Random Forest With Learned Representations for Semantic Segmentation

Byeongkeun Kang and Truong Q. Nguyen, *Fellow, IEEE*

***Abstract*— We present a random forest framework that learns the weights, shapes, and sparsities of feature representations for real-time semantic segmentation. Typical filters (kernels) have predetermined shapes and sparsities and learn only weights. A few feature extraction methods fix weights and learn only shapes and sparsities. These predetermined constraints restrict learning and extracting optimal features. To overcome this limitation, we propose an unconstrained representation that is able to extract optimal features by learning weights, shapes, and sparsities. We then present the random forest framework that learns the flexible filters using an iterative optimization algorithm and segments input images using the learned representations. We demonstrate the effectiveness of the proposed method using a hand segmentation dataset for hand-object interaction and using two semantic segmentation datasets. The results show that the proposed method achieves real-time semantic segmentation using limited computational and memory resources.**

***Index Terms*— Semantic segmentation, random forest, feature extraction, object segmentation, real-time systems.**

I. Introduction

ACCURATE and efficient semantic segmentation is a fundamental task in a variety of computer vision applications including autonomous driving, human-machine interaction, and robot navigation. Reducing computational complexity and memory usage is important to minimize response time and power consumption for portable devices such as robots and virtual/augmented devices. It is also beneficial for vehicles and robots to navigate in actively changing environments and for human-machine interaction devices to communicate without delay. However, it is challenging to achieve accurate and efficient semantic segmentation because every pixel needs to be classified using limited computational resources.

To achieve accurate and efficient pixel-wise classification, Shotton *et al.* presented a random forest-based method and applied it for semantic segmentation and body pose estimation in [1]–[3]. This work has been broadly employed in many related applications [4]–[7] and in Microsoft Kinect [8]. To improve accuracy in semantic segmentation, convolutional neural network-based methods have been proposed in [9]–[14]. Part of the reason that deep learning-based methods outperform other methods is its ability to learn fine representations along with the hierarchical structure and non-linear activations. The methods using random forest and using deep neural networks outperform most of the other methods in the task of semantic segmentation in terms of efficiency and accuracy. Between two approaches, random forest-based methods have an advantage in computational complexity and memory usage while deep learning-based methods achieve higher accuracy. Therefore, we propose a random forest framework that employs unconstrained representations, learns optimal features by using an optimization algorithm, and inferences in real-time using limited computational and memory resources.

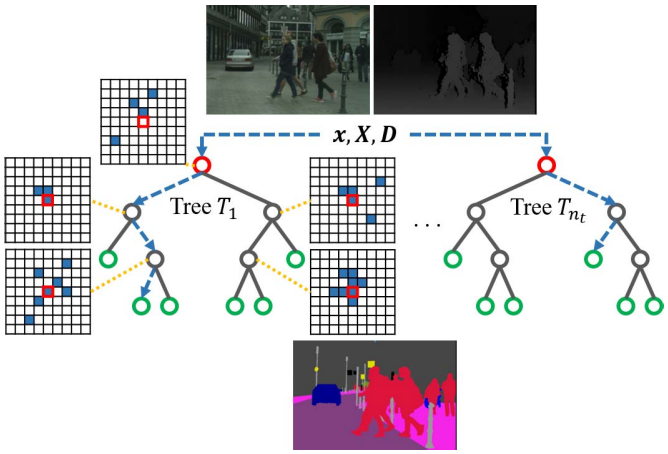


Fig. 1. Illustration of the random forest with learned representations for semantic segmentation. The top images show input images (a color image and a depth map). The bottom image shows an expected output (class label for each pixel). The rectangles with grid show examples of learned representations with various shapes, sparsities, and weights. Red bounding boxes present input data points, and blue color squares denote the offset points that are used to compute features.

The proposed method can be applied to any input (e.g. color image, depth map, point cloud) and for any pixel-wise classification task. We validate the proposed method in the task of semantic segmentation and hand segmentation for hand-object interaction. Semantic segmentation is needed in many applications such as autonomous driving and robot navigation [15]–[19]. We apply the proposed method to a road scene dataset [15] and an indoor scene dataset [20], and use color images along with disparity/depth maps as input. Hand segmentation is also a fundamental task in human-machine interaction that is demanded in virtual reality (VR), augmented reality (AR), robotics, and user interfaces

Manuscript received May 11, 2018; revised December 7, 2018 and January 20, 2019; accepted March 3, 2019. Date of publication March 14, 2019; date of current version June 4, 2019. This work was supported by NSF under Grant IIS-1522125. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ming-Ming Cheng. *(Corresponding author: Byeongkeun Kang.)*

The authors are with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: bkkang@ucsd.edu; tqn001@eng.ucsd.edu).

Digital Object Identifier 10.1109/TIP.2019.2905081

in an automobile [4]–[6], [21]–[35]. We apply the method to a depth map-based hand segmentation dataset for hand-object interaction [14]. By experiments, we show that the proposed method can be applied to real-time semantic segmentation task using limited computational and memory resources.

In this paper, we propose the novel unconstrained representation that is able to learn weights, shapes, and sparsities in Section III-B. Then, we propose a random forest framework that learns the weights, shapes, and sparsities of the representation and inferences input data using the learned representation in Section III-C. The section explains selecting training data using bootstrap aggregation and boosting, learning splitting functions using particle swarm optimization, and inferencing input images. In Section IV, we demonstrate the effectiveness of the proposed method on three tasks: road scene semantic segmentation, indoor scene semantic segmentation, and hand segmentation for hand-object interaction. We use publicly available Cityscapes dataset [15], NYUDv2 dataset [20], and HOI dataset [14].

In summary, the contributions of our work are as follows:

- We propose the unconstrained representation that is able to represent any weights, shapes, and sparsities.
- We develop the random forest framework that learns the weights, shapes, and sparsities of the representation by using particle swarm optimization.
- We verify the effectiveness and efficiency of the proposed method on semantic segmentation and hand segmentation tasks.

II. Related Works

A. Per-Pixel Classification Using Random Forest

Random forest is an ensemble learning method and consists of a set of decision trees [36]–[38] as shown in Figs. 1 and 2. It is robust to noisy and variant data because of the combination of multiple trees with varying features and splitting criteria. Also, it is computationally less complex than typical neural networks. Part of the reason is that an input data is processed only log-scale portion of each tree based on the conditions in the ancestral nodes (see blue dotted lines in Figs. 1 and 2).

Shotton *et al.* presented semantic texton forest for image categorization and semantic segmentation [1]. In order to avoid expensive computations of local descriptors (e.g. HOG [39], SIFT [40]) or filter-bank responses, they employed splitting functions using the value of a single pixel, the sum, the difference, and the absolute difference of a pair of pixels. The method was extremely fast comparing to k-means clustering or nearest-neighbor assignment using feature descriptor. Schroff *et al.* investigated using not only local features but also global and context-rich features in the random forest for semantic segmentation [41]. They showed that combining multiple features improves accuracy, and further demonstrated that relaxing constraints on features leads to higher classification accuracy. Shotton *et al.* extended the random forest [1], [42] to real-time body pose estimation by classifying each pixel to body parts [2], [3]. They employed the depth difference between a pair of pixels as a feature. The feature fulfilled depth invariance property by calculating the location of the pixels considering depth information. This work was extended to hand segmentation by Tompson *et al.* [6]. Sharp *et al.* [5] employed the memory-efficient random forest method [43] to predict initial hand pose in hand tracking. Kang *et al.* [7] proposed the two-stage random forest method consisting of detection and segmentation for hand segmentation in hand-object interaction.

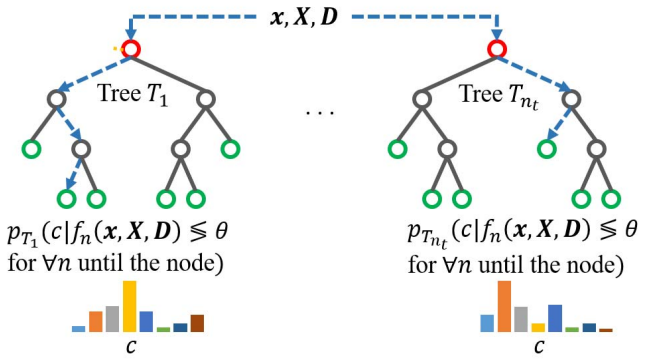


Fig. 2. Random forest. Red, black, and green circles denote root nodes, split nodes, and leaf nodes, respectively. At leaf nodes, the forest estimates and uses the conditional probability of being each class given the specific leaf node.

The earlier works used hand-crafted features such as local descriptors (e.g. HOG, SIFT) or filter-bank responses [44]. Then, relatively recent works used pixel value difference as a feature that learns offsets while using fixed weights ($+1$ and -1) [2], [3], [6], [7], [42]. As investigated on the benefits of relaxing constraints of features [41], we further relax constraints of feature representations and propose the unconstrained representation. We enable learning optimal representation by learning weights, shapes, and sparsities of the proposed representation. Then, the learned feature is employed in the random forest framework for efficiency to achieve real-time inferencing.

