IEICE TRANSACTIONS

on Information and Systems

VOL. E101-D NO. 5
MAY 2018

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

IEICE TRANS. INF. & SYST., VOL.E101–D, NO.5 MAY 2018

PAPER Special Section on Machine Vision and its Applications

Simultaneous Object Segmentation and Recognition by Merging CNN Outputs from Uniformly Distributed Multiple Viewpoints

Yoshikatsu NAKAJIMA[†], *Nonmember and* **Hideo SAITO**[†a)], *Fellow*

SUMMARY We propose a novel object recognition system that is able to (i) work in real-time while reconstructing segmented 3D maps and simultaneously recognize objects in a scene, (ii) manage various kinds of objects, including those with smooth surfaces and those with a large number of categories, utilizing a CNN for feature extraction, and (iii) maintain high accuracy no matter how the camera moves by distributing the viewpoints for each object uniformly and aggregating recognition results from each distributed viewpoint as the same weight. Through experiments, the advantages of our system with respect to current state-of-the-art object recognition approaches are demonstrated on the UW RGB-D Dataset and Scenes and on our own scenes prepared to verify the effectiveness of the Viewpoint-Class-based approach.

key words: *object recognition, convolutional neural network, SLAM, segmentation*

1. Introduction

Object recognition is a vital technology in computer vision and robotic perception. It can be applied in various fields, including robotic manipulation, autonomous driving, and augmented reality.

We propose a novel multi-view-based recognition method that has the following advantages with respect to existing methods:

- working in real-time while processing SLAM, segmentation, and object recognition,
- managing smooth-surfaced objects and a large number of categories, and
- maintaining high accuracy regardless of the motion of the camera.

Our method employs the state-of-the-art dense map reconstruction and segmentation techniques proposed by Tateno *et al.* [1] to find a candidate object for recognition (i.e., object proposal). It segments each input depth image, then merges the obtained segments into a Global Segment Map (GSM, see Fig. 3, bottom right) reconstructed using the SLAM framework. One of the main advantages of this method is that the computational cost is stable and in real-time regardless of the size of the GSM and the number of merged depth maps.

In our method, the techniques are modified to uniformly distribute multiple viewpoints for each object (see Fig. 3, upper right) while maintaining the computational complexity of $O(n^2)$ (i.e., the size of the input image). We call each distributed viewpoint a Viewpoint Class (i.e., each small sphere distributed around each segmented object in Fig. 4). Only when the object is observed from a new Viewpoint Class from which the object has not yet been recognized, we crop the region of the object from a current frame to input the cropped images into a trained Convolutional Neural Network (CNN) for feature extraction. The aim of this procedure is to improve the final recognition accuracy by avoiding repeating the recognition computation when the camera stops at a poor view direction. As a secondary effect, the processing time is reduced by limiting the number of times the region is input to a CNN. Therefore, there is no trade-off between accuracy and real-time in this method. Furthermore, by utilizing a CNN as a tool for feature extraction, high scalability is achieved. In our method, any CNN structure that takes one input image and outputs its category can be used, whereby the range of application of this method is widened to consider various kinds of datasets for a CNN and trained CNN models are provided recently [2], [3].

There are several applicable methods regarding the issue of avoiding repeating the recognition computation when the camera stops at a poor view direction. For example, by recognizing objects only in PTAM-like keyframes, recognition results from the same view direction would not be accumulated. However, with the PTAM-like keyframes method, we cannot detect the change of the viewpoint for "each object", but can only detect whether the camera itself has moved or not. For example, between two keyframes, objects close to the camera are observed from a sufficiently different viewing direction, but objects far from the camera are observed from almost the same viewpoint. Our method is significant in that it detects changes in view directions that greatly change recognition results by tracking each object region in the scene accurately using [1], modifying [1] to distribute viewpoint classes around each object region, and recognizing each object from uniformly distributed viewpoints.

We demonstrate the achievement of the three above-mentioned advantages with experiments that also validate the improved object recognition performance. First, the system is compared against the current state-of-the-art [4], [5] using the UW-RGBD Dataset [5], [6]. The run-time performance of our system is analyzed to verify that it works in real-time. Furthermore, its scalability is verified by increasing the number of categories to 295. Finally, the effective-

Manuscript received September 7, 2017.
Manuscript revised November 27, 2017.
Manuscript publicized February 16, 2018.

[†]The authors are with Keio University, Yokohama-shi, 223–8522 Japan.
a) E-mail: hs@keio.jp

DOI: 10.1587/transinf.2017MVP0024

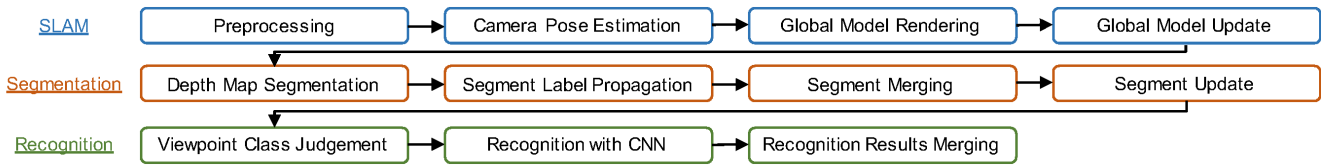


Fig. 1 Flow of the proposed method (upper: SLAM Phase, middle: Segmentation Phase, bottom: Recognition Phase).

ness of the Viewpoint-Class-based approach is verified under a scenario such that the camera idles in a poor position where recognition accuracy with a CNN is low due to scenes in which a part of the recognition target is occluded or has characteristics such as reflections and object shapes.

