2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Estimation of object class and orientation from multiple viewpoints and relative camera orientation constraints

Koichi Ogawara[1] and Keita Iseki[2]

Abstract— **In this research, we propose a method of estimating object class and orientation given multiple input images assuming the relative camera orientations are known. Input images are transformed to descriptors on 2-D manifolds defined for each class of object through a CNN, and the object class and orientation that minimize the distance between input descriptors and the descriptors associated with the estimated object class and orientation are selected. The object orientation is further optimized by interpolating the viewpoints in the database.**

The usefulness of the proposed method is demonstrated by comparative evaluation with other methods using publicly available datasets. The usefulness of the proposed method is also demonstrated by recognizing images taken from the cameras on our humanoid robot using our own dataset.

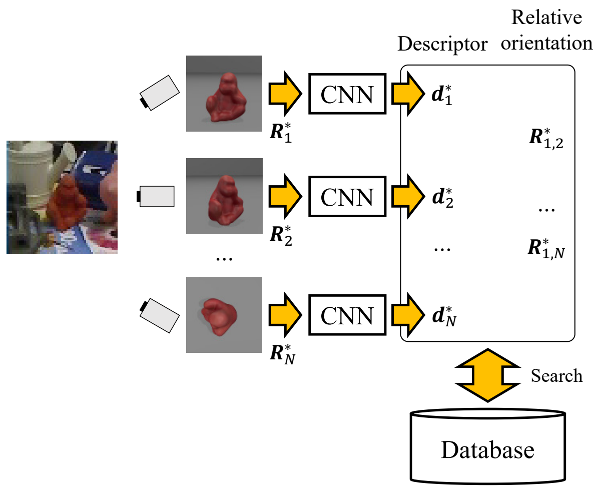

Fig. 1. Overview of object recognition.

I. INTRODUCTION

Recently, robots are expected to handle a variety of tasks including object manipulation in everyday environments. If a robot has limited knowledge of the environment, the robot needs to know the class and orientation of the object before manipulating it.

In this research, we assume that multiple cameras are available for object recognition and the relative orientation between cameras are known. This is typically the case where a robot has multiple cameras attached to the body such as the head and arms, and the relative orientation between the cameras can be calculated by forward kinematics.

So far, various methods of estimating object class and orientation from a single image or multiple images have been proposed such as those with image clustering [1], [2], i.e. support vector machine, random forest, etc., and those with approximate nearest neighbor search [3], [4]. However, they rely on manually designed image features such as HOG [5] and SURF [6] that are not optimal compared with the statistically learned image features.

Recently, end-to-end learning with deep convolutional neural networks (CNN), especially those using metric learning [7], [8], has received great attention in modeling the relationship between input images and the object class and orientation [9], [10], [11], [12], [13]. Wohlhart et al. proposed a method of estimating object class and orientation by training a CNN with a triplet loss function [14] to output descriptors where descriptors of different classes are mapped apart and descriptors of the same class and similar orientation are mapped nearby [13]. However, this method assumes a single input image and does not support multiple input images. Kanezaki et al. proposed a method of estimating object class and orientation by training a CNN to directly generate the likelihood of the object class and orientation given multiple input images [11]. However, this method does not take advantage of the known relative orientation between cameras. Furthermore, these methods select the estimated orientation from the orientations stored in the training data, however the true orientation is not necessarily included in the training data, and errors will occur.

In this research, we propose a method of estimating object class and orientation given multiple input images assuming the relative camera orientations are known as shown in Fig. 1.

The contributions of the proposed method are twofold: (1) the accuracy of object class and orientation estimation is improved by adding relative camera orientation constraints, (2) the accuracy of object orientation estimation is further improved by interpolating the training data.

The usefulness of the proposed method is demonstrated using several publicly available datasets and our own dataset.

II. ESTIMATION OF OBJECT CLASS AND ORIENTATION

Given N input images and their camera orientations, the proposed method estimates the class and orientation of the object in these images. At the training step, we train the CNN so that an input image is transformed to a descriptor on 2-D manifolds defined for each class of object. Then, we build a database whose elements consist of object classes, a camera orientation, and a descriptor. At the evaluation step,

[1]Koichi Ogawara is with Faculty of Systems Engineering, Wakayama University, 930, Sakaedani, Wakayama-shi, Wakayama, Japan ogawara@wakayama-u.ac.jp
[2]Keita Iseki is with Graduate School of Systems Engineering, Wakayama University, 930, Sakaedani, Wakayama-shi, Wakayama, Japan s192008@wakayama-u.ac.jp

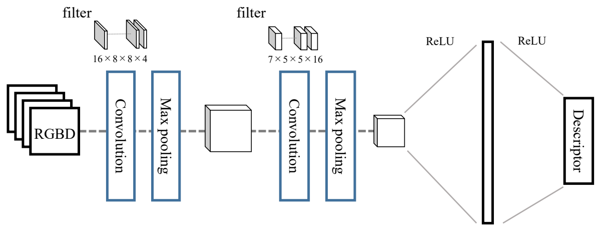

Fig. 2. Structure of CNN.