B. Per-Pixel Classification Using Deep Learning

Deep learning-based methods achieved state-of-the-art accuracy in per-pixel classification recently. Long *et al.* proposed fully convolutional neural networks (FCN) for semantic segmentation by converting fully connected layers to convolutional layers in the neural networks for image classification [9], [45]. Consequently, it takes an input of arbitrary size and produces an output of the corresponding size with pixel-wise prediction. Additional efforts have been made to achieve higher accuracy in [10]–[14] and [46]. Zheng *et al.* [11] proposed the convolutional neural networks that combine the strength of convolutional neural networks and conditional random field (CRF)-based probabilistic graphical modeling. They formulated CRF as recurrent neural networks and attached the recurrent neural networks following FCN. Chen *et al.* [12], [13] improved semantic segmentation using convolution with upsampled filters, atrous spatial pyramid pooling, and fully connected CRF. Yu and Koltun [10] proposed an additional context module to aggregate multiscale

information without losing resolution. Kang *et al.* [14] presented the depth-adaptive deep neural network that compensates depth variation to achieve depth-invariance property for semantic segmentation. Wang *et al.* [46] proposed the non-local operation that captures long-range dependencies to improve the accuracy of object detection/segmentation as well as video classification.

Several other neural networks were proposed considering computational complexity [47], [48]. Badrinarayanan *et al.* [47] proposed an efficient convolutional neural network in terms of memory and computational time during inference. The network consists of an encoder network and a decoder network. The encoder network records pooling indices in max-pooling steps and the corresponding decoder network uses the recorded indices to perform nonlinear upsampling. Paszke *et al.* [48] presented an efficient neural network to achieve real-time semantic segmentation. They analyzed various network architectures including early downsampling, nonlinear operations, and factoring filtering.

While neural network-based methods achieve state-of-the-arts accuracy by learning optimal weights and biases in multiple layers with nonlinear activation functions and by extracting meaningful information from input data, they demand high computational and memory resources. Hence, we aim to present a random forest-based framework that is able to process real-time semantic segmentation using only limited computational and memory resources (e.g. low-end GPU or embedded system). Moreover, while typical neural networks use a hand-picked shape and sparsity [10]–[13], we learn shapes and sparsities along with weights by using an optimization algorithm.

C. Learning Representation

In random forest, learning two offset points for a feature representation has been one of the most popular representation learning methods [2], [3], [6], [7], [42]. While it can learn any offset vectors to represent various shapes and sparsities, it has constraints of fixed weights ($+1$ or -1) and of using only two offset points.

Typical neural networks learn representations using dense filters with various filter-sizes [49]–[51]. Recently, dilated convolution (also known as atrous convolution) was introduced to learn sparse representations [13]. The sparse convolution was also applied in depth-adaptive convolution to learn depth-invariant representations [14]. While these representations have square shapes, active convolution and deformable convolution were presented to learn representations with various shapes in [52] and [53]. Both methods learn shapes of convolution filters using a training dataset. Active convolution defines learnable position parameters to represent various forms of receptive fields [52]. Deformable convolution uses the offset field similar to the position parameters [53]. It computes the offset field using the input feature map at each spatial location.

In this work, we aim to learn weights, shapes, and sparsities of descriptors by proposing an unconstrained representation and by applying particle swarm optimization. Comparing to learning two offset points in random forest, we also learn weights and the number of offset points. Concerning the recent convolution layers, we further learn the number of offset points. Moreover, the proposed representation has depth-invariance property as offset points are computed with the consideration of the distance from a camera.

III. Proposed Method

In this section, we introduce an unconstrained representation that is able to represent any weights, shapes, and sparsities in Section III-B. The unconstrained filter is proposed to learn optimal filters for a given task and dataset during training. We employ the proposed representation in the framework of random forest in Section III-C. We describe details about training procedure including learning optimal representations and selecting training data points in Section III-C1 and about inference method in Section III-C2.

A. Notation

Let $\boldsymbol{X} \in \mathbb{R}^{p \times q}$ and $\boldsymbol{Y} \in \mathbb{R}^{p \times q \times n_c}$ be the matrices denoting an input image and an output probability map of a random forest where p, q, and n_c represent the height, the width, and the number of classes, respectively. At a location $\boldsymbol{x} \in \mathbb{R}^2$ on the image $\boldsymbol{X}$, the intensity is represented as $\boldsymbol{X}_{\boldsymbol{x}}$. Also, let $\mathcal{T}$, T_i, and n_t indicate the random forest, the i-th decision tree, and the number of trees in the forest.

$$\mathcal{T} = \{T_i | i \leq n_t \text{ and } i \in \mathbb{Z}^+\}. \tag{1}$$

B. Unconstrained Representation

Learning optimal representation is essential to achieve higher accuracy and to avoid unnecessary use of computation and memory. Hence, we design an unconstrained representation that is able to learn optimal weights, shapes, and sparsities. In the proposed framework, the unconstrained representation's weights, shapes, and sparsities are learned at each node to split data points of different classes to separate child nodes as shown in Fig. 1. Moreover, the representation extracts depth and shift invariant features by compensating the associated changes as described in Fig. 3. Hence, the proposed novel representation improves accuracy and reduces computational complexity and memory usage.

Given an input data (spatial location) $\boldsymbol{x}$ and the corresponding input image and depth map ($\boldsymbol{X}$, $\boldsymbol{D}$), the feature $f(\cdot)$ is defined as follows:

$$f(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D}) = \sum_{i=1}^{n_f} w_i \boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}_i/\boldsymbol{D}_{\boldsymbol{x}},h} \tag{2}$$

where n_f is the number of data points used to compute each feature; w and $\boldsymbol{u} \in \mathbb{R}^2$ are a weight and an offset parameter vector, respectively; h is the channel index of $\boldsymbol{X}$.

In details, the offset data point $\boldsymbol{x} + \boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}}$ (the other end-point from a red circle $\boldsymbol{x}$ in Fig. 3) is determined based on $\boldsymbol{x}$, $\boldsymbol{u}$, and $\boldsymbol{D}_{\boldsymbol{x}}$. The weight w controls the influence of the information at the offset data point $\boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}}}$. The feature response is computed by the weighted summation of the offset

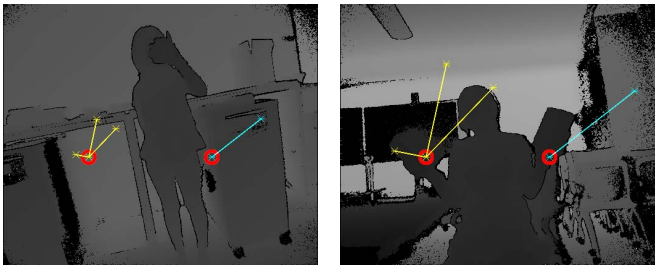


Fig. 3. Examples of the proposed feature representation. Each figure shows two different features where the same color line denotes the same feature. Each red circle represents a data point $\boldsymbol{x}$. The length of each line denotes the magnitude of each offset $\boldsymbol{u}_i/\boldsymbol{D}_{\boldsymbol{x}}$ considering the depth $\boldsymbol{D}_{\boldsymbol{x}}$ at the data point $\boldsymbol{x}$. The end-point of each line shows each offset point $\boldsymbol{x}+\boldsymbol{u}_i/\boldsymbol{D}_{\boldsymbol{x}}$ used to compute the corresponding feature. The feature is computed by Eq. (2) where the training algorithm in III-C learns weights (coefficients) w, offsets $\boldsymbol{u}$, and the number n_f of data points.

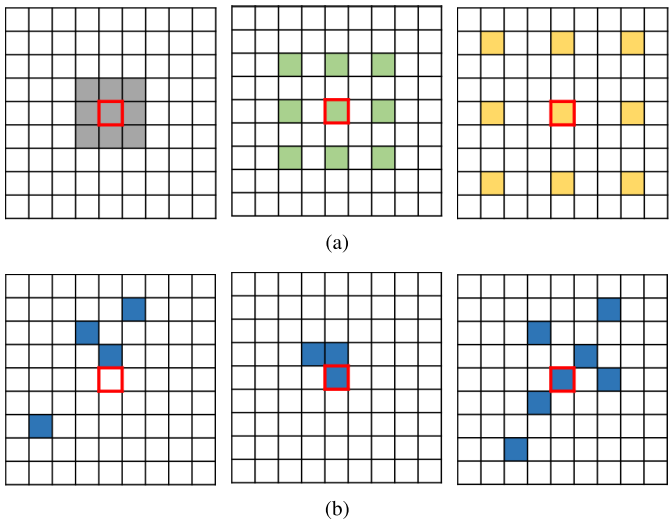


Fig. 4. Examples of the proposed representation and the representation of a convolutional layer in convolutional neural networks. (a) Feature in convolution layer. (b) Proposed feature. The proposed representation is not constrained by a specific shape. Color squares denote offset points $\boldsymbol{x}+\boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}}$ used to compute the feature, and the red bounding boxes represent the data points $\boldsymbol{x}$.

data points. All the parameters including the number n_f of data points, the weights w, the offsets $\boldsymbol{u}$, and the channel h are learned in the training procedure in Section III-C1. Each learned representation consists of $4 \times n_f + 1$ parameters where the factor 4 includes a weight, two offset parameters, and a channel index and the addition of 1 is for the number of data points. The proposed representation is described in Figs. 3 and 4.

