Active Object Recognition with Trained Multi-view Based 3D Object Recognition Network

Yunyi Guan[1] and Asako Kanezaki[2]

***Abstract*— We tackle the active object recognition (AOR) problem, in which agents learn effective exploration actions to actively acquire new images with partial knowledge of observation for better object recognition. Previous studies have typically used reinforcement learning to jointly train a single-input classifier and a policy network to learn to find new observations. However, this joint learning process is very laborious and cannot reach the accuracy level of existing object classifiers. It is also reported that when using highly accurate classifiers such as ResNet, the active vision capabilities of such jointly trained models will disappear. To overcome these problems, we propose a framework using a highly accurate pre-trained multi-view-based 3D object recognition network to train AOR agents by reinforcement learning, aiming at higher accuracy of object recognition in a lighter way. Through evaluation experiments with several benchmark datasets, we show that the performance of our approach outperforms several previous studies.**

I. INTRODUCTION

Driven by benchmarks such as human-taken images or videos, 3D object recognition has focused on extrapolating semantic labels from pre-defined images, employing every conceivable possible observation to gather complete information – so-called "passive object recognition".

However, not every view is needed in practice to fully understand 3D shapes. Therefore, robot visual recognition requires inferences from observations and decisions about what to observe. So, the robot should learn to actively choose the information-rich views within a limited time budget to guide better recognition. This leads to the active object recognition (AOR) problem: learning a suitable view-changing policy that enables the robot to actively obtain new views of objects, thus speeding up the real-time recognition process and achieving better accuracy, as shown in Fig. 1.

The standard framework of AOR usually consists of three modules: object recognition, evidence fusion, and action selection. Specifically, pre-rendered images are input to extract features, and aggregated historical information is used for recognition and action selection. By jointly training the three modules, agents can effectively explore 3D objects. However, this joint learning process is time-consuming and unable to reach the accuracy level of the existing 3D object recognition models. Moreover, it is reported that when using highly accurate classifiers such as ResNet [5], the active vision capabilities of jointly trained agents will disappear.

To alleviate the above issues, we propose a new AOR method, which uses a highly accurate pre-trained multi-view-based 3D object recognition network to train AOR agents by reinforcement learning. Previous work has either used small convolutional neural networks or single-input 2D classification networks, as shown in Table I. Moreover, a common issue in existing attention-based models is the unbalanced training of view estimation and shape classification [1]. Our approach directly uses a matrix accumulating historical information to avoid this problem.

[1]Yunyi Guan, graduated from School of Computing, Tokyo Institute of Technology, Japan `guanyunyi0728@gmail.com`

[2]Asako Kanezaki, Associate Professor, School of Computing, Tokyo Institute of Technology, Japan `kanezaki@c.titech.ac.jp`

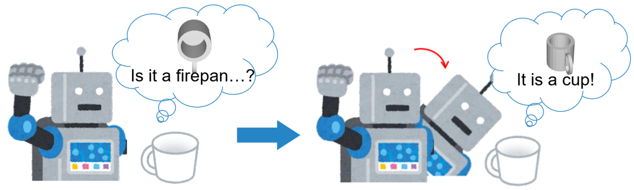


Fig. 1: Illustration of active object recognition (AOR).

Our contributions are summarized as follows.

- For the first time, a highly accurate pre-trained multi-view-based 3D object recognition network was used as the recognition module for the AOR system.
- No complex training methods of joint learning were used. We use a trained object classifier with fixed parameters instead to lighten the pipeline while achieving better accuracy.
- No recurrent structure like RNN [14] or LSTM [4] is used for the evidence fusion.

II. RELATED WORK

A. Multi-view based 3D Object Recognition.

Among traditional 3D object recognition methods, the multi-view-based method is the easiest to understand and has higher accuracy, which first projects 3D objects into multiple views, extracts corresponding view features, and then fuses features for accurate recognition. A typical approach is MVCNN [19], which projects the point cloud to different views as input and integrates multi-views in a view-pooling layer. It requires a complete set of multi-view images recorded from all the pre-defined views for object inference. Unlike MVCNN, RotationNet [10] can jointly estimate the pose and category of an object using a partial set of multi-view images that a moving camera may sequentially observe.