2. Related Works

Single-view-based Object Recognition Traditional state-of-the-art techniques for object recognition are based on HOG [7] and deformable-part-based models (DPM), as proposed by Felzenszwalb *et al.* [8] These methods exploit HOG features from the shape of each object and its parts across several scales to reduce its dimensions. However, the scalability is limited because the entire image is scanned in a sliding-window fashion for each object type that needs to be identified. While Dean *et al.* [9] proposed a method to improve such techniques for handling more object categories, there is still a trade-off between recognition performance and processing speed. LeCun *et al.* [10] showed that object recognition with a CNN is robust against Support Vector Machines (SVM) and k-Nearest Neighbor (k-NN) in terms of lighting and pose change since then, object recognition with a CNN has been actively researched. R-CNN, proposed by Girshick *et al.* [11], employs Selective Search [12] as a means of object proposal instead of sliding-window-based detection to reduce the computational cost. Furthermore, a CNN [3], [13]–[15] is employed for feature extraction. These two techniques are category-independent, so that high scalability is achieved.

Multi-view-based Object Recognition The above-mentioned single-view-based recognition methods, however, have a fundamental problem in that recognition performance depends on the appearance of target objects in the frame. Intuitively, by aggregating object evidence across multiple viewpoints, the recognition accuracy can be made more precise. Lai *et al.* [4] proposed a multi-view-based recognition method that aims to detect and label objects in 3D scenes by applying HOG-based detectors to assigning class probabilities to pixels of each RGB-D frame. These probabilities are incremented in voxels, and a labeled 3D map is built. While the performance is improved compared to single-view-based methods, 4 seconds are required to process each frame because a HOG-based approach is employed, which uses a large amount of computational time for feature extraction and sliding-window classification. Bao *et al.* [16], [17] proposed other multi-view-based recognition methods that jointly estimate camera pose, 3D points, and object regions by expanding the Structure-from-Motion (SfM) framework. Although the recognition performance and its robustness are improved, its computational cost is so huge that it cannot carry out real-time applications.

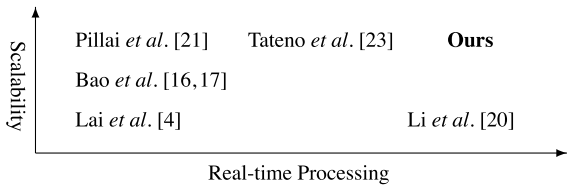


Fig. 2 Comparison of the proposed method and conventional methods in multi-view-based object recognition.

SLAM Framework for Multi-view-based Object Recognition Other works [5], [18]–[21] expand the SLAM framework for multi-view-based object recognition. Li *et al.* [20] proposed a multi-view-based object recognition method that works in real-time by extracting keypoints near 3D corners. While it achieves high efficiency with respect to the computational cost, it cannot manage smooth-surfaced objects whose keypoints are hard to extract (e.g., spherical objects). Pillai *et al.* [21] developed an ORB-SLAM-based object recognition method. Since it exploits features with spatial pyramid pooling using FLAIR [22], it can manage smooth-surfaced objects. However, not only does it not work in real-time, but it also has a problem in that class probabilities are computed across all frames in which the object is observed, so that the recognition accuracy is decreased if the camera remains in a poor position. [23] utilized reconstructed 3D shapes for object recognition by expanding the RGB-D SLAM framework for the sake of managing spherical objects and estimating the 3D poses of each object in a scene. Although such methods using depth information for recognition are highly accurate, their range of application is limited since full 3D models of each recognition target are required for machine learning. Figure 2 summarizes the conventional methods mentioned above and the proposed method along two axes: real-time processing and scalability.

Object Detection and Tracking Many methods [24]–[27] for robust object tracking by collecting learning samples of tracking targets online and updating the models have been proposed. However, unlike the method of Tateno *et al.* [1], these methods do not reconstruct dense 3D model in which each object is segmented (i.e., GSM), furthermore, camera pose against each object is not estimated. These are critical issues for achieving high accuracy regardless of the

motion of the camera, which is realized by uniformly distributing viewpoint classes around each segmented object and recognizing objects from these viewpoint classes with our method.

3. Proposed Method

In this section, we describe our proposed method, which simultaneously processes reconstruction, segmentation, and object recognition. Figure 1 shows a flow diagram of the proposed method. Our method consists of three phases (SLAM, Segmentation, and Recognition). Firstly, we provide an overview of the SLAM and Segmentation Phases in order to reach parameters used in the Recognition Phase. After that, we describe the Recognition Phase in detail, which is the main contribution of this work. The inputs are just RGB and depth images obtained from a moving RGB-D sensor, which we process individually.

3.1 SLAM Phase

This section provides an overview of the SLAM Phase (see Fig. 1, upper stage). We employed the SLAM system proposed by Keller *et al.* [28] because a global model, which is a model reconstructed through the SLAM framework, consists only of point clouds. Thus, it can manage a wider environment compared to voxel-based methods, including Kinect Fusion [29]. Each point s_k of a global map $\mathcal{S}$ has information including a 3D position $\boldsymbol{v}_k \in \mathbb{R}^3$, a normal $\boldsymbol{n}_k \in \mathbb{R}^3$, a confidence $c_k \in \mathbb{R}$, and a time stamp $t_k \in \mathbb{N}$.

