HAND SEGMENTATION FOR HAND-OBJECT INTERACTION FROM DEPTH MAP

Byeongkeun Kang[⋆] *Kar-Han Tan*[†] *Hung-Shuo Tai*[†] *Daniel Tretter*[‡] *Truong Nguyen*[⋆]

[⋆] Department of Electrical and Computer Engineering, UC San Diego, La Jolla, CA 92093 USA
[†] NovuMind Inc., Santa Clara, CA 95054 USA
[‡] Hewlett-Packard, Inc., Palo Alto, CA 94304 USA

ABSTRACT

Hand-object interaction is important for many applications such as augmented reality, medical application, and human-robot interaction. To understand hand-object interaction, hand segmentation is a necessary pre-process. However, current method 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 only depth map. The proposed method includes randomized decision forest (RDF), bilateral filtering, decision adjustment, and post-processing. We demonstrate the effectiveness of the method by testing for five objects. The method achieves the average F_1 score of 0.8409 and 0.8163 for the same object and new object, respectively. Also, the method takes less than $10ms$ to process each frame.

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

1. INTRODUCTION

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. We believe that this is because hand segmentation is much more difficult in hand-object interaction. Thus, we propose and analyze hand segmentation for hand-object interaction using depth map.

1.1. 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 [1–6]. Alternative method is using a color glove [7]. 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 is simple and effective, however, is inconvenient and unnatural because of wearing a black wrist band [8–10]. Importantly, since this method processes segmentation by finding connected components, the object in interaction cannot be separated from hand. In RDF-based method, Tompson *et al.* collected dataset and trained a RDF classifier to perform segmentation [11]. Sharp *et al.* first estimated a rough hand position using motion extrapolation, a hand detector, or the hand position from Kinect skeletal tracker [12]. Then RDF was applied to process segmentation. Both methods did not consider touching or handling any object.

For hand-object interaction, to our knowledge, all the proposed system uses color information to process hand segmentation. Oikonomidis *et al.* and Romero *et al.* segmented hand by thresholding skin color in HSV space [13–16]. Wang *et al.* processed hand segmentation using a learned probabilistic model where the model is constructed from the color histogram of the first frame [17]. Tzionas *et al.* applied skin color based segmentation using Gaussian mixture model [18]. 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.

Therefore, we propose the first hand segmentation method for hand-object interaction using only depth map to avoid the limitations of skin color based method.

2. METHOD

2.1. Dataset

Dataset is collected using both a depth sensor and a color camera in the Sprout by HP. 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

This work is supported in part by NSF grant IIS-1522125.

(a) (b)

Fig. 1. Example of collected data. (a) Depth map. (b) Ground truth segmentation.

(a) (b) (c) (d) (e)

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.

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. Each pixel on color image is classified to potential hand region using both HSV representation and YCrCb representation of the image. Among the classified pixels, the largest blob is selected as hand blob. 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.

2.2. Randomized decision forests

2.2.1. 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.

As proposed by [19], at each node in a tree, we train offset parameter vector $\boldsymbol{\theta}$ in feature computation and threshold t to compare with computed feature value. The offset parameter vector $\boldsymbol{\theta} \in \mathbb{R}^4$ consists of two separate offsets $\boldsymbol{u} \in \mathbb{R}^2$ and $\boldsymbol{v} \in \mathbb{R}^2$. The two offsets are used to compute scalar feature value for each sampled data $\boldsymbol{x}$ using the following equation:

$$f_{\boldsymbol{\theta}}(\boldsymbol{I}, \boldsymbol{x}) = d_{\boldsymbol{I}}\left(\boldsymbol{x} + \frac{\boldsymbol{u}}{d_{\boldsymbol{I}}(\boldsymbol{x})}\right) - d_{\boldsymbol{I}}\left(\boldsymbol{x} + \frac{\boldsymbol{v}}{d_{\boldsymbol{I}}(\boldsymbol{x})}\right) \quad (1)$$

where $d_{\boldsymbol{I}}(\boldsymbol{x})$ is the depth at pixel $\boldsymbol{x}$ on image $\boldsymbol{I}$. 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 $f_{\boldsymbol{\theta}}(\boldsymbol{I}, \boldsymbol{x})$ to threshold t, the sampled data $\boldsymbol{x}$ is classified to left child or right child.