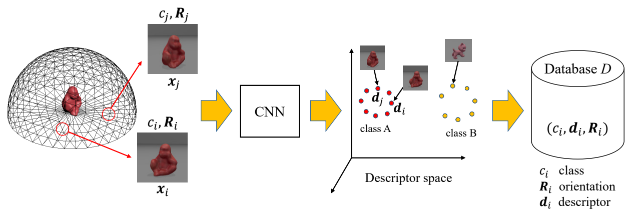

Fig. 3. Database of viewpoints.

the database is searched for the set of elements that are most similar to the input descriptors and satisfy the relative camera orientation constraints. The class and camera orientations associated with the obtained set of elements are the estimated result.

A. Training the CNN

The CNN is trained to output descriptors that form 2-D manifolds where the Euclidean distance between descriptors of different classes becomes longer and the Euclidean distance between descriptors of the same class and similar orientation becomes shorter.

We use the CNN proposed in [13]. The CNN consists of two sets of a convolution layer and a max-pooling layer, followed by two fully connected layers as shown in Fig. 2. The convolution layers and the first fully connected layer employ a rectified linear (ReLU) activation function. The output of the second fully connected layer forms the 3-d descriptor of an input image.

The CNN is trained with a loss function $\mathcal{L}$ similar to one in [13] that consists of the following three terms:

$$\mathcal{L} = \mathcal{L}_{\text{triplets}} + \mathcal{L}_{\text{pairs}} + \lambda\|\omega'\|_2^2. \tag{1}$$

The first term $\mathcal{L}_{\text{triplets}}$ is called a triplet loss and is defined as in Eq. (2) by x_i, x_j, x_k that satisfy either of the conditions: (1) x_i, x_j belong to the same class and x_i, x_k belong to different classes, (2) x_i, x_j, x_k belong to the same class but the orientations of x_i, x_j are more similar than those of x_i, x_k.

$$\mathcal{L}_{\text{triplets}} = \sum_{(x_i,x_j,x_k)\in T} \max\left(0, 1 - \frac{\|f_w(x_i) - f_w(x_k)\|_2}{\|f_w(x_i) - f_w(x_j)\|_2 + \beta}\right) \tag{2}$$

where $f_w(x)$ is the output of the CNN given x.

The second term $\mathcal{L}_{\text{pairs}}$ is defined as in Eq. (3) by x_i, x_j that belong to the same class and their orientations are similar.

$$\mathcal{L}_{\text{pairs}} = \sum_{(x_i,x_j)\in P} \|f_w(x_i) - f_w(x_j)\|_2^2 \tag{3}$$

The third term is for normalizing all the parameters ω' of the CNN except the bias.

B. Building the database

Fig. 3 shows the structure of the database used for object recognition. The database consists of triplets $(c_i, \boldsymbol{R}_i, \boldsymbol{d}_i)$ where c_i is the class of the object in the training image x_i, $\boldsymbol{R}_i$ is the camera orientation of x_i, and $\boldsymbol{d}_i$ is the descriptor obtained from $f_w(x_i)$.

C. Object recognition

The previous method [13] searches the database for the element whose descriptor is most similar to the input descriptor, and returns the class and orientation associated with the descriptor.

In this research, we assume that the relative camera orientations between N cameras are known. Each input image x_i is provided to the learned CNN separately and the corresponding descriptor $\boldsymbol{d}_i$ is obtained. Then, the proposed method searches the database for the set of elements where their descriptors are most similar to the input descriptors and their orientations satisfy the relative camera orientation constraints, and returns the class and orientations associated with the elements.

To ensure that the obtained set of elements satisfies the relative camera orientation constraints, we enumerate combinations of viewpoints in the database that satisfy the relative camera orientation constraints and then select the most appropriate combination among them.

If the orientation of each camera is known, we can calculate the relative orientation between cameras as:

$$\boldsymbol{R}_{i,i+1} = \boldsymbol{R}_{i+1}^{-1}\boldsymbol{R}_i \tag{4}$$

where $\boldsymbol{R}_i$ is the rotation matrix from the i-th camera coordinate system to the world coordinate system.