The Preprocessing Stage is for smoothing a depth image $\mathcal{D}_t$ at current frame t with a bilateral filter [30] and transforming $\mathcal{D}_t$ into a vertex map $\mathcal{V}_t(\boldsymbol{u}) = \boldsymbol{K}^{-1}\dot{\boldsymbol{u}}\mathcal{D}_t(\boldsymbol{u})$ using the camera intrinsic parameter $\boldsymbol{K}$, a depth map element $\boldsymbol{u} = (x, y)^{\mathrm{T}}$ in the image domain $\boldsymbol{u} \in \mathbb{R}^2$, and its homogeneous coordinate $\dot{\boldsymbol{u}}$. The normal map $\mathcal{N}_t$ is also generated in this stage by using a cross-product calculation to $\mathcal{V}_t$.

The Camera Pose Estimation Stage is for calculating the current camera pose $\boldsymbol{T}_t = [\boldsymbol{R}_t, \boldsymbol{t}_t] \in \mathbb{SE}(3)$, $\boldsymbol{R}_t \in \mathbb{SO}(3)$, and $\boldsymbol{t}_t \in \mathbb{R}^3$ by using $\mathcal{V}_t$, $\mathcal{V}^m_{t-1}$, and $\mathcal{N}^m_{t-1}$. At this time, we denote the rendered map of the global model with respect to a particular camera pose as m. The point-to-plane ICP algorithm proposed by Low [31] takes these three maps and outputs a rotation and translation between the current frame and the previous frame.

The Global Model Rendering Stage is for obtaining the correspondences between the point clouds generated by the current depth map $\mathcal{D}_t$ and the global model $\mathcal{S}$. The index map is generated in this stage by projecting point clouds from the global map via the projection matrix $\boldsymbol{P}_t$, which consists of the current camera pose $\boldsymbol{T}_t$ and the intrinsic parameters $\boldsymbol{K}$. $\mathcal{V}^m_t$ and $\mathcal{N}^m_t$ are also generated at this stage for the Camera Pose Estimation stage in the next frame.

The Global Model Update Stage is for merging or adding the point clouds generated from the current depth map $\mathcal{D}_t$ to the global model. Only when specific geometric conditions are satisfied is the point $\mathcal{D}_t(\boldsymbol{u})$ merged to a point s_k already present on the global map $\mathcal{S}$, and the associated confidence c_k is incremented.

3.2 Segmentation Phase

This section provides an overview of the Segmentation Phase (see Fig. 1, middle stage) which determines the object targeted for object recognition. The segmentation framework we employed is based on the method by Tateno *et al.* [1]. It takes the current depth map $\mathcal{D}_t$ and incrementally builds up and updates a Global Segmented Map (GSM) $\mathcal{L}$ for each frame. The components of the GSM are the same as those for the global map $\mathcal{S}$, and each point on the GSM is labeled. The main advantage of this system, and our reason for employing it, is that the computational cost for updating a GSM never increases, as opposed to other segmentation systems [32].

The Depth Map Segmentation Stage is for segmenting the inputed depth map $\mathcal{D}_t$ by conducting normal edge analysis. The process takes its vertex map $\mathcal{V}_t$ and normal map $\mathcal{N}_t$ as inputs and a binary edge map $\mathcal{B}_t$ is outputted by comparing the nearby normal angles and vertex distances. Then, a connected component algorithm is applied to the binary map to obtain a label map $\mathcal{L}_t$ on which each element $\mathcal{L}_t(\boldsymbol{u})$ is associated to the label l_j, $l_j \subset \mathbb{Z}_{\geq 0}$.

The Segment Label Propagation Stage is for generating a propagated label map $\mathcal{L}^p_t$, where each element $\mathcal{L}^p_t(\boldsymbol{u})$ is associated with a label assigned to each point constituting a GSM. To achieve this goal, firstly, the rendered label map $\mathcal{L}^m_t$ is computed by projecting the GSM with $\boldsymbol{P}_t$, which was created in the Camera Pose Estimation Stage. Next, the overlap percentage between the label $l_i \in \mathcal{L}^m_t$, $l_i \subset \mathbb{Z}_{\geq 0}$ and $l_j \in \mathcal{L}_t$ is computed and used to decide whether l_i is propagated to $\mathcal{L}^p_t$ or l_j is to be used directly. Finally, a propagated label map $\mathcal{L}^p_t$ of a current frame t (see Fig. 3, left bottom) is obtained.

Fig. 3 Actual image of Viewpoint Class uniformly distributed around each segmented object in the Global Segment Map (GSM). Green pyramids represent the camera trajectory up to the current frame t. The Viewpoint Class colored in red is the one from which the object has already been recognized. Left side, top to bottom: input RGB image, normal map $\mathcal{N}_t$, propagated label map $\mathcal{L}^p_t$.

The Segment Merging Stage is for merging segments that originally consisted of the same object. When the overlapped percentage of $l_a, l_b \in \mathcal{L}_t^m$, calculated in the Segment Label Propagation Stage, is sufficiently larger than the threshold, the segment pair (l_a, l_b) is merged and replaced with l_a.

The Segment Update Stage is for updating the GSM with $\mathcal{L}_t^p$. When reconstructing the GSM in real time, it is not robust to directly modify the GSM based on the label indications contained in $\mathcal{L}_t^p$, since such a label map which is created from one single depth frame usually contains noisy information. Therefore, [1] introduced a confidence-based approach by assigning each element of the GSM map to a confidence. The label of each point in the GSM is updated only when its label confidence exceeds the threshold, otherwise, only its label confidence is changed.