It is worth noting that multi-view-based object recognition is also in line with the behavioral characteristics of robot vision; that is, for a specific object, the robot can continuously obtain new observations by changing the view, thus gradually increasing the credibility of recognition. This characteristic

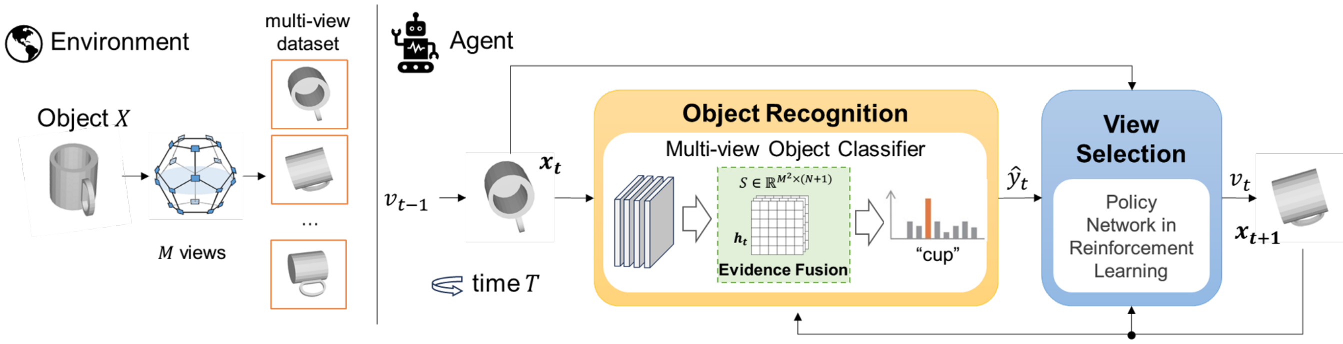


Fig. 2: Overview of the proposed AOR system. X is a 3D object, T is the total timestep size, and M and N are the total number of views and categories. At each timestep t, the object recognition module takes image x_t observed from the last output view v_{t-1} as the input to make category prediction $\hat{y}_t$, inside the historical information h_t will be recorded in the matrix S. After that, x_t and $\hat{y}_t$ are used in the view selection module to predict the next appropriate view v_t.

TABLE I: Comparison of previous work and proposed method.

Method	Object Recognition	Evidence Fusion	When to predict	Learning
LookAhead [7], [8]	GoogleNet + fully-connected layer	RNN	after T steps	joint learning
LookAround [9], [17]	pooling layers + fully-connected layer	LSTM	after T steps	joint learning
VERAM [1]	AlexNet + fully-connected layer	LSTM	after T steps	joint learning
Lingering [2]	3-layer convolutional layers+ linear layers	LSTM	at each timestep	joint learning
Ours	RotationNet	matrix	at each timestep	two-phase learning

of multi-view networks can lead to an interesting problem: The AOR system should adopt an intelligent control policy so that the robot can move to the next appropriate view to avoid undesirable visual conditions and obtain differentiated information to accelerate real-time recognition. To this end, it can be considered to use the trained multi-view-based classifiers as the recognition module of AOR systems. RotationNet, in particular, is able to incrementally input images and improve the recognition accuracy, which inspires our work.