If any offset data point $\boldsymbol{x} + \boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}}$ is beyond the boundary of the image, the intensity $\boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}}}$ is replaced by a constant (the maximum intensity of the input image $\boldsymbol{X}$ if $\boldsymbol{X}$ is a depth map; 0 otherwise).

$$\boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}}} = \begin{cases} \boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}}} & \text{if } \boldsymbol{x}+\boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}} \in \mathbb{Z}^+ \text{ and} \\ & \quad \boldsymbol{x}+\boldsymbol{u}/\boldsymbol{D}_{\boldsymbol{x}} \leq (p,q), \\ \max(\boldsymbol{X}) & \text{else if } \boldsymbol{X} \text{ is a depth map,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Obviously, when an input image $\boldsymbol{X}$ is a depth map $\boldsymbol{D}$, the feature can be expressed as follows:

$$f(\boldsymbol{x}, \boldsymbol{X}) = \sum_{i=1}^{n_f} w_i \boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}_i/\boldsymbol{X}_{\boldsymbol{x}}}. \tag{4}$$

Additionally, if depth information is not available, the requirement of depth data can be relaxed by sacrificing the property of depth invariance. The relaxed feature is defined as follows:

$$f(\boldsymbol{x}, \boldsymbol{X}) = \sum_{i=1}^{n_f} w_i \boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}_i}. \tag{5}$$

C. Random Forest

Random forest consists of a set of decision trees as shown in Figs. 1 and 2. It is robust to noisy and irregular data because of the combination of multiple trees with varying features. It is also computationally less complex than typical neural networks since, in the inference procedure, each input data is processed using only small portion of the forest based on the conditions in ancestral nodes (see blue dotted lines in Figs. 1 and 2).

Each decision tree in random forest is composed of a root node, splitting nodes, and leaf nodes (see Fig. 2). The input of each tree is the location $\boldsymbol{x} \in \mathbb{R}^2$ of an input data and the corresponding input image and depth map $(\boldsymbol{X}, \boldsymbol{D})$. Given the input $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D})$ at a root node, the input data is classified to a child node based on the splitting criteria $f_n(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D}) \lessgtr \theta$ where $f_n(\cdot)$ extracts a learned feature at the node n. The classification to a descendant node is terminated when the input data reaches a leaf node. At the leaf node, the conditional probability $p(c|f_n(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D}) \lessgtr \theta$ for $\forall n$ until the leaf node) of being each class c is learned in a training stage and is used in an inference stage. The leaf nodes are also learned in a training procedure. For more details about random forest, we refer readers to [36]–[38] and [42].

1) Training: In a training stage, the random forest learns a splitting criteria $f_n(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D}) \lessgtr \theta$ at each splitting node n and a conditional probability distribution $p(c|f_n(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D}) \lessgtr \theta$ for $\forall n$ until the leaf node) of being each class c at each leaf node. The splitting criteria at a node n is denoted as follows:

$$f_n(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D}) = \sum_{i=1}^{n_f} w_i \boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}_i/\boldsymbol{D}_{\boldsymbol{x}}} \lessgtr \theta. \tag{6}$$

In (6), one redundant parameter can be eliminated by dividing each side by θ while the data split remains equivalent.

$$\sum_{i=1}^{n_f} \frac{w_i}{\theta} \boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}_i/\boldsymbol{D}_{\boldsymbol{x}}} \lessgtr 1,$$
$$\sum_{i=1}^{n_f} w_i' \boldsymbol{X}_{\boldsymbol{x}+\boldsymbol{u}_i/\boldsymbol{D}_{\boldsymbol{x}}} \lessgtr 1 \text{ where } w_i' = \frac{w_i}{\theta}. \tag{7}$$

Thus, the training algorithm only needs to learn the parameters in the representation function $f(\cdot)$ while the splitting boundary is always 1. In the rest of this paper, a weight w represents w'.

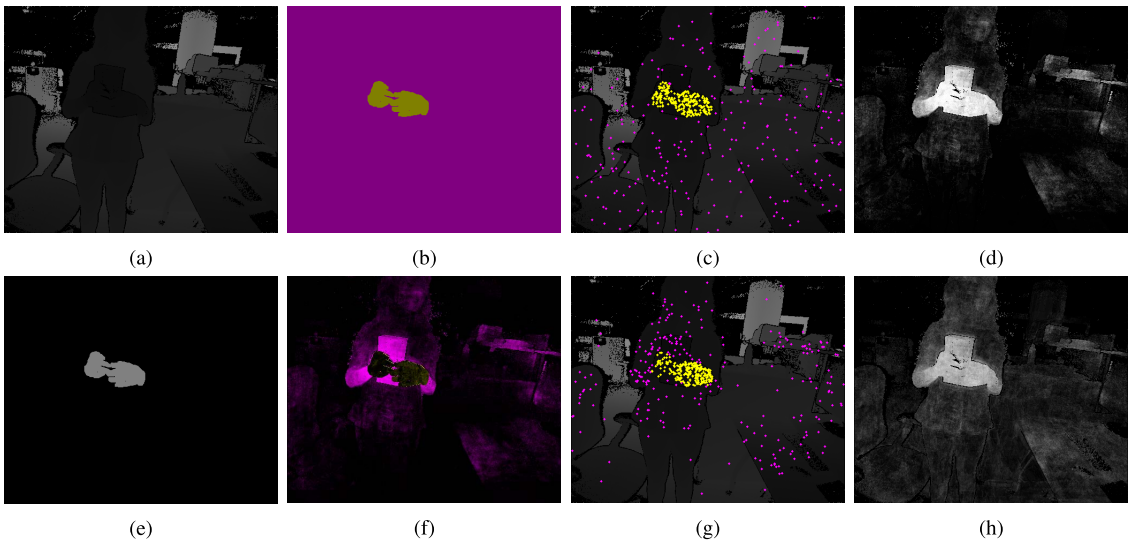


Fig. 5. Example of boosting and bootstrap aggregating. (a) Depth map. (b) Sampling probability $p_s(\cdot)$ in the first iteration of boosting. (c) Sampled data points in the first iteration. (d) Inferred probability using the first set of learned trees. (e) Ground truth label. Gray and black color denote hand and non-hand class, respectively. (f) Sampling probability $p_s(\cdot)$ in the second iteration of boosting. (g) Sampled data points in the second iteration. (h) Inferred probability using the first and the second set of trees. Magenta and yellow color represent non-hand and hand class, respectively. Brighter color in (b) and (f) denotes higher sampling probability $p_s(\cdot)$. Brighter color in (d) and (h) represents the higher probability of being hand class. The number of sampled data points per image is 250 for each class in this example.

In the following paragraphs, we introduce the strategy of selecting training data using bootstrap aggregating and boosting. We then present the method of learning representations using particle swarm optimization. Finally, we describe the condition for determining leaf nodes and the process of learning conditional probability distribution.

Bootstrap Aggregating: Bootstrap aggregating (bagging) is to amalgamate multiple classifiers trained using randomly selected training data sets. This method improves robustness and accuracy by integrating multiple classifiers with variance caused by the randomly selected training data. Given the training data sets $\mathcal{X}$ of images, the bagging algorithm selects random sets $\mathcal{X}_i \subset \mathcal{X}$ of images for the i-th tree. Then, the algorithm samples n_d data points (pixels) for each class from the images $\boldsymbol{X} \in \mathcal{X}_i$. Thus, each tree is trained using $n_d \times n_c$ data points where n_c is the total number of classes.

For hand segmentation, we sample the same number of data points for each image and for each class to consider diverse images equally regardless of the sizes of hands, etc. For semantic segmentation, since all objects do not appear on all the images, we only constrain the same number of data points for each class.

Boosting: Boosting is to combine a set of weak classifiers to form a stronger classifier. Especially, adaptive boosting is learning weak classifiers based on the obtained result by applying previously trained weak classifiers [54]. We apply adaptive boosting at each set of trees by adjusting the sampling of training data based on the segmentation result at the stage.

As explained in the bagging, a random set $\mathcal{X}_i \subset \mathcal{X}$ of images is selected for the i-th tree T_i. Then, n_d data points $\boldsymbol{x}$ are sampled for each class c. The proposed boosting algorithm adjusts this sampling of data points $\boldsymbol{x}$ for each set of trees. For the first set of trees, data points $\boldsymbol{x}$ are sampled randomly with uniform distribution for each class. Thus, the probability $p_s(\cdot)$ of being sampled for a data point $\boldsymbol{x}$ of a class c is as follows:

$$p_s(\boldsymbol{x}) = \frac{n_d}{n_{d_c} n_{\boldsymbol{X}}} \tag{8}$$

where n_{d_c} is the number of the data points of a class c in the image $\boldsymbol{X}$ for hand segmentation and in the set $\mathcal{X}_1$ of images for semantic segmentation, respectively. $\boldsymbol{X}$ is the image that $\boldsymbol{x}$ belongs to. $n_{\boldsymbol{X}}$ is 1 for semantic segmentation and is the number of images in $\mathcal{X}_1$ for hand segmentation.