To enumerate combinations of viewpoints that satisfy the relative camera orientation constraints, we select the m-th element in the database as the first viewpoint of the m-th combination. The camera orientation of the j-th ($j \geq 2$) viewpoint can be calculated from the camera orientation of the first viewpoint as:

$$\hat{\boldsymbol{R}}_j = \boldsymbol{R}_m\boldsymbol{R}_{m,j}^{-1}. \tag{5}$$

Since the same rotation matrix as $\hat{\boldsymbol{R}}_j$ is not always stored in the database, the j-th viewpoint is selected as the $k_{j,m}$-th element whose rotation matrix is closest to $\hat{\boldsymbol{R}}_j$ and whose class is equal to that of the first viewpoint as shown in Fig. 4. $k_{j,m}$ is calculated as:

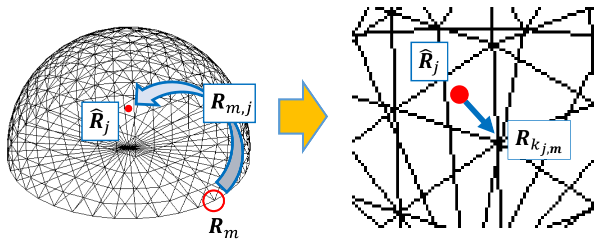

Fig. 4. Determination of viewpoints.

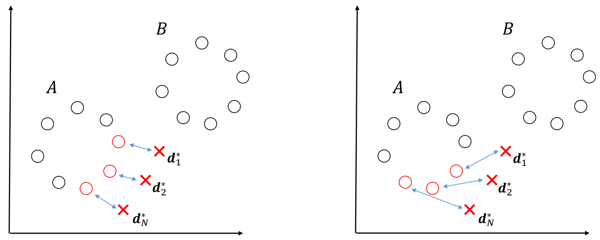

Fig. 5. Correspondences of descriptors.

$$k_{j,m} = \underset{i \le M, c_i = c_m}{\operatorname{argmin}} \cos^{-1}\left(\frac{Tr(\boldsymbol{R}_i^{-1}\hat{\boldsymbol{R}}_j) - 1}{2}\right) \tag{6}$$

where M is the number of elements in the database.

Then, the m-th combination of viewpoints is defined by the object class and a set of element indices as:

$$v_m = (c_m, k_{1,m}, k_{2,m}, \dots, k_{N,m}) \tag{7}$$

where $k_{1,m} = m$.

We repeat this process and obtain a set of combinations $V = \{v_1, \dots, v_M\}$.

The evaluation function of v_m is defined as:

$$E_1(v_m) = \sum_{i=1}^{N} \min(\|\boldsymbol{d}_{k_{i,m}} - \boldsymbol{d}_i^*\|_2^2, \lambda) \tag{8}$$

where $\boldsymbol{d}_i^*$ is the i-th input descriptor. The combination v_{m^*} that minimizes the Euclidean distance between corresponding descriptors is selected as shown in Fig. 5. m^* is calculated as:

$$m^* = \underset{m \le M}{\operatorname{argmin}} E_1(v_m). \tag{9}$$

The class and camera orientations associated with the obtained combination are the estimated results.

When the object is occluded by obstacles in some of the input images, the Euclidean distance between descriptors is not the correct measure to evaluate a combination. To suppress the effect of outliers, we set the threshold λ in Eq. (8).

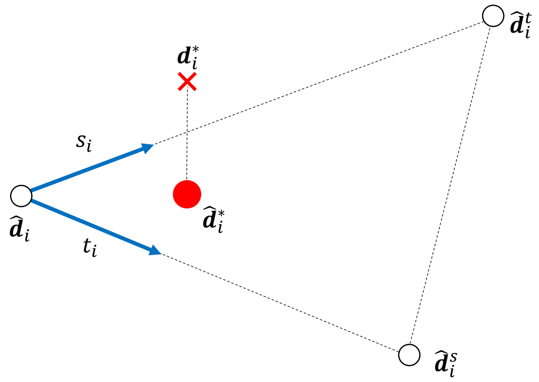
$\boldsymbol{d}_i^*$
$\hat{\boldsymbol{d}}_i^t$
s_i
$\hat{\boldsymbol{d}}_i$
$\hat{\boldsymbol{d}}_i^*$
t_i
$\hat{\boldsymbol{d}}_i^s$

Fig. 6. Interpolation of descriptors.