B. Active Object Recognition.

AOR was first proposed in [21], which developed a system integrating a camera-mounted robot arm and a mobile base. Later, reinforcement learning has been applied to AOR and proved to be effective. Early work can be found in [15], which proposed a system that fuses information by probabilistically encoding 2D views and enforcing actions that lead to discriminative views. With the powerful expressive power of deep networks, Malmir et al. [13] use object beliefs as the representation of the current states to learn actions by deep Q-learning to minimize the overall classification error. The visual features of each view are pre-trained offline. In contrast, Jayaraman et al. [7], [8] first proposed an end-to-end pipeline to jointly learn object recognition, evidence fusion, and action selection. On top of this work, they also developed an approach [9], [17] where the learned policies are not tied to any recognition task nor the particular semantic content seen during training, so the downstream tasks are extended to panoramic natural scenes. To solve the unbalance that the classifier is easy to overfit while the view selection is usually poorly trained with joint learning, Chen et al. [1] proposed three view augmentation strategies, and prediction is made only at the last timestep. Wei et al. [20] applies meta-learning to deal with the challenges of AOR in few-shot settings. Fan et al. [3] combined AOR with lifelong learning; instead of only training on the dataset with a fixed number of categories, new categories are added gradually. Another work of Fan et al. [2] aims to solve the problem of selecting several views with correct predictions, so adversarial disturbance is added to the policy network. Different from using reinforcement learning, Liu et al. [12] introduce STN [6] to RNN [14] to form an end-to-end differentiable 3D attention structure, through which 3D spherical coordinates can simply be regressed to train the classification and view selection losses.

However, all these works use single-input classification networks for 3D recognition and focus on joint learning of the three modules of object recognition, evidence fusion, and action selection, which lead to laborious training processes. Also, because of the unbalanced training problem, even the state-of-the-art methods cannot achieve the level of pure object recognition accuracy. We attempt to use a pre-trained multi-view based 3D object recognition network, RotationNet [10], to aim at higher accuracy in a lighter way. Due to the special structure of RotationNet [10], our AOR system can be realized without additional evidence fusion module, thus making the pipeline more streamlined.

III. METHOD

A. Problem Setup

AOR is formulated as an agent interacting with 3D objects X over T timesteps. At each timestep t, it moves to view v_t (where v_1 is random), then obtains new observation x_t and makes prediction $\hat{y}_t$ for object category label (the total number of categories is N). The agent's goal is to perform exploratory actions to gradually improve classification ac-

Algorithm 1 Training AOR system.

// Step 1: Pre-training object classifier
Input: Training set $D_{\text{train}} = \{x_i, y_i\}_{i=i}^{N}$, number of epochs E, timestep $T = 5$
Initialization: Object classifier F_{class}, matrix $S \leftarrow \emptyset$
for $e = 1$ to E **do**
  $[h^j]_{j=1}^{20} \leftarrow F_{\text{class}}([x^j]_{j=1}^{20}, S)$ // input multi-views
  $\hat{y} \leftarrow S$ // S is updated with $[h^j]_{j=1}^{20}$
  loss $\leftarrow$ CrossEntropyLoss$(\hat{y}, y)$
  Update F_{class}
end for
Output: Pre-trained object classifier F_{class}

// Step 2: Training view selection module
Input: Training set $D_{\text{train}} = \{x_i, y_i\}_{i=1}^{N}$, trained object classifier F_{class}, number of episodes E', timestep $T = 5$
Initialization: Action space A, observation space O, policy network F_{policy}
for $e = 1$ to E' **do**
  for $t = 1$ to T **do**
    $v_t \in A \leftarrow F_{\text{policy}}$
    Get x_t according to v_t
    $h_t \leftarrow F_{\text{class}}(x_t, S)$ // input single view
    $\hat{y}_t \leftarrow S$ // S is updated with h_t
    Observe reward R_t
    Update F_{policy}
  end for
end for
Output: Trained policy network F_{policy}

curacy effectively, so it keeps output history $h = [h_1, ..., h_T]$ for updating the internal representation after obtaining each new observation. Later, x_t and h are fed to the reinforcement learning model to learn to select the next view v_{t+1} actively. After T timesteps, the agent should be able to classify the object, and the accuracy should gradually increase.

In our setting, we follow the view setting of the case (ii) in [10], which places virtual cameras on the $M = 20$ vertices of a dodecahedron encompassing the object, as shown in the left side of Fig. 2. The views are completely uniformly distributed in 3D space, and v_t are defined as discrete natural numbers, that is, $v_t \in [0, 1, ...19]$. Even if each sample has a different pose, the view number is bound to the observation from the corresponding view. Therefore, we set the action as v_t directly, i.e., action$_t = v_t \in [0, 1, ...19]$.