After training the first set of trees, the sampling probability $p_s(\cdot)$ is updated to train the second set of trees more effectively. It is achieved by adjusting $p_s(\cdot)$ to sample more data points with higher errors. Hence, $p_s(\cdot)$ is increased for the data points with a high error and is decreased for the data with a low error. Given the learned set of trees $\mathcal{T}$ until the current iteration, an inference is processed to estimate the probability $p_c(c|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D})$ of being a target class c for the data in the training dataset $\mathcal{X}$. Then, $p_s(\cdot)$ is updated as follows:

$$p_s(\boldsymbol{x}) = 1 - p_c(c|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D}). \tag{9}$$

By using the data points sampled based on the updated $p_s(\cdot)$, the next set of trees is trained. $p_s(\cdot)$ is adjusted repeatedly at each iteration until the training terminates.

Fig. 5 shows a visual example of boosting and bootstrap aggregating. The first row and the second row show the first iteration and the second iteration of boosting, respectively. In the first iteration, $p_s(\cdot)$ in (b) is the same for all data points of the same class. Thus, the sampled data points in (c) are distributed uniformly. In the second iteration, $p_s(\cdot)$ in (f) varies for each data point depending on the inferred probability in (d) that is obtained using the first set of trees. Consequently, the sampled data points in (g) are distributed based on $p_s(\cdot)$ in (f).

Particle Swarm Optimization: Particle swarm optimization (PSO) is applied to learn a feature representation that splits the data points of different classes into separate child nodes at each splitting node. PSO is selected to find a more optimal solution in a high-dimensional parameter space ($\mathbb{R}^{3\times n_f}\cdot\mathbb{Z}^1$ if $\boldsymbol{X}$ is a depth map, and additional $\mathbb{Z}^{n_f}$ if $\boldsymbol{X}$ is a color image).

Assuming $\boldsymbol{X}$ is a depth map, a representation parameter vector $\boldsymbol{p}\in\mathbb{R}^{3\times n_f}\cdot\mathbb{Z}^1$ consists of the number $n_f\in\mathbb{Z}^+$ of data points and a weight $w\in\mathbb{R}$ and an offset parameter vector $\boldsymbol{u}\in\mathbb{R}^2$ for each data point. Thus, the total number of parameters is $3\times n_f+1$. To limit the solution space ($\mathbb{R}^{3\times n_f}\cdot\mathbb{Z}^1$), the maximum number of data points n_f is chosen as 9 which is equivalent to the number of data points in a filter (kernel) with the size of 3×3. It is also the most common size of a kernel in convolutional neural networks.

In the first iteration, the algorithm generates 100 offset candidates $\boldsymbol{u}$ and 100 weight candidates w where the candidates are sampled from the uniform distribution. Then, the algorithm tries the combinations of the 9 different numbers n_f of data points, 100 offset candidates $\boldsymbol{u}$, and 100 weight candidates w, totaling 90,000 particles (candidates). By applying the particles, the optimization algorithm learns the global best state $\boldsymbol{q}_g$ that is the best solution among the entire particles. To simplify and expedite training, the number n_f of data points is decided as the number of data points of the global best state $\boldsymbol{q}_g$ in the first iteration. Also, among 10,000 particles, 100 particles are chosen by selecting the best weight candidate for each offset candidate. It is feasible since 10,000 particles are generated by the combinations of 100 offset candidates $\boldsymbol{u}$ and 100 weight candidates w.

From the second iteration, the 100 particles are used to find the optimal solution. Let $\boldsymbol{q}\in\mathbb{R}^{3\times n_f}$ represent each particle consisting of weights w and offsets $\boldsymbol{u}$. At an iteration t, the particles $\boldsymbol{q}^t$ are first updated using the particles $\boldsymbol{q}^{t-1}$, the personal best states $\boldsymbol{q}_p^{t-1}$, and the global best state $\boldsymbol{q}_g^{t-1}$ at the previous iteration $t-1$. The personal best state $\boldsymbol{q}_p^t$ is the best state of each particle until the current iteration t. The global best state $\boldsymbol{q}_g^t$ is the best state among all the particles until the current iteration t. The particles at an iteration t are updated as follows:

$$\begin{aligned}\boldsymbol{q}_{p,i}^{t-1} &= \{\widetilde{\boldsymbol{q}_i^t}|\widetilde{t}=\underset{t}{\operatorname{argmin}}\,L(q_i^t)\},\\ \boldsymbol{q}_g^{t-1} &= \{\widetilde{\boldsymbol{q}_i^t}|(\widetilde{i},\widetilde{t})=\underset{i,t}{\operatorname{argmin}}\,L(q_i^t)\},\\ \boldsymbol{q}_i^t &= \boldsymbol{q}_i^{t-1}+\alpha_p(\boldsymbol{q}_{p,i}^{t-1}-\boldsymbol{q}_i^{t-1})+\alpha_g(\boldsymbol{q}_g^{t-1}-\boldsymbol{q}_i^{t-1})\end{aligned}\quad(10)$$

where i denotes the index for each particle. $L(\cdot)$ is the objective function in (12). α_p and α_g are the weights towards the personal best state $\boldsymbol{q}_p$ and the global best state $\boldsymbol{q}_g$. Both parameters (α_p and α_g) are randomly generated from the Normal distribution $\mathcal{N}(\mu,\sigma^2)$ as follows:

$$\begin{aligned}\widetilde{\alpha} &\sim \mathcal{N}(1.0, 0.25),\\ \alpha &= \max(0,\widetilde{\alpha})\end{aligned}\quad(11)$$

where μ and σ represent a mean and a standard deviation. The iteration of PSO is terminated when the loss $L(\cdot)$ is not decreased at each iteration or after maximum 100 iterations.

The objective function $L(\cdot)$ consists of a term for classification loss and two terms for regularization. The classification loss is to evaluate a representation's ability of separating the data points of different classes to separate child nodes. The regularization is to prefer smaller weights w and the smaller number n_f of data points.

$$\begin{aligned}L(\boldsymbol{p}) &= \underbrace{C(\boldsymbol{p})}_{\text{classification loss}} + \underbrace{\lambda_w\sum_{i=1}^{n_f}{w_i}^2+\lambda_{n_f}n_f}_{\text{regularization}}\\ &= \underbrace{-\sum_h\sum_c\frac{n(h)}{\sum_h n(h)}p(c|h)\log p(c|h)}_{\text{classification loss}}\\ &\quad+\underbrace{\lambda_w\sum_{i=1}^{n_f}{w_i}^2+\lambda_{n_f}n_f}_{\text{regularization}}\end{aligned}\quad(12)$$

where h is an index for a child node (e.g. left child, right child); c is an index for a class; $n(h)$ denotes the number of data points in a child node h; $p(c|h)$ is the probability of being the class c at the node h; λ_w and λ_{n_f} are weights for each regularization term.

After learning an optimal representation at a splitting node, the data is split into child nodes. If the child node does not meet the condition for becoming a leaf node, learning representation and splitting to child nodes are repeated. Otherwise, splitting is terminated, and a leaf node is formed.

Leaf Node: A leaf node is formed based on the following criterias: (1) the maximum depth of a tree, (2) the probability distribution $p(c|h)$, and (3) the number of training data $\boldsymbol{x}$ at the node. In details, a leaf node is generated if (1) the current depth of a tree is deeper than the maximum depth; (2) the probability at the node is considerably confident for a class; (3) the number of remaining training data is too small. When a leaf node is formed, the conditional probability is stored for inference processing. The conditional probability $p(c|f_n(\boldsymbol{x},\boldsymbol{X},\boldsymbol{D})\lesseqgtr\theta$ for $\forall n$ until the leaf node) is computed using the number of data points for each class at the leaf node h.

$$\begin{aligned}&p(c|f_n(\boldsymbol{x},\boldsymbol{X},\boldsymbol{D})\lesseqgtr\theta \text{ for } \forall n \text{ until the leaf node})\\ &\qquad= p(c|h)=\frac{n(h,c)}{\sum_{c=1}^{n_c}n(h,c)}\end{aligned}\quad(13)$$

where $n(h,c)$ represents the number of data points for the class c at the leaf node h.