III. Optimization of Object Orientation

The method described in Section II searches the database for the set of descriptors that are similar to the input descriptors, and returns the class and camera orientations associated with the obtained descriptors. This means the estimated object orientation contains an error if none of the training images has exactly the same orientation as that of the input image. To reduce the estimation error, we optimize the object orientation by interpolating the viewpoints in the database.

The proposed method searches the database for the corresponding descriptor to an input descriptor as described in Section II and interpolates the descriptor with the descriptor itself and its nearby two descriptors.

Let the i-th input descriptor be $\boldsymbol{d}_i^*$ and the corresponding descriptor found in the database be $\hat{\boldsymbol{d}}_i$, the distance between the j-the descriptor in the database and $\hat{\boldsymbol{d}}_i$ can be defined as:

$$l_{ij} = \|\hat{\boldsymbol{d}}_i - \boldsymbol{d}_j\|_2. \tag{10}$$

K candidates for nearby descriptors to $\hat{\boldsymbol{d}}_i$ are selected in ascending order of distance l_{ij}. Note that $\hat{\boldsymbol{d}}_i$ is excluded from these candidates. Then, two descriptors $\hat{\boldsymbol{d}}_i^s, \hat{\boldsymbol{d}}_i^t$ with the shortest and the second shortest distance to $\boldsymbol{d}_i^*$ are selected from these candidates.

As shown in Fig. 6, the descriptor $\hat{\boldsymbol{d}}_i^*$ corresponding to the i-th input descriptor is interpolated by $\hat{\boldsymbol{d}}_i, \hat{\boldsymbol{d}}_i^s, \hat{\boldsymbol{d}}_i^t$ as:

$$\hat{\boldsymbol{d}}_i^* = \hat{\boldsymbol{d}}_i + s_i(\hat{\boldsymbol{d}}_i^s - \hat{\boldsymbol{d}}_i) + t_i(\hat{\boldsymbol{d}}_i^t - \hat{\boldsymbol{d}}_i) \tag{11}$$

where s_i, t_i are the parameters.

Similarly, as shown in Fig. 7, the object orientation $\hat{\boldsymbol{q}}_i^*$ corresponding to the i-th input camera orientation is interpolated using the same parameters as:

$$\hat{\boldsymbol{q}}_i^* \propto \hat{\boldsymbol{q}}_i + s_i(\hat{\boldsymbol{q}}_i^s - \hat{\boldsymbol{q}}_i) + t_i(\hat{\boldsymbol{q}}_i^t - \hat{\boldsymbol{q}}_i) \tag{12}$$

where $\hat{\boldsymbol{q}}_i, \hat{\boldsymbol{q}}_i^s, \hat{\boldsymbol{q}}_i^t$ are the camera orientations associated with $\hat{\boldsymbol{d}}_i, \hat{\boldsymbol{d}}_i^s, \hat{\boldsymbol{d}}_i^t$ and $\hat{\boldsymbol{q}}_i^*, \hat{\boldsymbol{q}}_i, \hat{\boldsymbol{q}}_i^s, \hat{\boldsymbol{q}}_i^t$ are represented by unit quaternions.

The evaluation function of $\boldsymbol{s}, \boldsymbol{t}$ is defined as:

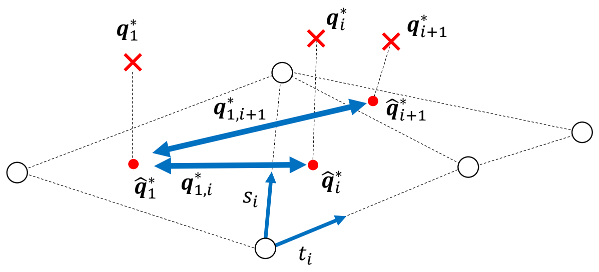

Fig. 7. Interpolation of object orientations.

$$E_2(\boldsymbol{s},\boldsymbol{t}) = \sum_{i=1}^{N} \min(\|\hat{\boldsymbol{d}}_i^* - \boldsymbol{d}_i^*\|_2^2, \lambda) + \alpha \sum_{i=2}^{N} \|(\hat{\boldsymbol{q}}_i^*)^{-1}\hat{\boldsymbol{q}}_1^* - \boldsymbol{q}_{1,i}^*\|_2^2 \tag{13}$$

where $\boldsymbol{s} = \{s_i\}, \boldsymbol{t} = \{t_i\}$, and $\boldsymbol{q}_{1,i}^*$ is the relative orientation between the first input camera orientation $\boldsymbol{q}_1^*$ and the i-th input camera orientation $\boldsymbol{q}_i^*$ that is calculated as:

$$\boldsymbol{q}_{1,i}^* = (\boldsymbol{q}_i^*)^{-1}\boldsymbol{q}_1^*. \tag{14}$$

The first term of Eq. (13) is a constraint term for preventing the interpolated descriptors from being away from the input descriptors. The second term of Eq. (13) is for satisfying the relative camera orientation constraints. By setting the parameters in this way, the interpolated descriptors are constrained to lie on the 2-D manifolds defined for each class of object.