B. Architecture

The whole system can be simplified into two modules as shown on the right side of Fig. 2. The object recognition module has two jobs: one is to encode what was observed, and another is to maintain an internal state that encodes the agent's knowledge of the environment and summarizes the information extracted from the history of past observations so as to make better category predictions and view selection. RotationNet [10] serves as the classifier in our object recognition module. For each object, RotationNet will initialize a matrix $S \in \mathbb{R}^{M^2(N+1)}$ to store the predicted scores for each view, which can be used as historical information. Each row corresponds to a view from M views, and each column corresponds to the score for a category. Each view has $N+1$ predicted score, where N is the number of categories, and $+1$ is to denote the "incorrect view" class, indicating how likely the estimated view is incorrect. After updating S with the raw output obtained by feeding image x_t into RotationNet at each step, S is in turn used to calculate the current object pose and category. Aimed at this characteristic, we use S as the evidence fusion module. In this case, $h = S$, and h_t is the row corresponding to the current view in S.

Based on current observations and historical information, the view selection module controls where the agent is to observe next. It inputs the x_t and h_t and outputs the next view v_t. We choose TRPO [18] to represent this part and set up a two-part reward function, where reward r_1 is for better object recognition and reward r_2 is for reducing view repetition. When the prediction is correct, we set reward $r_1 = 1$; otherwise, $r_1 = -1$. Moreover, because selecting a view that has appeared before can lead to stagnation and makes no sense, in order not to interfere with the learning for the recognition when v_t only appears once, we set another reward $r_2 = 0$, otherwise $r_2 = -1$. The final reward is $R = r_1 + r_2$.

C. Learning

The learning process is divided into two stages: pre-training of the object recognition module, RotationNet [10], and training of the view selection module, TRPO [18]. Each object follows a timestep of length $T = 5$ to update the weights of the networks. Each step makes a category prediction and then gives rewards according to the reward function. Algorithm 1 shows our whole training process.

TABLE II: Active object recognition accuracy on different datasets.

	ModelNet10	ModelNet40	ShapeNetCore55
LookAhead [8]	92.50 ± 0.07	-	63.4 ± 0.3
LookAround [9]	92.50	89.00	-
VERAM [1]	95.30	92.10	-
Lingering [2]	-	-	76.9 ± 0.3
Ours	**95.50 ± 0.0036**	**95.14 ± 0.0027**	**85.75 ± 0.0017**

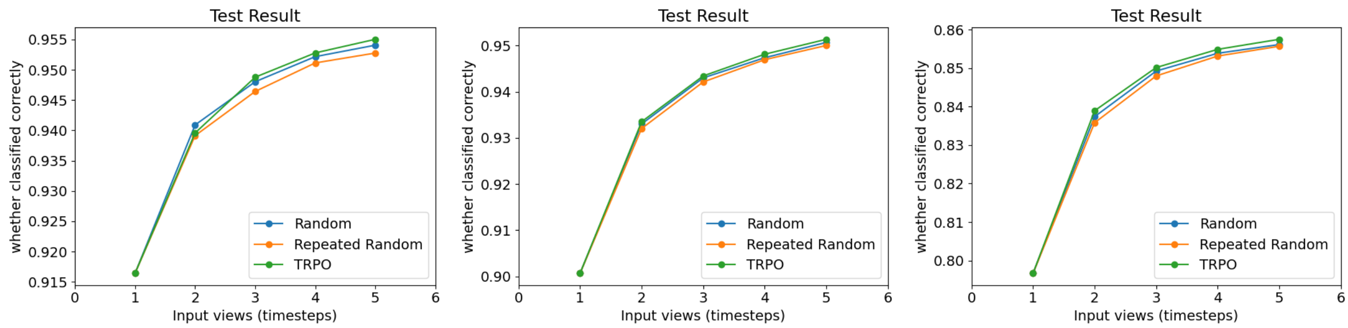
TABLE III: Average time consuming. Training time is for overall episodes and testing time is for one object.