3.3 Recognition Phase

In this section, we describe the Recognition Phase, which is the core of our proposal. As shown in Fig. 4, the main contribution of this work is uniformly distributing the viewpoints around each object in the GSM and impartially merging the recognition results from each distributed viewpoint with the same weight. We call each distributed viewpoint *Viewpoint Class*. To achieve this goal, in contrast to [1], each segmented object O_j has information about its centroid $\boldsymbol{C}_j \in \mathbb{R}^3$ for centering the Viewpoint Class on the centroid $\boldsymbol{C}_j$. These centroids are updated in the Segment Update Stage. Furthermore, O_j possesses recognition results from each Viewpoint Class.

The Recognition Phase, depicted in red in Fig. 1, is processed in every frame as in the SLAM and Segmentation Phase. However, Viewpoint Class generation is performed only once before the initial frame as a pre-processing step. The Viewpoint Class shown in Fig. 4 is distributed over the circle. The methodology of distributing viewpoints is based on work by Saff *et al.* [33]. N points can be distributed uniformly over the surface of a sphere whose radius r is 1 with following equations.

$$\begin{aligned} &\theta_\gamma = \arccos(h_\gamma), h_\gamma = -1 + \frac{2(\gamma-1)}{(N-1)}, 1 \le \gamma \le N, \\ &\phi_\gamma = \left(\phi_{\gamma-1} + \frac{3.6}{\sqrt{N}} \frac{1}{\sqrt{1-{h_\gamma}^2}} \right) (\mathrm{mod}\ 2\pi), \\ &2 \le \gamma \le N-1,\ \phi_1 = \phi_N = 0, \\ &0 \le \theta_\gamma \le \pi,\ 0 \le \phi_\gamma \le 2\pi \end{aligned} \tag{1}$$

θ_γ and ϕ_γ are defined in the polar coordinate system. We store each coordinate $\boldsymbol{\psi}_\gamma \in \mathbb{R}^3$ that is generated by converting θ_γ and ϕ_γ into xyz coordinates.

As shown in Fig. 1, the Recognition Phase also consists of 3 stages. In the first stage (i.e., Sect. 3.3.1), we judge whether each object region included in the current frame is observed from a new Viewpoint Class. In the second stage (i.e., Sect. 3.3.2), for each object that is judged to be observed from a new Viewpoint Class, the object region in the current RGB frame is fed into the CNN. In the last stage (i.e., Sect. 3.3.3), the output of the CNN is accumulated to the recognition result of the object region of GSM. Therefore, our method doesn't require the recognition of the object type of each segment before Sect. 3.3.1, since each object in the current frame is recognized in the Recognition with CNN stage and the recognition result is incremented to the GSM in Sect. 3.3.3. Following are the details for each stage.

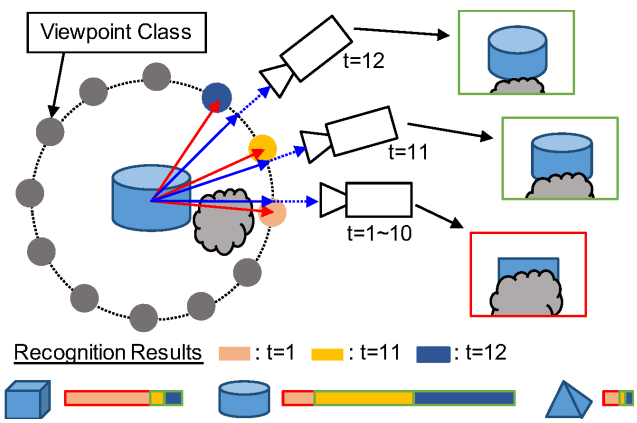


Fig. 4 The concept of our Viewpoint-Class-based recognition system. Each circle uniformly distributed around the object indicates a Viewpoint Class. Since the recognition results from each Viewpoint Class (i.e., CNN outputs at $t = 1, 11, 12$) are aggregated as the same weight, even if the camera idles in a bad position ($t = 1 \sim 10$), the accuracy of the recognition result increases in the end.

3.3.1 Viewpoint Class Judgment

The objective of this stage is to determine if the current camera pose belongs to the new Viewpoint Class for each object O_j. We perform the following processing for each object appearing in the propagated label map $\mathcal{L}_t^p$. Notably, because processing targets are limited in objects appearing in $\mathcal{L}_t^p$, the computational cost is maintained in $O(n^2)$ (i.e., the size of the input image) even if the GSM becomes large.