We find $\boldsymbol{s}, \boldsymbol{t}$ that minimizes the evaluation function with a non-linear optimization method as:

$$\boldsymbol{s}, \boldsymbol{t} = \operatorname*{argmin}_{\boldsymbol{s},\boldsymbol{t}} E_2(\boldsymbol{s},\boldsymbol{t}). \tag{15}$$

IV. Experiments

The following experiments are performed to show the usefulness of the proposed method.

1) Evaluation with different number of input images.
2) Comparison with a method using multiple images.
3) Evaluation of object orientation optimization.
4) Evaluation with multiple images from our own dataset.

A. Evaluation with different number of input images

To demonstrate the usefulness of using multi-view images and known relative camera orientations, we perform the proposed method with different number of input images and evaluate the accuracy of object class and orientation estimation.

We use LineMOD dataset [4] to train the CNN and evaluate the methods. LineMOD dataset consists of real images of objects, rendered images of their 3D models, and their camera orientations. Each object is imaged from 301 viewpoints located on an upper hemisphere. We select 10 objects from the dataset and divide their images into 72000 images for training and $N \times 300$ images for evaluation where N is the number of viewpoints. The CNN is trained for 10000 epochs with Adam, the learning rate of 0.001, and the batch size of 300.

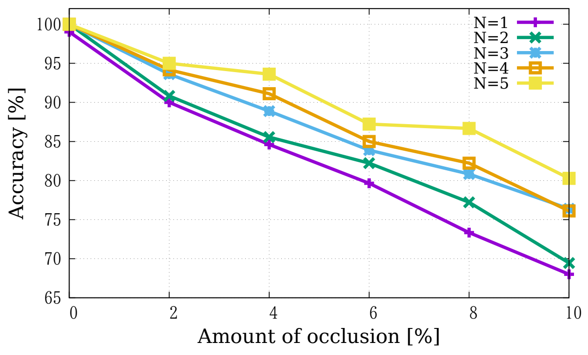

Fig. 8. Average accuracy of object class estimation.

TABLE I
Average accuracy of object class estimation [%].

	Amount of occlusion [%]					
N	0	2	4	6	8	10
1 [13]	99.03	90.01	84.61	79.65	73.34	68.00
2	100.00	90.83	85.56	82.22	77.22	69.44
3	100.00	93.61	88.89	83.89	80.83	76.39
4	100.00	94.17	91.11	85.00	82.22	76.11
5	100.00	95.00	93.61	87.22	86.67	80.28

To evaluate the performance of the methods when the object is partially occluded by obstacles, a rectangular area in each input image is filled with uniform noise whose location is randomly selected to overlap the object. The error of object orientation estimation is represented by the angle between the true value of the camera orientation $\boldsymbol{R}_{\mathrm{gt}}$ and the estimated object orientation $\hat{\boldsymbol{R}}$ defined as:

$$\theta = \cos^{-1}\left(\frac{Tr(\boldsymbol{R}_{\mathrm{gt}}^{-1}\hat{\boldsymbol{R}}) - 1}{2}\right). \tag{16}$$

Fig. 8 and Table I show the average accuracy of object class estimation. Fig. 9 and Table II show the average error of camera orientation estimation. $N \geq 2$ indicates the results of the proposed method using multi-view images and known relative camera orientations. $N = 1$ is equivalent to the results of the method using a single image [13] where relative camera orientation constraints are not used. We can see that the accuracy of object class and orientation estimation improves as the number of images increases at different amounts of occlusion. This demonstrates the usefulness of the proposed algorithm.

B. Comparison with a method using multiple images

We compare the proposed method with RotationNet [11] that uses multiple input images, and evaluate the accuracy of object class estimation.

