUSION**

Hand segmentation is a necessary pre-processing step for hand pose estimation, object recognition, and gesture recognition while handling an object. We propose the first hand segmentation method for hand-object interaction using depth map to avoid the limitations of color information such as the color of objects, skin pigment difference, light condition variations, and segmentation from other body parts. The proposed method includes two-stage RDF with validation process, decision adjustment, bilateral filtering, and post-processing. The method is analyzed using five objects and using two different systems. The result demonstrates the effectiveness of the method by achieving $F_1$ score 0.8826 using different models for each object and 0.8645 using a global model for entire objects in about 10 ms per frame. We believe that this is the state-of-the-art hand segmentation algorithm for hand-object interaction.

**REFERENCES**

[1] M. J. Jones and J. M. Rehg, “Statistical color models with application to skin detection,” International Journal of Computer Vision, vol. 46, pp. 81–96, Jan. 2002.
[2] R. Khan, A. Hanbury, J. Stittinger, and A. Bais, “Color based skin classification,” Pattern Recognition Letters, vol. 33, pp. 157–163, Jan. 2012.
[3] C. Li and K. Kitani, “Pixel-level hand detection in ego-centric videos,” in Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, Jun. 2013, pp. 3570–3577.
[4] M. Cai, K. Kitani, and Y. Sato, “A scalable approach for understanding the visual structures of hand grasps,” in Robotics and Automation (ICRA), 2015 IEEE International Conference on, May 2015, pp. 1360–1366.
[5] S. L. Phung, A. Bouzerdoum, and D. Chai, “Skin segmentation using color pixel classification: analysis and comparison,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 27, no. 1, pp. 148–154, Jan. 2005.
[6] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, “A survey of skin-color modeling and detection methods,” Pattern Recognition, vol. 40, pp. 1106–1122, Mar. 2007.
[7] R. Y. Wang and J. Popović, “Real-time hand-tracking with a color glove,” ACM Trans. Graph., vol. 28, no. 3, 2009.
[8] I. Oikonomidis, N. Kyriazis, and A. Argyros, “Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints,” in Computer Vision (ICCV), 2011 IEEE International Conference on, Nov. 2011, pp. 2088–2095.
[9] J. Romero, H. Kjellstrom, and D. Kragic, “Hands in action: real-time 3d reconstruction of hands in interaction with objects,” in Robotics and Automation (ICRA), 2010 IEEE International Conference on, May 2010, pp. 458–463.
[10] J. Romero, H. KjellstroM, C. H. Ek, and D. Kragic, “Non-parametric hand pose estimation with object context,” Image Vision Comput., vol. 31, no. 8, pp. 555–564, Aug. 2013.
[11] A. A. Argyros and M. I. A. Lourakis, “Real-time tracking of multiple skin-colored objects with a possibly moving camera,” in Computer Vision - ECCV 2004, May 2004, pp. 368–379.
[12] Y. Wang, J. Min, J. Zhang, Y. Liu, F. Xu, Q. Dai, and J. Chai, “Video-based hand manipulation capture through composite motion control,” ACM Trans. Graph., vol. 32, no. 4, pp. 43:1–43:14, Jul. 2013.
Fig. 6. Comparison of segmentation result. (a) Depth map; (b) Ground truth; (c) Result using RDF from different model for each object with threshold (0.87), bilateral filtering, and post-processing; (d) Result using two-stage RDF from different model for each object with threshold (0.50, 0.74), bilateral filtering, post-processing, and validation process; (e) Result using global two-stage RDF with the same processes in (d). Two-stage RDF achieves much better segmentation result than RDF visually. The result using RDF shows that in some cases, object is not rejected clearly. However, two-stage RDF segments hand quite clearly even though the depth difference between hand and object is very small. Global two-stage RDF achieves slightly lower segmentation performance than two-stage RDF from different model for each object. The bounding box on the first column is to visualize the corresponding region of other columns. The region is selected around the center of hand with the size of $280 \times 280$ pixels. It is different to the bounding box from the first RDF in two-stage RDF (see Figure 3).