First, we compute the vector $\boldsymbol{V}_j^{ct}$ starting at the centroid $\boldsymbol{C}_j$ and ending at the current camera position $\boldsymbol{t}_t$ in world coordinates with $\boldsymbol{V}_j^{ct} = \boldsymbol{t}_t - \boldsymbol{C}_j$ for each object O_j. At this time, the current camera position $\boldsymbol{t}_t$ is already computed in the Camera Pose Estimation Stage. Next, the vector $\boldsymbol{V}_j^{ct}$ is normalized to length r (i.e., a radius of the sphere in which the Viewpoint Classes are distributed). Figure 4 shows the vector $\boldsymbol{V}_j^{ct}$ in blue. Considering that each prepared Viewpoint Class $\boldsymbol{\psi}_\gamma$ is distributed on a sphere whose center is the origin of the coordinate, we can determine the Viewpoint Class to which the current camera pose belongs by comparing vectors $\boldsymbol{\psi}_\gamma$ and $\boldsymbol{V}_j^{ct}$. Thus, the γ that minimizes the distance between $\boldsymbol{\psi}_\gamma$ and $\boldsymbol{V}_j^{ct}$ is the Viewpoint Class to which the current camera pose belongs.

$$\bar{\gamma}_j = \underset{1 \le \gamma \le N}{\operatorname{argmin}} \|\boldsymbol{\psi}_\gamma - \boldsymbol{V}_j^{ct}\| \tag{2}$$

We denote the Viewpoint Class as $\bar{\gamma}_j$. We denote the recognition result of an object O_j from Viewpoint Class γ as Ω_{γ_j}. If the recognition result $\Omega_{\bar{\gamma}_j}$ is empty, the object O_j is recognized in the next stage and its index is denoted as $\hat{j}$.

The upper right image in Fig. 3 shows the Viewpoint Classes with the centroids of each object O_j as the center by simply adding each vector $\boldsymbol{\psi}_\gamma$ to each centroid $\boldsymbol{C}_j$. The Viewpoint Classes from which objects have already been recognized are colored red and the others are gray.

3.3.2 Recognition with CNN

After the objects recognized in this stage are determined, segments of each object $O_{\hat{j}}$ in the RGB image of the current frame are cropped based on the propagated label map $\mathcal{L}_t^p$. Each cropped image $\mathcal{I}_{\hat{j}}$ shows the appearance of the object $O_{\hat{j}}$ from the new Viewpoint Class $\psi_{\bar{\gamma}_{\hat{j}}}$. Then, if the image $\mathcal{I}_{\hat{j}}$ is much smaller than the input size of the CNN or the percentage between the total number of labeled pixels of the object $O_{\hat{j}}$, which can be calculated by the propagated label map $\mathcal{L}_t^p$, and the size of image $\mathcal{I}_{\hat{j}}$ is lower than the threshold, it is discarded.

Next, these images are input into the CNN, which has been tuned by deep learning with a specific dataset (e.g., ImageNet [2], [3]). In our method, any CNN structure that takes one input RGB image and outputs class probabilities can be used as described in Sect. 1. Therefore, we can use any object database as long as it has RGB images with a correct object label. Since a variety of datasets for CNN and trained CNN models are recently provided, the range of application of our method can be extended.

At this time, the softmax function is not applied to the output of the CNN because merging the outputs of the CNN from each Viewpoint Class and calculating the class probability are performed in the next stage. CNN models for object recognition usually apply the softmax function to the output of the CNN to calculate probabilities. In our method, the output of the CNN is stored without applying the softmax function, and the outputs of the CNN from multiple viewpoints are accumulated for each object, and then the accumulated result up to the current frame is applied to the softmax function to calculate class probabilities of the object. We call the CNN output without the softmax function "the raw output." The raw output of the CNN is stored as $\Omega_{\bar{\gamma}_{\hat{j}}}$, which signifies the recognition result of the object $O_{\hat{j}}$ from the Viewpoint Class $\psi_{\bar{\gamma}_{\hat{j}}}$.

3.3.3 Merging the Recognition Results

To recognize each object, the recognition results are merged and renewed for each object $O_{\hat{j}}$ with the following equation, where $\psi_{\hat{j}}^r$ represents a subset of Viewpoint Classes from which the object O_j has already been recognized.

$$y_{\hat{j}}^{\lambda} = \frac{\exp\left(\sum_{\gamma_j \in \psi_{\hat{j}}^r} \Omega_{\gamma_j}(\lambda)\right)}{\sum_{i=1}^{i=\Lambda} \exp\left(\sum_{\gamma_j \in \psi_{\hat{j}}^r} \Omega_{\gamma_j}(i)\right)} \tag{3}$$

At this time, λ shows the object category to be recognized. Therefore, when the total number of categories to be recognized is Λ, the domain of λ is $1 \le \lambda \le \Lambda$, $\lambda \subset \mathbb{N}$. The probability $y_{\hat{j}}^{\lambda}$ that the object $O_{\hat{j}}$ categorized to λ is calculated with $\psi_{\hat{j}}^r$, the total number of categories Λ, and $\Omega_{\gamma_j}(i)$, which denotes the CNN output of category i from a Viewpoint Class γ_j of an object O_j. The concept of this equation is applying a softmax function after adding the CNN outputs. In Fig. 4, the "Recognition Results" refer to the added CNN outputs and the colored circles represent the Viewpoint Classes from which the object has already been recognized. In this case, the probabilities are simply calculated by applying a softmax function to the added outputs.

4. Experiments

In this section, we experimentally demonstrated the validity of our method. In our experiments, we evaluated our method on the popular UW RGB-D Dataset (v2) [5], [6] and our own dataset. In Sect. 4.1, we compared our method with the current state-of-the-art methods by Lai *et al.* [4], [5] and Pillai *et al.* [21] that utilize full map and camera positions, respectively, for improved recognition performance. The UW RGB-D Dataset contains a total 295 object categories, however, for a fair comparison, we considered the same 5 categories as noted in [4], [5], [21]. Subsequently, we demonstrated the performance of our method by increasing the number of objects to all 295 categories. In Sect. 4.2, the validity of the recognition based on Viewpoint Class was demonstrated using our own dataset and scenes.