In this experiment, we use MIRO dataset [11] to train the CNN and evaluate the methods. MIRO dataset consists of rendered images where 3D object models are imaged

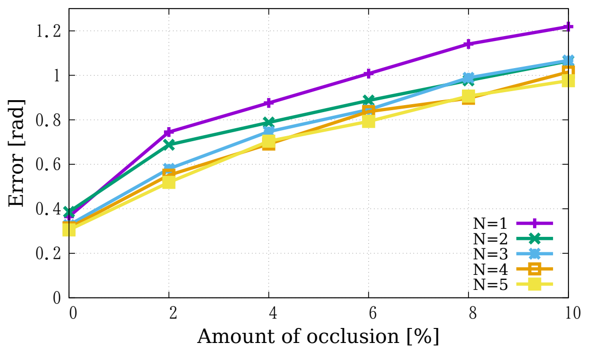

Fig. 9. Average error of object orientation estimation.

TABLE II

AVERAGE ERROR OF OBJECT ORIENTATION ESTIMATION [RAD].

	Amount of occlusion [%]					
N	0	2	4	6	8	10
1	0.365	0.745	0.876	1.008	1.141	1.219
2	0.386	0.688	0.788	0.887	0.977	1.064
3	0.327	0.580	0.747	0.846	0.989	1.067
4	0.314	0.551	0.691	0.837	0.897	1.015
5	0.305	0.519	0.704	0.793	0.907	0.976

from 160 viewpoints determined by dividing a sphere into 16 sections in the azimuth direction and 10 sections in the elevation direction. We select 10 objects from the dataset and divide their images into 14400 images for training and 1600 images for evaluation. The number of input images N is fixed to 4. The other experimental conditions are the same as those in the experiment in Section IV-A.

Fig. 10 and Table III show the average accuracy of object class estimation. We can see that the accuracy of object class estimation of the proposed method is superior to RotationNet at different amounts of occlusion. This demonstrates the usefulness of the proposed method.

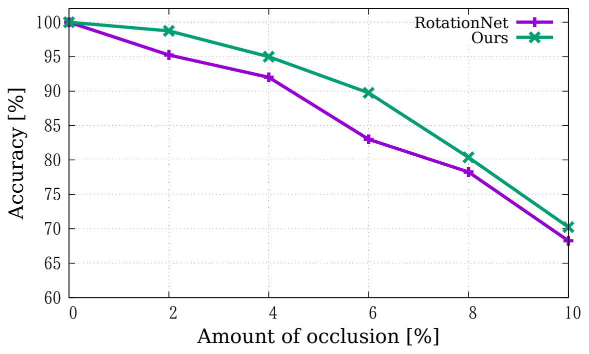

Fig. 10. Average accuracy of object class estimation.

TABLE III

AVERAGE ACCURACY OF OBJECT CLASS ESTIMATION [%].

	Amount of occlusion [%]					
	0	2	4	6	8	10
RotationNet	100.00	95.25	92.00	83.00	78.25	68.25
Ours	100.00	98.75	95.00	89.75	80.37	70.25

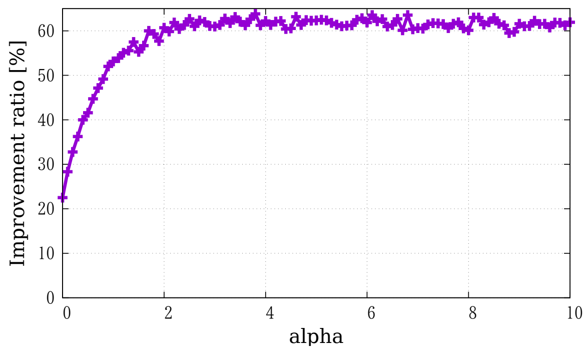

Fig. 11. Improvement ratio of object orientation estimation.

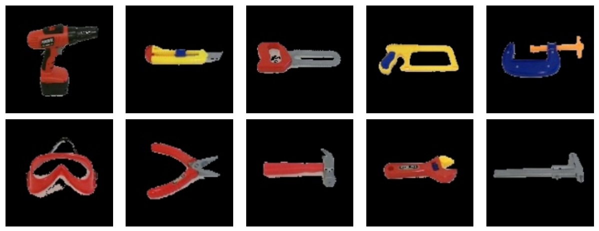

Fig. 12. Objects used in our own dataset.

C. Evaluation of object orientation optimization

To demonstrate the usefulness of object orientation optimization, we perform the proposed method with different α in Eq. (13) and evaluate the improvement ratio of object orientation estimation. Note that $\alpha = 0$ means no optimization.

The error of object orientation estimation is defined by Eq. (16). The improvement ratio is defined as the ratio of the number of images where the error is reduced after optimization to the total number of images used for evaluation.

We use LineMOD dataset and the experimental conditions are the same as those in the experiment in Section IV-A.

Fig. 11 shows the improvement ratio for different α. We can see that the improvement ratio increases as α increases and saturates at around 60% after a while that means more than half of the estimates are improved. This demonstrates the usefulness of the object orientation optimization.