	ModelNet10	ModelNet40	ShapeNetCore55
Training classifiers	3.222 (h)	14.504 (h)	22.411 (h)
Training agents	7.624 (h)	19.572 (h)	25.856 (h)
Testing	0.078 (sec)	0.222 (sec)	0.286 (sec)

IV. EXPERIMENTS

A. Dataset

ModelNet ModelNet40 consists of 12,311 CAD-generated meshes in 40 categories, of which 9,843 are used for training



(a) ModelNet10 (b) ModelNet40 (c) ShapeNetCore55

Fig. 3: Object classification accuracies on different datasets in the AOR setting. Static object classifier accuracy (with multi-view images as input) is (a) 96.15%, (b) 96.11%, and (c) 86.23%.

and 2,468 for testing. ModelNet10 dataset is a part of ModelNet40, containing 4,899 pre-aligned shapes from 10 categories, 3,991 for training and 908 for testing.

ShapeNetCore55 We use the ShapeNetCore subset of ShapeNet which contains about 51,300 3D models over 55 categories, 36,147 for training, and 5,165 for testing.

For all the datasets, we followed the $M = 20$ fixed view settings in [10] and made multi-view images using the rendering software published in [19].

B. Baseline

Random This baseline indicates a random selection forbidding repeated views. T different views are randomly selected with the same probability. Because the 3D model should have the same view at $t = 1$ when different methods are tested, we use this random baseline to generate initial views.

Repeated Random The difference from Random is that T views can be repeated when selected. Even with reward r_2 for reducing view repetition , the reinforcement learning models will inevitably select appeared views, so to guarantee some level of a fair comparison, we add this baseline.

C. Implementation details

We train RotationNet using pre-trained AlexNet [11] architecture. The final classification accuracy is 96.15% for ModelNet10, 96.11% for ModelNet40, and 86.23% for ShapeNetCore55. The parameters of the TRPO [18] policy network are updated using policy gradient from Stable Baselines3 [16].

D. Results

1) Active Object Recognition Results: Predictions at each step are used to measure the accuracy. We compare the performance of the proposed method with the previous works and the baselines described above. The results are the average of over 50 runs with different initializations.

Table II shows the average accuracy of our AOR models compared with previous works. It can be seen that on all the datasets, our method outperforms previous AOR systems. The small standard deviation results also show that it can achieve consistently high-level accuracies over several tests.

Fig. 3 shows the average accuracy of our AOR models compared with random baselines. It can be seen that all AOR models can achieve similar accuracy levels to the used RotationNet, and the active vision ability isn't failing as reported by Lingering [2]. Note that even the random baseline achieves high accuracy since RotationNet itself has the ability to improve its classification accuracy with incremental inputs. Nevertheless, our method outperformed this challenging baseline. In addition, the lower the level of raw accuracy of the classifier, the greater the difference in accuracy between the random baseline and the AOR model. This is the same as our intuition that the task of AOR is better suited for the case where the classifier is imperfect.

2) Training and Testing Time: In addition to the accuracy results, we also measured training time for all episodes and test time for one object of the proposed method, as shown in Table III. We trained the classifier with a single GTX1080 and trained and tested the agents with a single Quadro P6000. In both training and testing, our approach achieves active object recognition quickly, thanks to our lightweight pipeline.

E. Limitation and Future Work

Our goal is effective recognition, so we can design rewards using both appearance and geometric features acquired by RotationNet. Since we use a pre-trained classifier, our method will be inferior when the classifier itself is not optimal, which also illustrates that without joint learning the recognition and view selection modules cannot be optimized simultaneously. This could inspire future work on overcoming the problem of active vision disappearing when using a high-level classifier in the context of joint learning to achieve accuracy beyond the level of that classifier.

V. CONCLUSION

In this paper, we proposed an AOR framework based on a pre-trained multi-view based 3D object recognition network that is used to train AOR agents. Experimental results on three 3D object datasets show that the proposed method can achieve higher accuracy than previous AOR systems, and closer accuracy to existing object recognition classifiers level in a lighter way without laborious joint learning.

References

[1] Songle Chen, Lintao Zheng, Yan Zhang, Zhixin Sun, and Kai Xu. Veram: View-enhanced recurrent attention model for 3d shape classification. *IEEE transactions on visualization and computer graphics*, 25(12):3244–3257, 2018.