Following are the details of the evaluation environment. CPU: Intel Core i7-4770K 3.50GHz, GPU: GeForce GTX 760 and RAM: 16GB. The deep learning framework used in this evaluation experiment was Chainer [34]. Throughout the experiment, the number of Viewpoint Classes was 700.

4.1 UW RGB-D Dataset

The CNN model selected for use in this experiment was Network In Network (NIN) [35] because it is useful for cutting the classification processing time by reducing the number of parameters while maintaining high accuracy. Since the UW RGB-D Dataset provides mask images, we masked the region for each object on each training image. Next, we trained the CNN model by randomly rescaling and adding noise for robust predictions. In the Recognition with CNN stage (Sect. 3.3.2), the regions for each object determined to be input to the CNN under the conditions of the Viewpoint Class Judgment stage (Sect. 3.3.1) were masked based on the propagated label map $\mathcal{L}_t^p$ and input to the CNN as in the training.

Table 1 Precision/Recall rate using the UW RGB-D scene dataset [5], [6].

Method	View(s)	Input	Precision/Recall						
			Bowl	Cap	Cereal Box	Coffee Mug	Soda Can	Background	Overall
DetOnly [4]	Single	RGB	46.9/90.7	54.1/90.5	76.1/90.7	42.7/74.1	51.6/87.4	98.8/93.9	61.7/87.9
Det3DMRF [4]	Multiple	RGB-D	91.5/85.1	90.5/91.4	93.6/94.9	90.0/75.1	81.5/87.4	99.0/99.1	91.0/88.8
HMP2D+3D [5]	Multiple	RGB-D	97.0/89.1	82.7/99.0	96.2/99.3	81.0/92.6	97.7/98.0	95.8/95.0	90.9/95.6
BoVW+FLAIR [21]	Multiple	RGB	88.7/70.2	99.4/72.0	95.6/84.3	80.1/64.1	89.1/75.6	96.6/96.8	89.8/72.0
Ours	Multiple	RGB	96.2/91.8	92.2/95.9	98.4/96.1	91.9/87.1	91.7/89.3	94.0/100.0	94.1/93.4

Table 2 Average time spent on each processing stage.

(Unit: ms)

	Proposed	**Frame-based**	DetOnly [4]	HMP2D+3D [5]	BoVW+FLAIR [21]
Viewpoint Class Judgment	1.0	-	-	-	-
Recognition with CNN	98.9	229.8	-	-	-
Recognition Result Merging	10.7	-	-	-	-
Total	110.6	229.8	$\approx$ 1800	$\approx$ 4000	$\approx$ 1600

We evaluated the recognition performance of our method on each scene in the UW RGB-D Dataset. We calculated precision, recall, and mean-Average Precision using the ground truth annotations provided in a bounding box. In this method, object recognition is performed in the pixel level. For the fair comparison of the proposed object recognition with the other methods, we compute the recognition accuracy for the pixel area of each recognized object surrounded by a bounding box by comparing it with the ground truth. Therefore, if segmentation fails, the bounding box will be drawn in a different part from the ground truth, which makes the score lower. Table 1 shows the mean-Average Precision (mAP) estimates of our method and the existing methods reported in [4], [5], [21]. As shown in Table 1, we were able to achieve a performance of 94.1 mAP as compared to the detector performance of 61.7 and the SLAM-aware BoVW+FLAIR performance of 89.8.

Table 2 shows the processing time for each stage. We compared our proposed method with the frame-based recognition method. In this comparison target, the CNN model for recognition was the same as the one used in the proposed method and was trained with the same dataset. However, the recognition result was not aggregated as in the proposed method. In other words, the recognition result of the comparison method was simply computed by inputting cropped images based on the propagated label map $\mathcal{L}_t^p$ into the trained CNN model for each frame. The average processing time for the Viewpoint Class Judgment stage was relatively short because the comparison of the vector to the current camera position and each Viewpoint Class is performed by Nearest Neighbor search for three-dimensional vectors. Furthermore, the average processing time for the Recognition with CNN stage of the proposed method was short compared to the frame-based method because only the cropped image of the object whose recognition result from current Viewpoint Class is empty is recognized in the Recognition with CNN stage. Thus, the number of images inputted to the CNN was decreased and the processing time was shorter than the frame-based method. Considering that the SLAM and Segmentation Phase achieved 72 fps [1], our system can work in real-time. Table 2 also shows advantages of SLAM-and GSM-based object proposal and CNN-based feature extraction in terms of processing time, compared with conventional methods using sliding-window-based detection and HOG-and FLAIR-based classification.

Next, we describe recognition performance where the number of objects is increased to 295 to verify the scalability of our method. Figure 5 shows the 3D model of the scene reconstructed by the SLAM framework, the camera trajectory depicted in green line, and recognition results in several frames. The recognition results were shown by filling in the objects with red based on the propagated label map $\mathcal{L}_t^p$ and denoting the most probable category and its probability as calculated by Eq. (3). As shown in Fig. 5, even if the number of categories was increased to 295, sufficient recognition accuracy was achieved. Figure 6 shows the flow as the recognition result becomes increasingly accurate by aggregating object evidence across multiple viewpoints. As shown in Fig. 6, one of the limitations of our method is that the object in the scene is divided (e.g., inside and outside of coffee mug) since the segmentation is based on normal and vertex information.