D. Evaluation with multiple images from our own dataset

To demonstrate the usefulness of the proposed method for images taken from the cameras on our humanoid robot, we evaluate the accuracy of object class and object orientation estimation using our own dataset.

Our dataset consists of images of 10 objects as shown in Fig. 12 that are captured from 128 viewpoints determined by dividing a sphere into 16 sections in the azimuth direction and 8 sections in the elevation direction. Several images are taken from each viewpoint by slightly moving the camera and the background region is removed from each captured image by background subtraction. The dataset is divided into 115200 images for training and 1280 images for evaluation.

The CNN is trained for 1000 epochs. The other experimental conditions are the same as those in the experiment in Section IV-A.

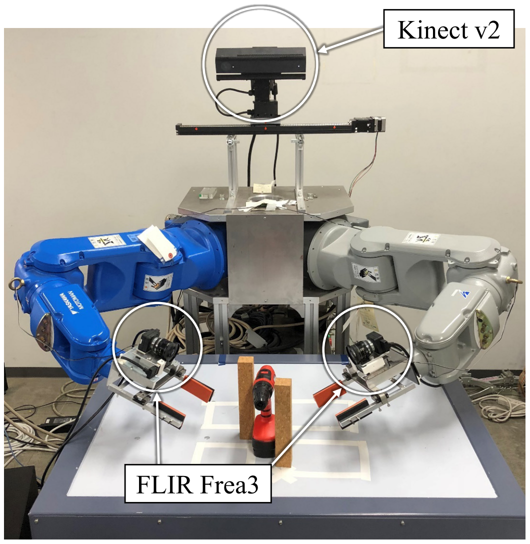

Fig. 13. Multiple cameras on our humanoid robot.

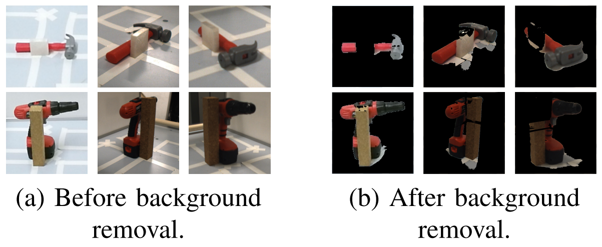

Fig. 14. Images where the object is partially occluded by obstacles.

To evaluate the accuracy of object class and orientation estimation, the proposed method is applied to three sets of images. The first set consists of 1280 images described above. The second set consists of 30 images taken from the three cameras on our humanoid robot as shown in Fig. 13. A rectangular area of each image in the first and second sets is filled with uniform noise as described in Section IV-A. The third set consists of 30 images taken from the three cameras on our humanoid robot where the object is partially occluded by obstacles as shown in Fig. 14.

For the first and second set of images, Fig. 15 and Table IV show the average accuracy of object class estimation, and Fig. 16 and Table V show the average error of object orientation estimation. We can see that the accuracy of object class and orientation estimation of the proposed method is superior to the conventional method [13] at different amounts of occlusion.

For the third set of images, Table VI show the average accuracy of object class estimation and Table VII show the average error of object orientation estimation. We can also see that the accuracy of object class and orientation estimation of the proposed method is superior to the conventional method [13] at different amounts of occlusion.

This demonstrates the usefulness of the proposed method for images taken from the cameras on our humanoid robot.

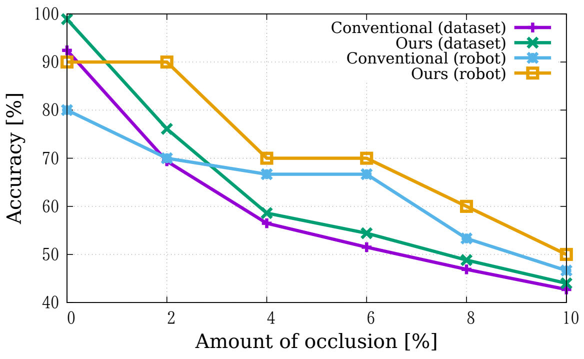

Fig. 15. Average accuracy of object class estimation.

TABLE IV

AVERAGE ACCURACY OF OBJECT CLASS ESTIMATION [%].

Method	Amount of occlusion [%] 0	2	4	6	8	10
Conv. [13] [a]	92.42	69.38	56.48	51.51	46.90	42.73
Ours [a]	98.89	76.11	58.61	54.41	48.83	44.00
Conv. [13] [b]	80.00	70.00	66.67	66.67	53.33	46.67
Ours [b]	90.00	90.00	70.00	70.00	60.00	50.00

[a] Using the first set of images.
[b] Using the second set of images.