2) Inference: Given the trained random forest $\mathcal{T}$, each data point $\boldsymbol{x}$ on an image $\boldsymbol{X}$ is classified to child nodes using each tree until it reaches a leaf node. When it reaches a leaf node using each tree T_i, the learned conditional probability distribution $p_{T_i}(c|f_n(\boldsymbol{x},\boldsymbol{X},\boldsymbol{D})\lesseqgtr\theta$ for $\forall n$ until the leaf node) of being each class c at the leaf node is loaded. Then, the conditional probability distributions from the entire trees $T_i\in\mathcal{T}$ are averaged to estimate the inferred probability

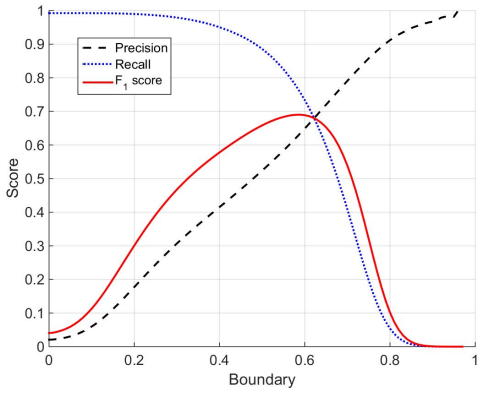


Fig. 6. Scores depending on decision boundary. The scores are computed on the validation set of hand segmentation data using the proposed method (30 trees) without the bilateral filtering.

$p_c(c|\boldsymbol{x})$ of the data point $\boldsymbol{x}$ being a class c.

$$p_c(c|\boldsymbol{x}) = \frac{1}{n_t}\sum_{i=1}^{n_t} p_{T_i}(c|f_n(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{D}) \lessgtr \theta \quad \text{for } \forall n \text{ until the node}) \tag{14}$$

where n_t is the number of trees in the random forest $\mathcal{T}$.

Modified Bilateral Filter: Since the probability $p_c(c|\boldsymbol{x})$ is predicted for each pixel independently, the probability can be stabilized considering nearby predictions [7], [55]. In this paper, we apply simple modified bilateral filter [7], [56] that processes weighted averaging of the probabilities of the data points in close distance and similar intensity on the input $\boldsymbol{X}$. The filtering is defined as follows:

$$\widetilde{p}_c(c|\boldsymbol{x}) = \frac{1}{N}\sum_{\boldsymbol{x}_i \in \Omega} g_r(|\boldsymbol{X}_{\boldsymbol{x}_i} - \boldsymbol{X}_{\boldsymbol{x}}|) g_s(\|\boldsymbol{x}_i - \boldsymbol{x}\|) p_c(c|\boldsymbol{x}_i) \tag{15}$$

where $\boldsymbol{\Omega}$ is the set of pixels within the filter's radius and the input difference; N is the normalization term.

$$N = \sum_{\boldsymbol{x}_i \in \Omega} g_r(|\boldsymbol{X}_{\boldsymbol{x}_i} - \boldsymbol{X}_{\boldsymbol{x}}|) g_s(\|\boldsymbol{x}_i - \boldsymbol{x}\|). \tag{16}$$

$g_r(\cdot)$ and $g_s(\cdot)$ are the Gaussian functions for an input difference r and a spatial distance s, respectively.

$$g_r(r) = \exp\left(-\frac{r^2}{2\sigma_r^2}\right), \quad g_s(s) = \exp\left(-\frac{s^2}{2\sigma_s^2}\right). \tag{17}$$

By experimenting on the validation dataset, the maximum color and depth difference to consider are 100 and 400mm, respectively. Both the standard deviations (σ_r and σ_s) are 100.

Decision Boundary: Given the filtered probability $\widetilde{p}_c(c|\boldsymbol{x})$, a decision boundary needs to be determined for classification. Although the most common method is choosing the class c with the highest probability $\widetilde{p}_c(c|\boldsymbol{x})$, it is not guaranteed to be the best solution. Thus, for hand segmentation, the possible boundaries are tested with the step size of 0.01 using the validation dataset. Fig. 6 shows the F_1 score, precision, and recall on the validation dataset depending on the decision boundary. The scores are computed using the proposed method with 30 trees and without the bilateral filtering. The best F_1 score is achieved at the decision boundary of 0.59. For semantic segmentation, the class with the highest probability is selected considering the high complexity caused by the high dimensional decision boundary.

IV. Experiments and Results

The proposed method is applied to three applications: road scene semantic segmentation, hand segmentation for hand-object interaction, and indoor scene semantic segmentation. The experimental results demonstrate that the proposed method outperforms typical random forest by learning unconstrained representations using particle swarm optimization and processes efficiently comparing to neural network-based methods.

For comparison, we report mean intersection over union (IU) per category and per class for road scene semantic segmentation dataset and precision, recall, and F_1 score for hand segmentation for hand-object interaction. Let n_{ij} be the number of pixels that belong to the class i and are predicted to the class j, and n_c be the total number of classes or categories.

$$\begin{aligned} \text{IU} &= \frac{1}{n_c}\sum_i \left(\frac{n_{ii}}{\sum_j n_{ij} + \sum_j n_{ji} - n_{ii}}\right), \\ \text{Precision} &= \frac{n_{11}}{n_{11} + n_{01}}, \\ \text{Recall} &= \frac{n_{11}}{n_{11} + n_{10}}, \\ F_1 &= \frac{2n_{11}}{2n_{11} + n_{01} + n_{10}}, \end{aligned} \tag{18}$$

where for hand segmentation, classes 1 and 0 denote "hand" and "others", respectively.

For quantitative comparison of efficiency, we measure processing time using a machine with Intel i7-4790K CPU, 16.0GB RAM, and NVIDIA GeForce GTX 770 for hand segmentation and the same machine with NVIDIA Tesla K40c for semantic segmentation.

A. Hand-Object Interaction (HOI)

1) Dataset: We experiment using the publicly available HOI dataset [14] which consists of 27,525 pairs of depth maps and ground truth labels. The dataset was collected from 6 people (3 males and 3 females) interacting with 21 different objects using Microsoft Kinect v2 camera. The dataset also includes the cases of one hand and both hands in a scene. We follow standard dataset split (19,470 pairs for training, 2,706 pairs for validation, and 5,349 pairs for testing).

2) Experiments: We trained 30 trees using the proposed method. Each tree is trained with 2,000 pairs of depth maps and ground truth labels and 500 data points from each image. After training every 10 trees, we applied boosting by computing errors and by updating sampling probability based on the errors. The condition for becoming a leaf node was 0.99 for class probability, 0.0001 for remaining data portion, and 25 for maximum depth.

TABLE I
THE QUANTITATIVE RESULTS OF THE HOI DATASET

Method	# of trees	Decision boundary	Bilateral filter	Precision	Recall	F_1 score	Time (ms) GTX 770
RF [2], [6]	20	0.50	-	37.9	91.9	53.7	10
	30	0.50	-	37.9	92.1	53.7	14
RF [2], [6] + DB adjustment	20	0.79	-	54.9	72.7	62.5	10
	30	0.79	-	54.8	73.0	62.6	14
FCN-32s [9]	-	-	-	70.0	68.6	69.3	376
FCN-16s [9]	-	-	-	68.0	72.2	70.1	376
FCN-8s [9]	-	-	-	70.4	74.4	72.3	377
Frontend [10]	-	-	-	72.4	70.2	71.3	718
Proposed	20	0.50	-	50.7	89.4	64.7	28
	20	0.59	-	62.2	76.3	68.5	28
	30	0.50	-	53.4	88.4	66.5	39
	30	0.59	-	64.7	75.7	69.8	39
	30	0.59	5×5	65.3	76.0	70.3	41

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

Fig. 7. The qualitative comparison of the result for the HOI dataset. (a) Results of the random forest [2], [6]. (b) Results of the random forest [2], [6] with the decision boundary adjustment in III-C2. (c) Results of the FCN-8s model [9]. (d) Results of the proposed method. (e) Ground truth labels. The results and the ground truth labels are visualized on the depth maps with different color channels for better visualization.

3) Results: The quantitative results and the qualitative results are shown in Table I and Fig. 7, respectively. The proposed method achieves about 31% and 12% relative improvement in F_1 score comparing to the typical RF-based method [2], [6] and its combination with the decision boundary adjustment in Section III-C2. Comparing to the deep learning-based methods [9], [10], it achieves quite competitive results (3% lower than the best method) in F_1 score. Also, the processing time of the proposed method is about 9 times faster than those of the deep learning-based methods [9], [10]. Fig. 8 shows the analysis of the methods in accuracy and efficiency. In Table II, we show empirical results of changing the maximum depth in the proposed random forest. In this analysis, the filter-size of bilateral filtering is 5×5.

B. Semantic Segmentation (Cityscapes)

1) Dataset: The Cityscapes dataset contains the images of urban street scenes [15]. The dataset consists of 5,000 finely annotated images and 19,998 sparsely annotated images. We train models for the standard 19 classes problem using the standard data separation of 2,975 finely annotated images and 19,998 sparsely annotated images for training, 500 images for validation, and 1,525 images for testing.