[2] Lei Fan and Ying Wu. Avoiding lingering in learning active recognition by adversarial disturbance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4612–4621, 2023.

[3] Lei Fan, Peixi Xiong, Wei Wei, and Ying Wu. Flar: a unified prototype framework for few-sample lifelong active recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15394–15403, 2021.

[4] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[7] Dinesh Jayaraman and Kristen Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 489–505. Springer, 2016.

[8] Dinesh Jayaraman and Kristen Grauman. End-to-end policy learning for active visual categorization. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1601–1614, 2018.

[9] Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1238–1247, 2018.

[10] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5010–5019, 2018.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[12] Min Liu, Yifei Shi, Lintao Zheng, Kai Xu, Hui Huang, and Dinesh Manocha. Recurrent 3d attentional networks for end-to-end active object recognition. *Computational Visual Media*, 5:91–104, 2019.

[13] Mohsen Malmir, Karan Sikka, Deborah Forster, Javier R Movellan, and Garison Cottrell. Deep q-learning for active recognition of germs: Baseline performance on a standardized dataset for active learning. In *Proceedings of British Machine Vision Conference (BMVC)*, pages 161–1, 2015.

[14] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.

[15] Lucas Paletta and Axel Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31(1-2):71–86, 2000.

[16] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

[17] Santhosh K Ramakrishnan, Dinesh Jayaraman, and Kristen Grauman. Emergence of exploratory look-around behaviors through active observation completion. *Science Robotics*, 4(30):eaaw6326, 2019.

[18] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[19] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[20] Wei Wei, Haonan Yu, Haichao Zhang, Wei Xu, and Ying Wu. Metaview: Few-shot active object recognition. *arXiv preprint arXiv:2103.04242*, 2021.

[21] David Wilkes and John K Tsotsos. *Active object recognition*. University of Toronto, 1994.

Supplementary Material for Active Object Recognition with Trained Multi-view Based 3D Object Recognition Network

Yunyi Guan[1] and Asako Kanezaki[2]

I. Qualitative Evaluation Results

We show the view selection order of our method for several samples in different datasets.

A. ModelNet10

From Fig. 2, it can be seen that selecting repeated and misclassified views at the initial stage will lead to a reduction in accuracy (as shown in the second row). Since the accuracy of the classifier is high (96.15%), the category prediction can be maintained correctly even if chosen randomly. Still, our approach can select more views less similar to the initial view, such as views at $t = 4$ and $t = 5$.

B. ModelNet40

From Fig. 3, it can be seen that, even if a repeated view is selected at a later timestep, the category prediction will have an error turn into a correct one because the classifier has accumulated history (as shown in the second row). Since the accuracy of the classifier is high (96.11%), it is likely to consistently provide rewards to policy no matter which view is selected, so the results of our agent are not much different from the random baseline.

C. ShapeNetCore55

From Fig. 4, it can be seen that, when the classifier accuracy is not perfect (86.23%), even if the initial view is predicted correctly, it is possible to turn to make a wrong prediction after selecting an inappropriate view (as shown in the first and second rows), which can be avoided by our agent. This is the same as our intuition that AOR is better suited to the situation when the recognizer is imperfect.

II. AOR in Continuous View Setting

We also tried to propose an AOR framework in a more complex continuous view setting. The advantage of the continuous views is that it can perform subtle changes in action to access observations that are not accessible under the discrete views, such as the process of changing view between adjacent vertices. Note that we do not use joint learning either in this setting.

[1]Yunyi Guan, graduated from School of Computing, Tokyo Institute of Technology, Japan guanyunyi0728@gmail.com

[2]Asako Kanezaki, Associate Professor, School of Computing, Tokyo Institute of Technology, Japan kanezaki@c.titech.ac.jp

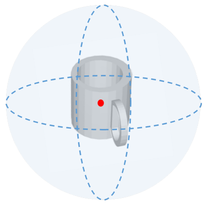


Fig. 1: Continuous view setting.