4.2 Validity of Viewpoint-Class-Based Approach

This experiment demonstrates that our method successfully improves the recognition accuracy by detecting the Viewpoint Class according to our proposed method. We compared our method with the accumulation-based method, which accumulates CNN outputs for each frame to each object in the scene without considering the Viewpoint Class. In other words, the accumulation-based method that simply integrates all of the frame recognition results to calculate the final recognition result, as with conventional methods typified by [21]. We prepared the scene such that the camera idled in a bad position in the second half frame. The target objects in this experiment were books. We picked cover images of 33 books from the Web and generated 500 learning images with a homography transformation for each book. At that point, we masked the region for each book and added noises to the learning images with the method described in Sect. 4.1.

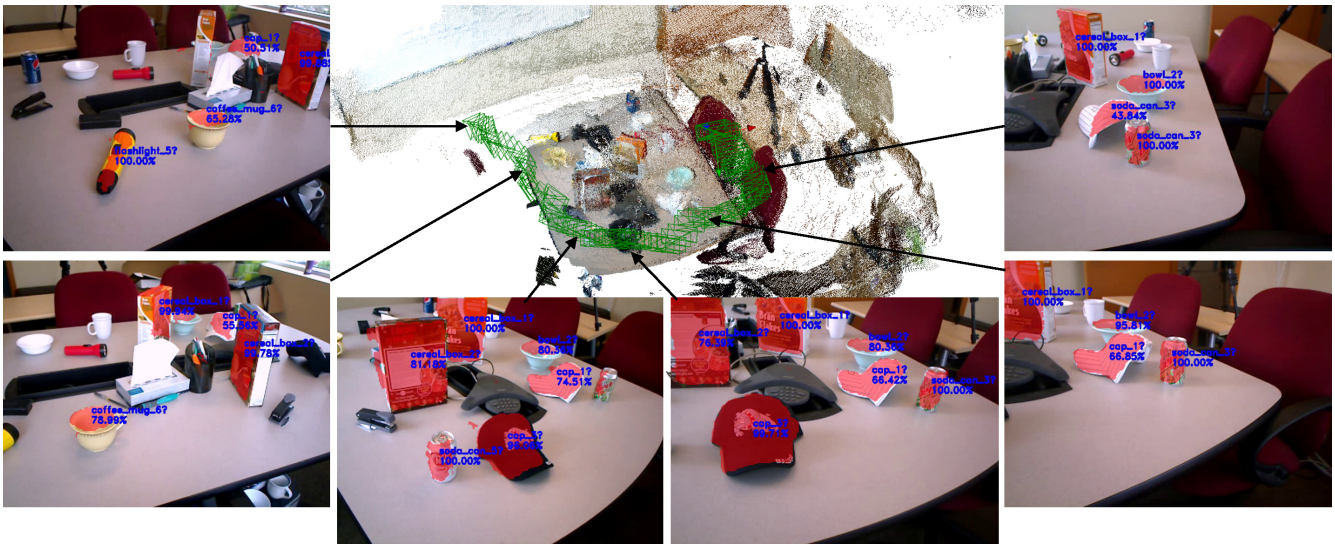
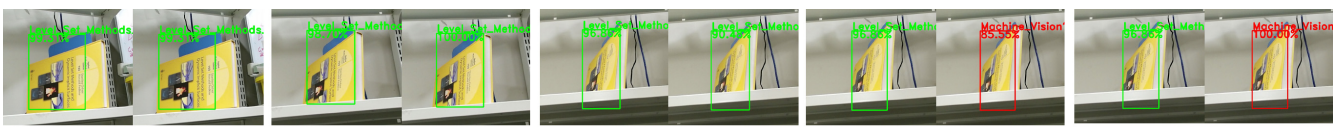


Fig. 5 Recognition results in several frames, 3D model of the scene reconstructed by SLAM framework, and camera trajectory depicted in green line.

(a) cap_1 (b) coffee_mug_6

Fig. 6 Flow as the recognition result becomes accurate through the multiple view approach.



(a) t=0 (b) t=13 (c) t=40 (d) t=52 (e) t=61

Fig. 7 Results of the experiment using our own dataset, which aimed to validate the efficiency of the Viewpoint-Class-based approach by comparing our method with the accumulation-based method (left: ours, right: accumulation-based).

Figure 7 shows the recognition results for each frame. The results of the Viewpoint-Class-based approach were more precise than the accumulation-based method, especially in the frame after camera stagnation. This is because the object was recognized from each Viewpoint Class with the same weight, while the accumulation-based method accumulated inaccurate recognition results from poor camera positions.

5. Conclusions

In this work, we developed a Viewpoint-Class-based object recognition system that achieves real-time processing, scalable performance, and robustness for camera movement. We leveraged a state-of-the-art SLAM-based segmentation method for object proposals and utilized a CNN for future extraction to handle even smooth-surfaced objects and achieve high scalability. Furthermore, by uniformly distributing Viewpoint Classes around each object and aggregating recognition results from each Viewpoint Class, robustness for camera movement was achieved. These contributions of our system were demonstrated through various experiments using the UW RGB-D Dataset and our own dataset. Moreover, the results were superior to conventional methods.

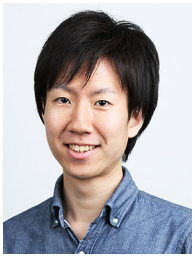
Acknowledgements

This research presentation is supported in part by a research assistantship of a Grant-in-Aid to the Program for Leading Graduate School for “Science for Development of Super Mature Society” from the Ministry of Education, Culture, Sport, Science, and Technology in Japan.