2) Experiments: We trained five trees using the proposed framework. Each tree is trained with [12,974, 12,973, 7,658, 7,656, 7,658] pairs of images and ground truth labels. The first two trees are trained using the entire finely annotated images and the half of sparsely annotated images. The last three trees are trained with 1/3 of entire training data sets. The selection

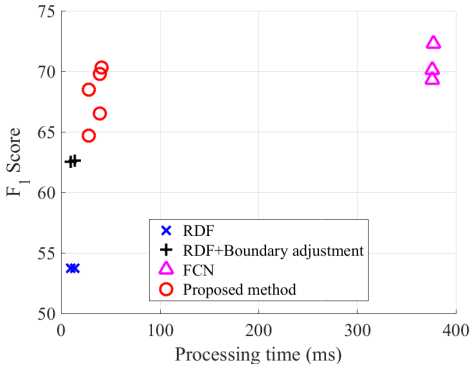


Fig. 8. Analysis in accuracy and efficiency.

TABLE II

ANALYSIS OF THE MAXIMUM DEPTH IN THE PROPOSED RANDOM FOREST USING THE HOI DATASET

Maximum depth	# of nodes	Score			Time (ms)	GPU (MB)
		Precision	Recall	F_1 score		
1	3	10.9	83.5	19.4	0.67	70
5	46	36.4	54.2	43.6	0.77	71
10	1157	43.5	66.5	52.6	0.96	71
15	15440	50.2	73.2	59.6	1.12	73
20	57176	56.9	74.0	64.4	1.22	81
25	100523	59.8	73.8	66.1	1.26	90

of the number of images is based on both experimental and intuitive choice. After training each tree, we applied boosting by sampling the set of data points for the next tree based on the current predictions. The condition for becoming a leaf node is 0.99 for class probability, [0.000001 or 0.000002] for remaining data portion, and 25 for maximum depth.

Although predicting using random forest is a computationally and memory efficient process, training large forest with a huge amount of data is computationally expensive. It is especially time-consuming during training procedure since each node needs to be optimized conditioned on ancestor nodes where the number of nodes can be up to 67,108,863 considering the maximum depth of 25. Hence, we improve the training algorithm to reduce learning time.

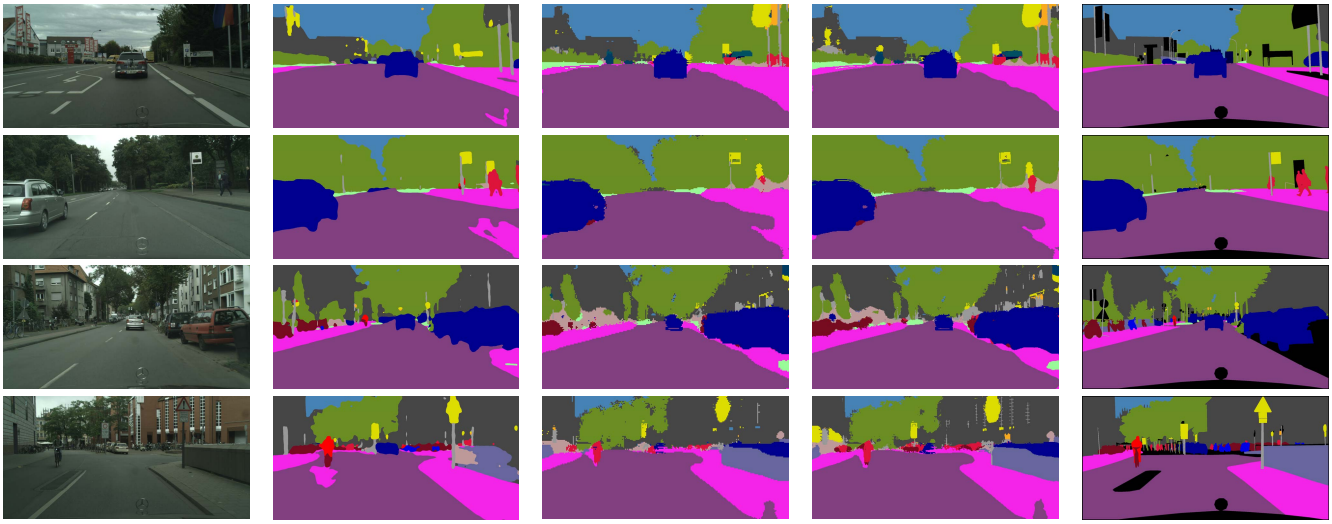
One of the most time-consuming processes is loading training images repeatedly. We used over 7,656 images to train each tree where the original resolution of each image is 2048×1024. The required memory to hold 7,656 color images is 7,656×2,048×1,024×3 bytes (44.9 GB) which are not accessible in modern single GPU. One option is loading partial sets of images multiple times at each node. However, it is a considerably time-consuming process since loading 1,000 images from a hard disk drive takes 51 seconds and repeating at 1,000 nodes demands 14 hours. Hence, we propose to decrease the resolution of images to hold the entire set of training data in memory after loading once for a tree. Specifically, we resized images to 512×256 so that the required memory to hold 12,974 color images is 4.8 GB. Given 11 GB in modern single GPU, the rest of memory is utilized to process learning algorithms.

Using the remaining memory, we were able to load and process 103,792/382,900 samples for 12,974/7,658 color images where each sample consists of frame index, $\boldsymbol{x}$, $\boldsymbol{D_x}$, and label. However, this number of samples is too small since the average number of data points at depth 15 becomes 3.2 (103,792 / 2^{15}) and 11.7 (382,900 / 2^{15}). Hence, we employ multiple sets of data points (specifically, 64 sets) so that the average number at depth 15 becomes 202.7 and 747.9 data points. We start training with 64 sets at a root node and merge the sets at a deeper node since the number of data points in a set decreases as the data splits to child nodes. While multiple sets are employed to provide enough data, we use a single set (possibly, merged from multiple sets) among them to train a splitting node in order to reduce computation and data transfer time.

3) Results: We compare the proposed method to deep learning-based methods [9], [11], [45], [47], [48], [57]–[61] in Table III and Fig. 9. We also show the results of the proposed method with various conditions on the validation set in Table IV. We report mean IU per category and per class, processing time, and memory usage. Since optimizing post-processing is beyond the scope of this paper, we report processing time and memory usage excluding the demand for bilateral filtering. The overall results show that the proposed method processes each image using small computation and memory resources while achieving meaningful precision. It demonstrates that the proposed method can be applied for real-time semantic segmentation. It can also be employed in a low-end GPU or embedded system that demand small power and memory consumption.

We show the results of applying the same random forest model to the inputs of varying resolutions using the validation set in Table IV. The results demonstrate that the proposed method is robust to scaled input images by simply adjusting the sparsities in the learned representations. It is feasible since the sparsity in the proposed representation is defined in floating-point precision. Table VI shows the results of applying the FCN model trained with the input resolution of 2048×1024 to the inputs of different resolutions using the validation set. The accuracy degrades significantly since deep learning-based methods utilize integer-point precision (mostly, 1) for sparsity.

4) Analysis: We demonstrate the effectiveness of the proposed unconstrained representation and the particle swarm optimization comparing to typical convolution filter and random search [2], [6], [7] in Table V. By comparing the unconstrained representation and the convolution filter, the decision tree trained using the unconstrained representation achieves higher accuracy than the tree with the typical convolution filter even with a smaller number of nodes, shorter processing time, and lower GPU memory usage in both optimization methods. For the typical convolution filter, 3×3×3 convolution filters are used to take inputs from 3×3 spatial regions and entire color channels (total 27 parameters). The analysis between particle swarm optimization and random search shows that the model trained with particle swarm optimization outperforms the other model optimized with random search while complexity (the number of nodes, processing time, and memory usage) is similar. It verifies that more optimal representation can be learned by using the unconstrained representation and



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

Fig. 9. The qualitative comparison of the result for the Cityscapes dataset [15]. (a) Input image. (b) Results of the FCN-8s model [9], [45]. (c) Results of the proposed method using five trees, bilateral filter of 13×13, and the input resolution of 256×128. (d) Results of the proposed method using five trees, bilateral filter of 23×23, and the input resolution of 512×256. (e) Ground truth labels.

TABLE III
THE QUANTITATIVE RESULTS OF THE CITYSCAPES DATASET

Method			Input resolution	Accuracy (IU)		Time (ms)	GPU memory (MB)
Method	# of trees	Bilateral filter		Category	Class		
OCNet [57]	-	-	Multiscale	91.6	81.7	58870	21994
RefineNet [58]	-	-		87.9	73.6	22950	17647
ICNet [59]	-	-		-	70.6	212	1549
CCNet [60]	-	-	2048×1024	-	81.4	9233	12016
PSPNet [61]	-	-		90.6	80.2	15254	63288
FCN-8s [9], [45]	-	-		85.7	65.3	1365	5800
CRF as RNN [11]	-	-		82.7	62.5	10704	31836
ENet [48]	-	-		80.4	58.3	2966	284
SegNet basic [47]	-	-		79.1	57.0	1392	4564
SegNet extended [47]	-	-		79.8	56.1	1992	9158
Proposed	5	23×23	512×256	60.2	30.0	30	1271

the particle swarm optimization. In these analyses, the resolutions of input images were 512×256 in training as explained in Section IV-B2

In Table VII, we show empirical results to decide hyperparameters for particle initialization in particle swarm optimization using the validation set. The hyperparameters are to set the ranges for the uniform distributions described in Section III-C1. Although evolved particles can move outside of the initial boundary, initializing particles in a desirable range is important to search solution space efficiently using the limited number of particles and computational resources.