To train offset parameter vector $\boldsymbol{\theta}$ 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 $\boldsymbol{u}$ or $\boldsymbol{v}$) 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. Then using each computed feature vector, distribution $\boldsymbol{p}_d$ is computed for each threshold candidate where distribution $\boldsymbol{p}_d$ consists of four probabilities $p_d(c, h)$. c and h are child parameter and class parameter, respectively.

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:

$$g = \sum_{c \in \{l,r\}} \sum_{h \in \{0,1\}} \frac{n(c)}{n(l) + n(r)} p_d(c, h) \log p_d(c, h) \quad (2)$$

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.

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

2.2.2. 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. Otherwise, each pixel on probability map is directly classified based on the probability at each pixel.

2.3. 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 [20]. The filtering is defined as follows:

$$\widetilde{p}(\boldsymbol{x}) = \frac{1}{w} \sum_{\boldsymbol{x}_i \in \Omega} g_r(|d_{\boldsymbol{I}}(\boldsymbol{x}_i) - d_{\boldsymbol{I}}(\boldsymbol{x})|) g_s(\|\boldsymbol{x}_i - \boldsymbol{x}\|) p(\boldsymbol{x}_i). \tag{3}$$

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

$$w = \sum_{\boldsymbol{x}_i \in \Omega} g_r(|d_{\boldsymbol{I}}(\boldsymbol{x}_i) - d_{\boldsymbol{I}}(\boldsymbol{x})|) g_s(\|\boldsymbol{x}_i - \boldsymbol{x}\|). \tag{4}$$

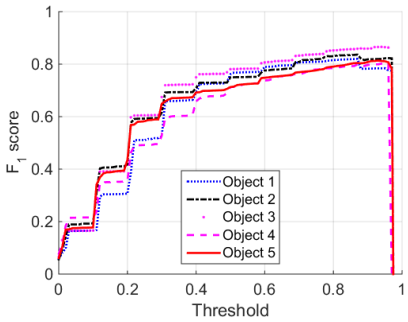
$g_r(\cdot)$ and $g_s(\cdot)$ are Gaussian functions for depth difference r and distance s from pixel, respectively.

$$g_r(r) = \exp\left(-\frac{r^2}{2\sigma_r^2}\right), \quad g_s(s) = \exp\left(-\frac{s^2}{2\sigma_s^2}\right). \tag{5}$$

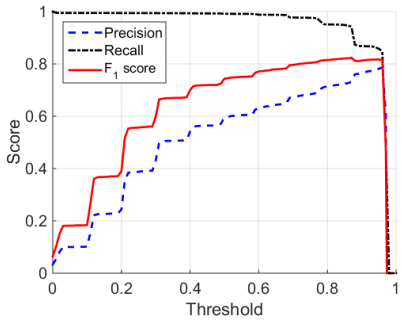
After some experiments using validation dataset, we decide to use the same filter parameters for all the cases because of simplicity and computational complexity. For some cases, larger filter radius improves segmentation performance, but also increases computational costs. The determined parameters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

2.4. Classification decision adjustment

For the 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. 3(a) shows F_1 score for each object depending on threshold. It shows that the trend of F_1 score is similar between different objects. So, we decide to use the same parameter for classification of entire objects for generality. Fig. 3(b) shows the average F_1 score, precision, and recall of validation dataset depending on threshold. From this experiment, we set the parameter to 0.87.



(a)



(b)

Fig. 3. Scores of validation dataset depending on threshold. The scores are computed without any filtering and post-processing. (a) F_1 score for each object. (b) Average score. The maximum F_1 score is at the threshold 0.87.

2.5. Post-processing

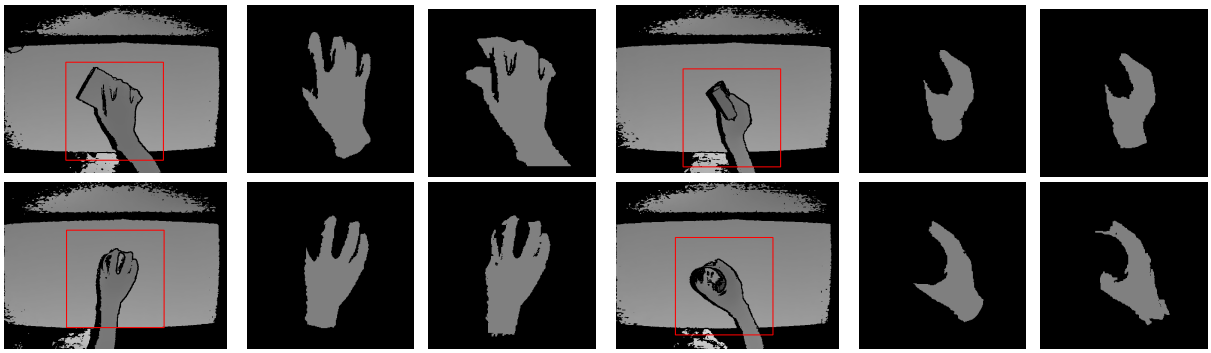
After classification, the largest blob is detected from classification result and is considered as hand. We also restrict the maximum size of blob on both vertical and horizontal axis.