[13] J. A. Palmer, K. Kreutz-Delgado, and S. Makeig, “Super-gaussian mixture source model for ica,” in Independent Component Analysis and Blind Signal Separation, 2006, pp. 854–861.
[14] D. Tzionas and J. Gall, “3d object reconstruction from hand-object interactions,” in International Conference on Computer Vision (ICCV), Dec. 2015.
[15] B. Kang, S. Tripathi, and T. Nguyen, “Real-time sign language fingerspelling recognition using convolutional neural networks from depth map,” in Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on, Nov. 2015.
[16] B. Kang, Y. Lee, and T. Nguyen, “Efficient hand articulations tracking using adaptive hand model and depth map,” in Advances in Visual Computing, Dec. 2015, pp. 586–598.
[17] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, “Realtime and robust hand tracking from depth,” in Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, Jun. 2014, pp. 1106–1113.
[18] 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, pp. 169:1–169:10, Sep. 2014.
[19] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi, “Accurate, robust, and flexible real-time hand tracking,” in Proceedings of the 23rd Annual ACM Conference on Human Factors in Computing Systems, ser. CHI ’15. ACM, 2015, pp. 3633–3642.
[20] 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,” in Computer Vision and Pattern Recognition.
[21] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in Computer Vision, Sixth International Conference on, Jan. 1998, pp. 839–846.

Byeongkeun Kang (S’16) received the B.S. degree with the highest honor in electrical and electronic engineering from Yonsei University, Seoul, Republic of Korea in 2013, and the M.S. degree in electrical and computer engineering from the University of California, San Diego, La Jolla, CA in 2015, where he is currently pursuing the Ph.D. degree with the Video Processing Lab. His current research interests include segmentation, pose estimation, and recognition for human-computer interaction.

Kar-Han Tan (SM’11) is Vice President of Engineering at NovuMind Inc., where he is leading the invention of next generation Deep Learning methods for the Intelligent Internet of Things. Previously, he was Head of Advanced Development at HP Immersive Computing, where he led the Computer Vision and real time collaboration application teams for Sprout 1.0, which launched in late 2014 and is currently rolling out worldwide. At HP Labs, Kar-Han was Master Technologist, where he worked on 3D capture and display technologies as well as next generation remote collaboration systems. He received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign, where he was a Beckman Graduate Fellow, M.S. from UCLA, and B.Sc. from the National University of Singapore. Kar-Han contributes actively to the research community and has received best paper awards for his work. Prior to HP, he was Manager of Algorithms Group at EPSON R&D, where he led the invention of View Projection, a technique that enables one-touch setup of light displays on arbitrary surfaces. He co-invented Multi-Flash Imaging at Mitsubishi Electric Research Lab (MERL), and the Virtual Structures algorithm at UCLA, widely recognized today as one of the fundamental techniques for mobile robot formation control. Kar-Han lives in Palo Alto, CA with his wife and two children.

Hung-Shuo Tai is a Research Engineer at NovuMind Inc., where he is working on computer vision application with deep learning. He received the B.S. degree in electrical engineering from National Cheng Kung University in Tainan, Taiwan, and earned his MSEE from Saint Louis University in 2014. His current research interests include 2D/3D computer vision, deep learning, and embedded system.

Daniel Tretter is a Research Manager at Hewlett Packard, where he oversees a team of computer vision and imaging specialists and leads the development of new technologies for HPs Sprout and other computer products. Previously, he was a researcher and project manager at HP Labs, where he published a number of technical papers and two book chapters. Daniel has also co-authored 60 patents in the areas of image processing and computer vision. He received B.S. degrees in mathematics and electrical engineering from the Rose-Hulman Institute of Technology in Terre Haute, IN in 1987 and earned his MSEE and PhD degrees from Purdue University in 1994. Daniel’s current research interests include 3D scanning, object recognition, and image segmentation.

Truong Q. Nguyen (F’05) is currently a Professor at the ECE Dept., UCSD. His current research interests are 3D video processing and communications and their efficient implementation. He is the co-author (with Prof. Gilbert Strang) of a popular textbook, Wavelets & Filter Banks, Wellesley-Cambridge Press, 1997, and the author of several matlab-based toolboxes on image compression, electrocardiogram compression, and filter bank design. He has over 400 publications.

Prof. Nguyen received the IEEE Transaction in Signal Processing Paper Award (Image and Multidimensional Processing area) for the paper he co-wrote with Prof. P. P. Vaidyanathan on linear-phase perfect-reconstruction filter banks (1992). He received the NSF Career Award in 1995 and is currently the Series Editor (Digital Signal Processing) for Academic Press. He served as Associate Editor for the IEEE Transaction on Signal Processing 1994-96, for the Signal Processing Letters 2001-2003, for the IEEE Transaction on Circuits & Systems from 1996-97, 2001-2004, and for the IEEE Transaction on Image Processing from 2004-2005.