In Table VIII, we show experimental analysis of adjusting the maximum depth in the proposed method. In this analysis, the input resolution and the filter-size of bilateral filtering are 512×256 and 19×19, respectively. The effects of varying the maximum depth in different datasets can be observed using Tables II and VIII.

We present per-category accuracy in Table IX. This analysis shows that the result of the proposed method is competitive for the categories of flat, sky, and nature comparing to CNN-based methods. However, the accuracy difference is greater for the categories of object and human that include relatively small and sparse things such as traffic light, traffic sign, and rider (also see Fig. 10). We believe it is greater for the categories because of image resolution and random sampling. First, we train and test using images with 512×256 resolution because of the memory limit during training. This scaling makes small objects even tinier. Consequently, it is challenging to classify them in pixel-level during testing. Moreover, because of randomly sampling from resized images without data augmentation, the variety of data points for small object classes is limited. Accordingly, the trained forest has limited generality for new data.

While the proposed method has advantages in computational complexity, memory demand, and ability to control complexity by adjusting the number of trees and the input resolution, the accuracy of the proposed method is limited comparing to deep learning-based methods. We discuss three possible explanations. First of all, while deep neural networks can

TABLE IV

ANALYSIS OF THE PROPOSED METHOD USING THE VALIDATION SET OF THE CITYSCAPES DATASET

Method	# of trees	Bilateral filter	Input resolution	Accuracy (IU) Category	Accuracy (IU) Class	Time (ms)	GPU memory (MB)
Proposed	1	-	2048×1024	40.0	16.9	75	329
		15×15		50.0	23.0		
		91×91		53.3	25.5		
	5	-		45.9	20.5	385	1319
		13×13		52.8	25.1		
		91×91		54.6	26.4		
	1	-	1024×512	41.1	17.4	21	290
		15×15		52.0	24.4		
		55×55		53.6	25.7		
	5	-		46.8	21.0	106	1280
		13×13		54.1	26.0		
		43×43		55.0	26.7		
	1	-	512×256	41.8	17.8	6	281
		15×15		53.3	25.4		
		27×27		53.7	25.8		
	5	-		47.4	21.4	30	1271
		13×13		54.9	26.6		
		23×23		55.1	26.8		
	1	15×15	256×128	52.8	25.2	2	279
	5	13×13		54.2	26.3	8	1269

TABLE V

ANALYSIS ON LEARNING FEATURES USING THE PROPOSED UNCONSTRAINED REPRESENTATION AND THE PARTICLE SWARM OPTIMIZATION

Method Representation	Method Optimization	# of nodes	Bilateral filter	Accuracy (IU) Category	Accuracy (IU) Class	Time (ms)	GPU memory (MB)
Convolution filter (3×3)	Random search	1,068,413	-	29.1	10.4	149	561
			11×11	36.2	14.6		
			51×51	38.5	16.7		
			71×71	38.7	16.9		
Proposed unconstrained	Random search	855,974	-	31.5	11.3	102	398
			11×11	40.4	15.7		
			51×51	44.9	18.6		
			71×71	45.6	19.1		
Convolution filter (3×3)	Proposed PSO	1,170,745	-	29.1	10.4	152	603
			11×11	37.0	15.1		
			51×51	39.4	17.2		
			71×71	39.5	17.4		
Proposed unconstrained	Proposed PSO	767,953	-	37.2	14.9	83	371
			11×11	47.2	21.0		
			51×51	52.7	24.6		
			71×71	53.3	25.0		

TABLE VI

ANALYSIS ON APPLYING NEURAL NETWORKS TO INPUT IMAGES WITH DIFFERENT RESOLUTION

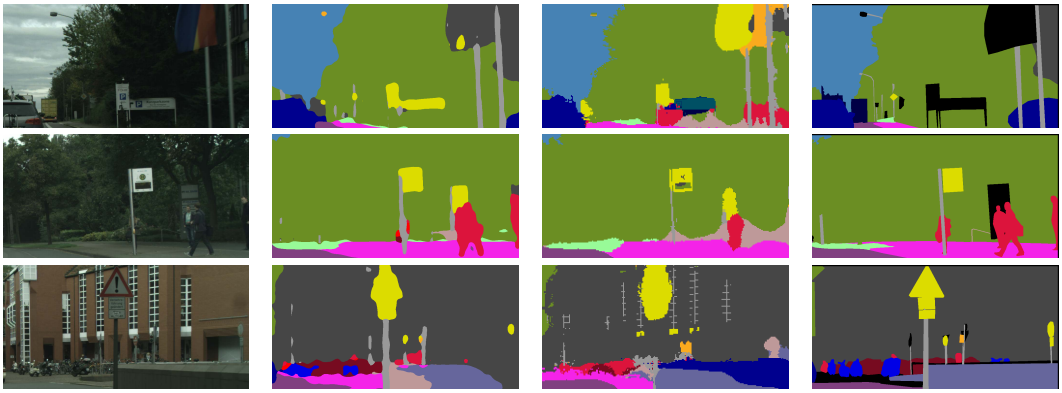
Method	Input resolution	Accuracy (IU) Category	Accuracy (IU) Class	Time (ms)	GPU (MB)
FCN-8s [9], [45]	2048×1024	84.0	64.0	1365	5800
	1024×512	77.9	57.4	468	2220
	512×256	66.9	42.3	216	1218
	256×128	50.8	23.0	162	911

TABLE VII

ANALYSIS ON HYPERPARAMETERS IN PARTICLE SWARM OPTIMIZATION

Hyperparameters Weight	Hyperparameters Offset	# of nodes	Accuracy (IU) Category	Accuracy (IU) Class
0.1	0.2	1,960	43.0	17.6
	0.3	1,984	43.1	17.6
0.2	0.1	1,942	42.5	16.8
	0.2	1,954	43.2	17.6
	0.3	1,988	**43.3**	**17.8**
	0.4	1,964	43.1	17.5
0.3	0.2	1,952	43.2	17.3
	0.3	1,950	43.2	17.7

describe nonlinear representations by using a hierarchical structure with nonlinear activation functions, the representation of the proposed method is a linear representation of the original input image at the root node (see (2)). Second, while deep neural networks are usually trained end-to-end by using back-propagation, the proposed method is trained in-order from a parent node to child nodes. As all the nodes are trained



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

Fig. 10. The qualitative comparison using enlarged results. Right-top parts of the images in Fig. 9 are visualized. (a) Input image. (b) Results of the FCN-8s model [9], [45]. (c) Results of the proposed method using five trees, bilateral filter of 23×23, and the input resolution of 512×256. (d) Ground truth labels.

TABLE VIII

ANALYSIS OF THE MAXIMUM DEPTH IN THE PROPOSED RANDOM FOREST USING THE CITYSCAPES DATASET

Maximum depth	# of nodes	Accuracy (IU)		Time (ms)	GPU (MB)
		Category	Class		
1	3	16.7	5.8	1.3	72
5	63	38.5	12.7	1.6	73
10	2037	47.4	19.3	2.2	74
15	53354	51.0	22.4	3.2	91
20	598317	53.9	25.3	4.6	291
25	2349554	56.1	27.1	6.9	938

TABLE IX

ANALYSIS OF ACCURACY PER CATEGORY

Category	Method		
	FCN-8s	ENet	Proposed
Flat	98.2	97.3	93.8
Nature	91.1	88.3	73.6
Object	57.0	46.8	15.7
Sky	93.9	90.6	84.6
Construction	89.6	85.4	61.0
Human	78.6	65.5	29.8
Vehicle	91.3	88.9	63.3

only once in-order, overall optimization might be relatively distant from global optima. Lastly, as Tables II and VIII show, increasing the size of a tree can improve accuracy at the cost of processing time and memory usage. However, training a deeper tree is quite computationally expensive since the number of nodes increases exponentially as the depth increases.

C. Semantic Segmentation (NYUDv2)

1) Dataset: The NYUDv2 dataset consists of 1,449 pairs of RGB-D images including various indoor scenes with pixel-wise annotations [20]. The pixel-wise annotations were coalesced into 40 dominant object categories by Gupta *et al.* [62]. We experimented with this 40 classes problem using the standard data separation [20], [62] of 795 training images and 654 testing images.