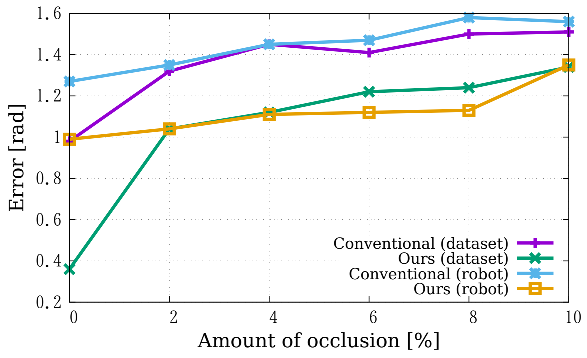

Fig. 16. Average error of object orientation estimation.

TABLE V

AVERAGE ERROR OF OBJECT ORIENTATION ESTIMATION [RAD].

Method	Amount of occlusion [%] 0	2	4	6	8	10
Conv. [13] [a]	0.98	1.32	1.45	1.41	1.50	1.51
Ours [a]	0.36	1.04	1.12	1.22	1.24	1.34
Conv. [13] [b]	1.27	1.35	1.45	1.47	1.58	1.56
Ours [b]	0.99	1.04	1.11	1.12	1.13	1.35

[a] Using the first set of images.
[b] Using the second set of images.

TABLE VI

AVERAGE ACCURACY OF OBJECT CLASS ESTIMATION [%].

Method	No obstacle	Small obstacle	Large obstacle
Conv. [13] [a]	80.00	76.76	26.27
Ours [a]	90.00	80.00	30.00

[a] Using the third set of images.

TABLE VII

AVERAGE ERROR OF OBJECT ORIENTATION ESTIMATION [RAD].

Method	No obstacle	Small obstacle	Large obstacle
Conv. [13] [a]	1.27	1.45	1.64
Ours [a]	0.73	0.99	1.47

[a] Using the third set of images.

V. CONCLUSIONS

In this research, we propose a method of estimating object class and orientation given multiple input images assuming the relative camera orientations are known. Input images are transformed to descriptors on 2-D manifolds defined for each class of object through a CNN, and the object class and orientation that minimize the distance between input descriptors and the descriptors associated with the estimated object class and orientation are selected. The object orientation is further optimized by interpolating the viewpoints in the database.

The usefulness of the proposed method has been demonstrated by comparative evaluation with other methods using publicly available datasets. The usefulness of the proposed method has also been demonstrated by recognizing images taken from the cameras on our humanoid robot using our own dataset.

The accuracy of object class and orientation estimation using our own dataset is worse than those using publicly available datasets. This is probably due to differences in cameras and lighting conditions between at the time of building the datasets and at the time of capturing images from the cameras on our humanoid robot. Future works include increasing the variety of the dataset and reviewing the structure of the CNN to improve the accuracy of object class and orientation estimation.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number JP19K12170.

REFERENCES

[1] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *Proc. of European Conference on Computer Vision*, 2014, pp. 536–551.

[2] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Proc. of International Conference on Robotics and Automation*, 2011, pp. 1817–1824.

[3] C.-Y. Tsai and S.-H. Tsai, "Simultaneous 3d object recognition and pose estimation based on rgb-d images," *IEEE Access*, vol. 6, pp. 28 859–28 869, 2018.

[4] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Proc. of Asian Conference on Computer Vision*, 2012, pp. 548–562.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[7] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," *ArXiv*, vol. abs/1703.09507, 2017.

[8] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. of Computer Vision and Pattern Recognition*, 2019, pp. 4685–4694.

[9] J. M. Wong, V. Kee, T. Le, S. Wagner, G.-L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. M. Johnson, J. Wu, B. Zhou, and A. Torralba, "Segicp: Integrated deep semantic segmentation and pose estimation," in *Proc. of International Conference on Intelligent Robots and Systems*, 2017, pp. 5784–5789.

[10] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. of International Conference on Computer Vision*, 2015, pp. 945–953.

[11] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proc. of Computer Vision and Pattern Recognition*, 2018, pp. 5010–5019.

[12] Y. Nakajima and H. Saito, "Robust camera pose estimation by viewpoint classification using deep learning," *Computational Visual Media*, vol. 3, no. 2, pp. 189–198, 2017.

[13] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proc. of Computer Vision and Pattern Recognition*, 2015, pp. 3109–3118.

[14] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proc. of Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.