References

[1] K. Tateno, F. Tombari, and N. Navab, “Real-time and scal-

able incremental segmentation on dense SLAM," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp.4465–4472, 2015.
[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.248–255, 2009.
[3] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems (NIPS), pp.1097–1105, 2012.
[4] K. Lai, L. Bo, X. Ren, and D. Fox, "Detection-based object labeling in 3D scenes," IEEE International Conference on Robotics and Automation (ICRA), pp.1330–1337, 2012.
[5] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," IEEE International Conference on Robotics and Automation (ICRA), pp.3050–3057, 2014.
[6] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," IEEE International Conference on Robotics and Automation (ICRA), pp.1817–1824, 2011.
[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.886–893, 2005.
[8] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," IEEE Trans. Pattern Anal. Mach. Intell., vol.32, no.9, pp.1627–1645, 2010.
[9] T. Dean, M.A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, accurate detection of 100,000 object classes on a single machine," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1814–1821, 2013.
[10] Y. LeCun, F.J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.97–104, 2004.
[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.580–587, 2014.
[12] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," International Journal of Computer Vision (IJCV), vol.104, no.2, pp.154–171, 2013.
[13] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural Computation, vol.1, no.4, pp.541–551, 1989.
[14] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," Pattern Recognition, vol.15, no.6, pp.455–469, 1982.
[15] D.H. Hubel and T.N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," The Journal of Physiology, vol.160, no.1, pp.106–154, 1962.
[16] S.Y. Bao and S. Savarese, "Semantic structure from motion," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2025–2032, 2011.
[17] S.Y. Bao, M. Bagra, Y.-W. Chao, and S. Savarese, "Semantic structure from motion with points, regions, and objects," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2703–2710, 2012.
[18] L. Nan, K. Xie, and A. Sharf, "A search-classify approach for cluttered indoor scene understanding," ACM Trans. Graphic., vol.31, no.6, article no.137, 2012.
[19] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, "An interactive approach to semantic modeling of indoor scenes with an RGBD camera," ACM Trans. Graphic., vol.31, no.6, article no.136, 2012.
[20] Y. Li, A. Dai, L. Guibas, and M. Nießner, "Database-assisted object retrieval for real-time 3D reconstruction," Computer Graphics Forum, vol.34, no.2, pp.435–446, 2015.
[21] S. Pillai and J. Leonard, "Monocular SLAM supported object recognition," arXiv preprint arXiv:1506.01732, 2015.
[22] K.E.A. van de Sande, C.G.M. Snoek, and A.W.M. Smeulders, "Fisher and VLAD with FLAIR," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2369–2376, 2014.
[23] K. Tateno, F. Tombari, and N. Navab, "When 2.5D is not enough: Simultaneous reconstruction, segmentation and recognition on dense SLAM," IEEE International Conference on Robotics and Automation (ICRA), pp.2295–2302, 2016.
[24] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," IEEE Trans. Pattern Anal. Mach. Intell., vol.33, no.8, pp.1619–1632, 2011.
[25] D.A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," International Journal of Computer Vision (IJCV), vol.77, no.1, pp.125–141, 2008.
[26] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit, "Online learning of patch perspective rectification for efficient object detection," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1–8, 2008.
[27] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2411–2418, 2013.
[28] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," IEEE International Conference on 3D Vision, pp.1–8, 2013.
[29] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera," Proc. 24th Annual ACM Symposium on User Interface Software and Technology, pp.559–568, 2011.
[30] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," Sixth International Conference on Computer Vision (ICCV), pp.839–846, 1998.
[31] K.L. Low, "Linear least-squares optimization for point-to-plane ICP surface registration," Chapel Hill, University of North Carolina, vol.4, 2004.
[32] P.F. Felzenszwalb and D.P. Huttenlocher, "Efficient graph-based image segmentation," International Journal of Computer Vision (IJCV), vol.59, no.2, pp.167–181, 2004.
[33] E.B. Saff and A.B.J. Kuijlaars, "Distributing many points on a sphere," The Mathematical Intelligencer, vol.19, no.1, pp.5–11, 1997.
[34] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: A next-generation open source framework for deep learning," Proc. Workshop on Machine Learning Systems in the Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS), 2015.
[35] M. Lin, Q. Chen, and S. Yan, "Network in network," arXiv preprint arXiv:1312.4400, 2013.



Yoshikatsu Nakajima received his B.E. degrees in information and computer science from Keio University, Japan, in 2016. Since 2016, he has been in Master course of Science and Technology at Keio University, Japan. His research interests include augmented reality, SLAM, object recognition, and computer vision.

Hideo Saito received his Ph.D. degree in Electrical Engineering from Keio University, Japan, in 1992. Since then, he has been on the Faculty of Science and Technology, Keio University. In 1997 to 1999, he had joined into Virtualized Reality Project in the Robotics Institute, Carnegie Mellon University as a visiting researcher. Since 2006, he has been a full Professor of Department of Information and Computer Science, Keio University. His recent activities for academic conferences includes a Program Chair of ACCV2014, a General Chair of ISMAR2015, and a Program Chair of ISMAR2016. His research interests include computer vision and pattern recognition, and their applications to augmented reality, virtual reality, and human robotics interaction.