TABLE X

THE QUANTITATIVE RESULTS OF THE NYUDV2 DATASET

Method	Accuracy	Time (ms)	GPU memory (MB)
DeepLab [13]	63.8	365	1397
Frontend [10]	62.1	457	1886
FCN-8s [45]	62.1	246	1573
FCN-16s [45]	62.3	246	1557
FCN-32s [45]	61.8	251	1549
FCN-32s [9]	60.0	251	1549
Proposed	32.6	8	1080

2) Experiments: We trained a decision tree using the proposed framework. To increase the variety of training data, the training images are augmented to total 5,565 images by randomly scaling, cropping, and altering color information. For scaling, a random number is sampled from a uniform distribution in [0.7, 1.3). If the scale is greater than one, the enlarged image is randomly cropped to an image with the original size. Concerning color alteration, images in RGB space are first transformed to images in HSV space. Then, values in H, S, and V space are scaled with a random number drawn from uniform distributions in [0.9, 1.1), [0.666, 1.5), and [0.666, 1.5), respectively. For training, we sampled 1,024 sets of 100,170 samples. The condition for becoming a leaf node is 0.9 for class probability, 0.000001 for the remaining data portion, and 25 for maximum depth.

3) Results: The quantitative results of an ablation study are shown in Table X. Although the pixel accuracy of the proposed method is lower than deep neural network-based methods, the processing time of the proposed method is about 30 times faster than the other methods [9], [10], [13], [45].

V. CONCLUSION

We present the random forest framework that employs the novel unconstrained representation to achieve real-time semantic segmentation. The unconstrained representation is proposed to learn optimal features to achieve higher accuracy. The random forest framework is selected to obtain high

efficiency for real-time processing. The results verify that the proposed method achieves higher accuracy comparing to previous random forest frameworks and processes using much smaller computational and memory resources comparing to deep learning-based methods.

REFERENCES

[1] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/4587503

[2] J. Shotton *et al.*, "Real-time human pose recognition in parts from single depth images," in *Proc. CVPR* , Jun. 2011, pp. 1297–1304.

[3] J. Shotton *et al.*, "Efficient human pose estimation from single depth images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2821–2840, Dec. 2013.

[4] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, "Robust part-based hand gesture recognition using kinect sensor," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1110–1120, Aug. 2013.

[5] T. Sharp *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proc. 333rd Annu. ACM Conf. Hum. Factors Comput. Syst.*, 2015, pp. 3633–3642.

[6] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *Trans. Graph.*, vol. 33, no. 5, p. 1691, Sep. 2014.

[7] B. Kang, K.-H. Tan, N. Jiang, H.-S. Tai, D. Tretter, and T. Nguyen, "Hand segmentation for hand-object interaction from depth map," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2017, pp. 259–263.

[8] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, Feb. 2012.

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[10] F. Yu and V. Koltun. (2016). "Multi-scale context aggregation by dilated convolutions." [Online]. Available: https://arxiv.org/abs/1511.07122

[11] S. Zheng *et al.*, "Conditional random fields as recurrent neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1529–1537.

[12] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *Proc. ICLR*, 2015.

[13] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[14] B. Kang, Y. Lee, and T. Q. Nguyen, "Depth-adaptive deep neural network for semantic segmentation," *IEEE Trans. Multimedia*, vol. 20, no. 9, pp. 2478–2490, Sep. 2018.

[15] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[17] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[18] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 44–57.

[19] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, 2008.

[20] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, Berlin Germany: Springer Oct. 2012, pp. 746–760.

[21] D. Tzionas and J. Gall, "3D object reconstruction from hand-object interactions," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 729–737.

[22] M. Cai, K. M. Kitani, and Y. Sato, "A scalable approach for understanding the visual structures of hand grasps," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 1360–1366.

[23] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Trans. Graph.*, vol. 28, no. 3, p. 63, 2009.

[24] Y. Wang *et al.*, "Video-based hand manipulation capture through composite motion control," *ACM Trans. Graph.*, vol. 32, no. 4, p. 43, Jul. 2013.

[25] J. Romero, H. Kjellström, C. H. Ek, and D. Kragic, "Non-parametric hand pose estimation with object context," *Image Vis. Comput.*, vol. 31, no. 8, pp. 555–564, Aug. 2013.

[26] J. Romero, H. Kjellström, and D. Kragic, "Hands in action: Real-time 3D reconstruction of hands in interaction with objects," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2010, pp. 458–463.

[27] 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.*, Jun. 2014, pp. 1106–1113.

[28] 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 *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2088–2095.

[29] B. Kang, Y. Lee, and T. Q. Nguyen, "Efficient hand articulations tracking using adaptive hand model and depth map," in *Proc. Int. Symp. Vis. Comput.*, Dec. 2015, pp. 586–598.

[30] B. Kang, S. Tripathi, and T. Q. Nguyen, "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map," in *Proc. 3rd IAPR Asian Conf. Pattern Recognit. (ACPR)*, Nov. 2015, pp. 136–140.

[31] 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. (CVPR)*, Jun. 2016, pp. 3593–3601.

[32] C. Li and K. M. Kitani, "Pixel-level hand detection in ego-centric videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3570–3577.

[33] A. Sinha, C. Choi, and K. Ramani, "DeepHand: Robust hand pose estimation by completing a matrix imputed with deep features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4150–4158.

[34] 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.

[35] X. Suau, J. Ruiz-Hidalgo, and J. R. Casas, "Real-time head and hand tracking based on 2.5D data," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 575–585, Jun. 2012.

[36] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, Aug. 1995, pp. 278–282.

[37] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324

[38] A. Criminisi and J. Shotton, *Decision Forests for Computer Vision and Medical Image Analysis*. London, England, Springer, 2013.

[39] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," MERL—Mitsubishi Electr. Res. Lab., Cambridge, MA, USA, Tech. Rep. TR94-03, Dec. 1994.

[40] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 2, Sep. 1999, pp. 1150–1157.

[41] F. Schroff, A. Criminisi, and A. Zisserman, "Object class segmentation using random forests," in *Proc. Brit. Mach. Vis. Conf.*, 2008, p. 54. doi: 10.5244/C.22.54.

[42] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1465–1479, Sep. 2006.

[43] J. Shotton, T. Sharp, P. Kohli, S. Nowozin, J. Winn, and A. Criminisi, "Decision jungles: Compact and rich models for classification," in *Advances in Neural Information Processing Systems*. New York, NY, USA: C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., pp. 234–242, 2013. [Online]. Available: https://papers.nips.cc/paper/5199-decision-jungles-compact-and-rich-models-for-classification

[44] G. Yu, N. A. Goussies, J. Yuan, and Z. Liu, "Fast action detection via discriminative random forest voting and top-k subvolume search," *IEEE Trans. Multimedia*, vol. 13, no. 3, pp. 507–517, Jun. 2011.

[45] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[46] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7794–7803.

[47] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[48] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. (2016). "Enet: A deep neural network architecture for real-time semantic segmentation." [Online]. Available: https://arxiv.org/abs/1606.02147
[49] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 1. New York, NY, USA: Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257
[51] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.
[52] Y. Jeon and J. Kim, "Active convolution: Learning the shape of convolution for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1846–1854.
[53] J. Dai *et al.*, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 764–773.
[54] Y. Freund and R. Schapire, "A short introduction to boosting," *J. Jpn. Soc. Artif. Intell.*, vol. 14, no. 5, pp. 771–780, 1999.
[55] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in Neural Information Processing Systems*. New York, NY, USA: J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 109–117. [Online]. Available:https://papers.nips.cc/paper/4296-efficient-inference-in-fully-connected-crfs-with-gaussian-edge-potentials
[56] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis.*, Jan. 1998, pp. 839–846.
[57] Y. Yuan and J. Wang. (2018). "Ocnet: Object context network for scene parsing." [Online]. Available: https://arxiv.org/abs/1809.00916
[58] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5168–5177.
[59] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 418–434. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-01219-9_25
[60] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, (2018). "CCNet: Criss-cross attention for semantic segmentation." [Online]. Available: http://arxiv.org/abs/1811.11721
[61] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6230–6239.
[62] S. Gupta, P. Arbeláez, and J. Malik, "Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 564–571.

Byeongkeun Kang received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2013, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California at San Diego, La Jolla, CA, USA, in 2015 and 2018, respectively. His current research interests include semantic segmentation, object detection, and human–machine interaction.

Truong Q. Nguyen (F'05) is currently a Professor and the Chair of the ECE Department, University of California at San Diego. He has coauthored (with Prof. G. Strang) a popular textbook *Wavelets and Filter Banks* (Wellesley-Cambridge Press, 1997) and has authored several MATLAB-based toolboxes on image compression, electrocardiogram compression, and filter bank design. He has over 400 publications. His current research interests are 3D video processing and communications and their efficient implementation.

Prof. Nguyen received the IEEE TRANSACTIONS IN SIGNAL PROCESSING Paper Award (image and multidimensional processing area) for the paper he coauthored with Prof. P. P. Vaidyanathan on linear-phase perfect-reconstruction filter banks (1992). He received the NSF Career Award in 1995. He is currently the Series Editor (Digital Signal Processing) for *Academic Press*. He served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 1994 to 1996, for the SIGNAL PROCESSING LETTERS from 2001 to 2003, for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1996 to 1997 and from 2001 to 2004, and for the IEEE TRANSACTIONS ON IMAGE PROCESSING from 2004 to 2005.