A. View Setup

In continuous view setting, the observation space is represented as a sphere encompassing the object as shown in Fig. 1, view v_t is located on its surface and is defined as a tuple of azimuth and elevation, that is, $v_t = (\mathrm{azim}_t, \mathrm{elev}_t)$, where $\mathrm{azim}_t \in [0°, 360°)$ and $\mathrm{elev}_t \in [-90°, 90°)$.

3D objects need to be rendered online at each step according to action a_t to obtain new observation x_t. Since the initial pose of each object is different, we set the action a_t to represent the relative angle, that is, $a_t = (\Delta\mathrm{azim}_t, \Delta\mathrm{elev}_t)$. Obviously, in order to obtain more image features, we do not want the actions taken in the continuous views to be too small, but if the actions are too large, it is difficult to reflect the difference from the discrete view setting, and thus lose the significance. Therefore, we set a_t to not exceed the angle difference between the adjacent vertices in the discrete view setting (the difference in azimuth between each adjacent vertex of the regular dodecahedron is 72°, and the elevation difference is 36°), i.e. $\Delta\mathrm{azim}_t \in [5, 72]$, $\Delta\mathrm{elev}_t \in [5, 36]$.

B. Architecture

In the continuous view setting, the view is rendered online, so the inputs are 3D object X and the last a_{t-1}. Since RotationNet can only handle discrete views and needs to be trained with multi-views that contain complete shape information rendered from fixed viewpoints, we use pre-trained ResNet18 [3] this time. Please note that for both training and testing, the input to ResNet18 is one image.

Expect for accumulating the historical information, to ensure that the classifier in the continuous view setting has the same ability to improve accuracy with increasing input as in the discrete view setting, we add an LSTM [1] after ResNet18, and then add a fully-connected linear layer for classification.

C. Learning

Since the views outputted by the subsequent view selection module are arbitrary, it's no longer appropriate to train