3. EXPERIMENTAL RESULTS

For quantitative analysis of segmentation performance, we measure F_1 score, precision, and recall. Processing time is measured using two 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

Table 1. Comparison of segmentation result on the same object.

Method			Score			Processing time (ms)	
Threshold	Bilateral filter	Post-process	Precision	Recall	F_1 score		
0.50	-	-	0.5952	0.9916	0.7434	-	-
	-	Yes	0.6928	0.9829	0.8122	-	-
0.87	-	-	0.7318	0.9417	0.8233	4.29	5.01
	-	Yes	0.7656	0.9338	**0.8409**	4.83	5.77
	Yes	-	0.7445	0.9300	0.8264	7.11	8.98
	Yes	Yes	0.7707	0.9253	**0.8404**	7.74	9.80



(a) (b) (c) (a) (b) (c)

Fig. 4. Segmentation result. (a) Depth map. (b) Ground truth. (c) Result using RDF in Table 1 with threshold (0.87), bilateral filtering, and post-processing. The enlarged region is selected for better visualization around the center of hand.

Table 2. Comparison of segmentation result on new object.

Method			Score		
Thres.	Bilateral	Post-p.	Prec.	Recall	F_1
0.50	-	-	0.5578	0.9898	0.7132
	-	Yes	0.6816	0.9731	0.8012
0.87	-	-	0.7017	0.9030	0.7883
	-	Yes	0.7568	0.8859	0.8144
	Yes	-	0.7152	0.9012	0.7964
	Yes	Yes	0.7549	0.8915	**0.8163**

time also includes reading images from hard drive.

We experiment the proposed method in two approaches. One is training and testing separate model for each object. To train and test a model for each object, the dataset of each object is separated to 50%, 25%, and 25% for training, validation, and testing, respectively. The average result is shown in Table 1. The trained 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. Average processing time is presented in ms. Overall, the proposed method achieves more than 100 frames per second.

The other approach is training using four objects and testing on the other object. We experimented all five possible combination. The average result is shown in Table 2. The trained RDF achieves F_1 score from 0.7132 using only RDF to 0.8163 using entire proposed method. Comparing the two results, precision is improved from 0.5578 to 0.7549, and recall is decreased from 0.9898 to 0.8915. Decision adjustment and post-processing improve F_1 score 0.0751 and 0.0880, respectively.

4. CONCLUSION

Hand segmentation is a necessary pre-processing step to understand hand-object interaction. We propose the first hand segmentation method for hand-object interaction using only 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 RDF with decision adjustment, bilateral filtering, and post-processing. The method is analyzed using five objects and using two systems. The result demonstrates the effectiveness of the method by achieving F_1 score 0.8409 and 0.8163 for the same object and new object, respectively, in less than $10ms$ per frame.

5. 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. Stttinger, and A. Bais, "Color based skin classification," *Pattern Recognition Letters*, vol. 33, pp. 157–163, Jan. 2012.

[3] C. Li and K.M. Kitani, "Pixel-level hand detection in ego-centric videos," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, June 2013, pp. 3570–3577.

[4] M. Cai, K.M. 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] 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.

[9] 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.

[10] 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*, June 2014, pp. 1106–1113.

[11] 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, Sept. 2014.

[12] 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 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, CHI '15, pp. 3633–3642, ACM.

[13] I. Oikonomidis, N. Kyriazis, and A.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.

[14] 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.

[15] J. Romero, H. KjellströM, 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.

[16] 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.

[17] 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, July 2013.

[18] D. Tzionas and J. Gall, "3d object reconstruction from hand-object interactions," in *International Conference on Computer Vision (ICCV)*, Dec. 2015.

[19] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 1297–1304.

[20] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, Sixth International Conference on*, Jan. 1998, pp. 839–846.