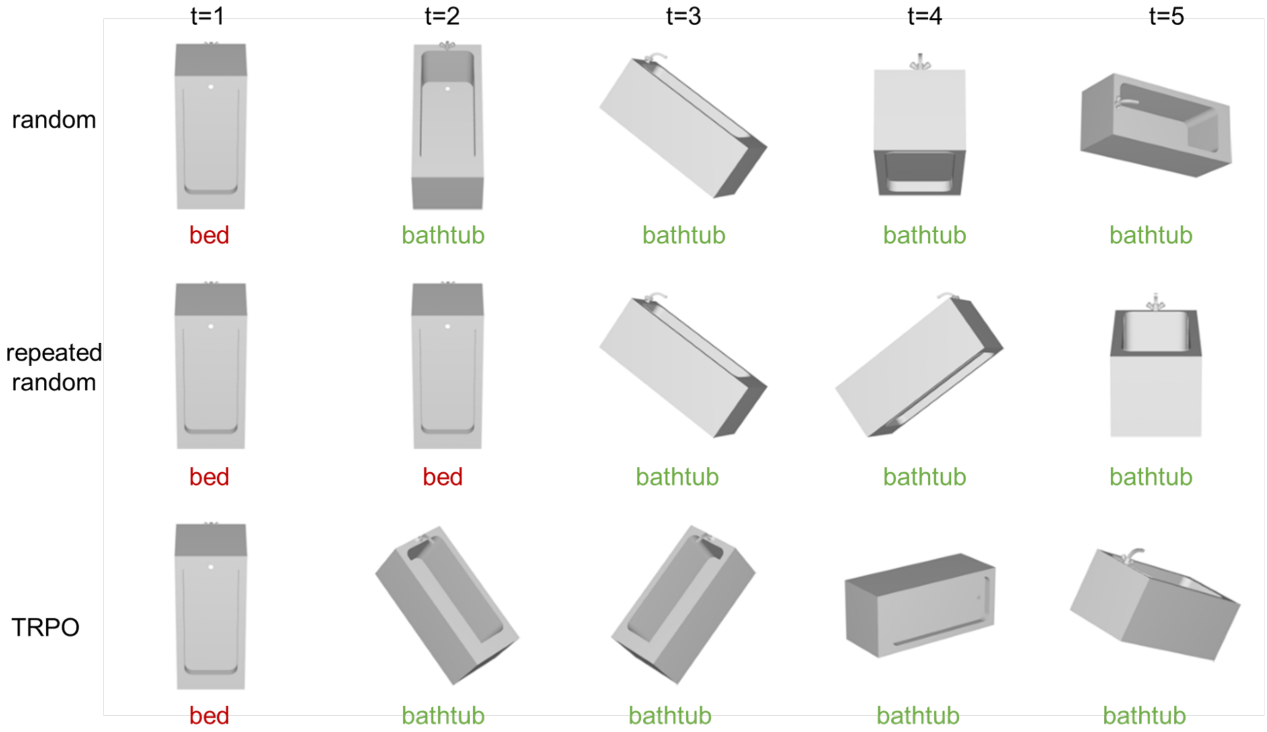


Fig. 2: View selection order for bathtub_0112 in ModelNet10.

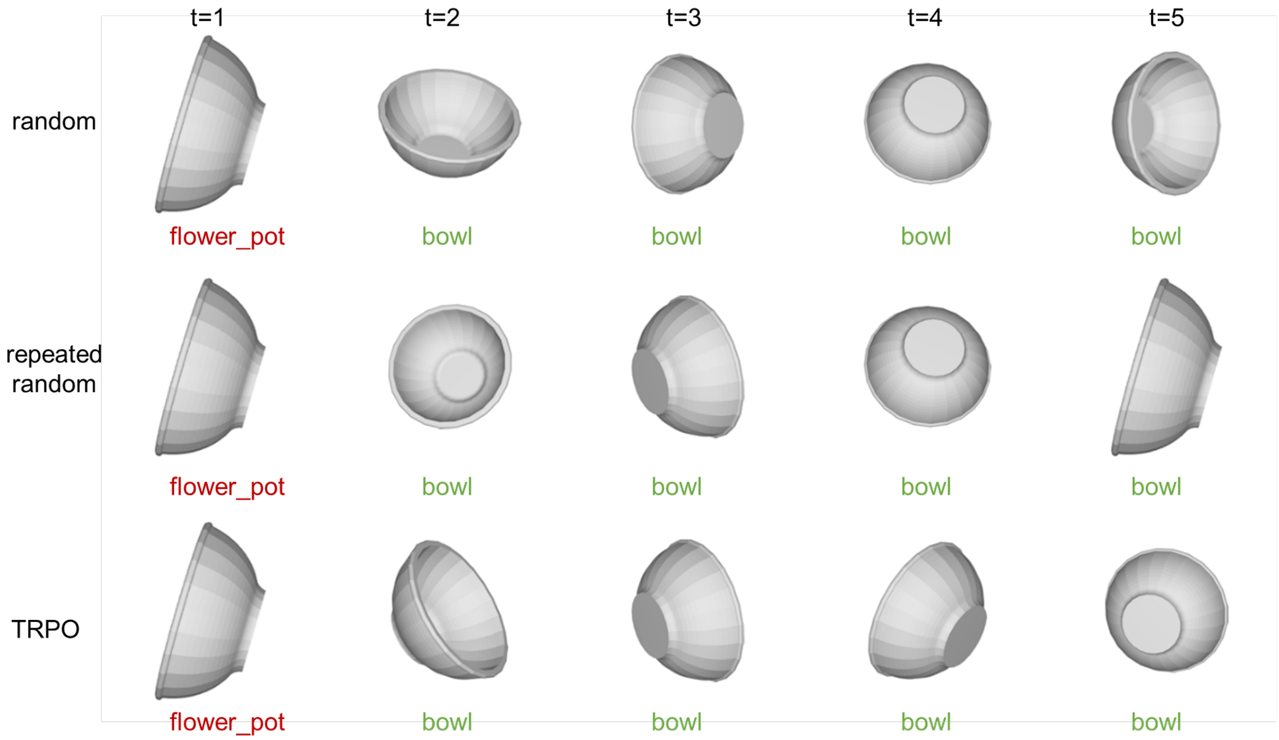


Fig. 3: View selection order for bowl_0068 in ModelNet40.

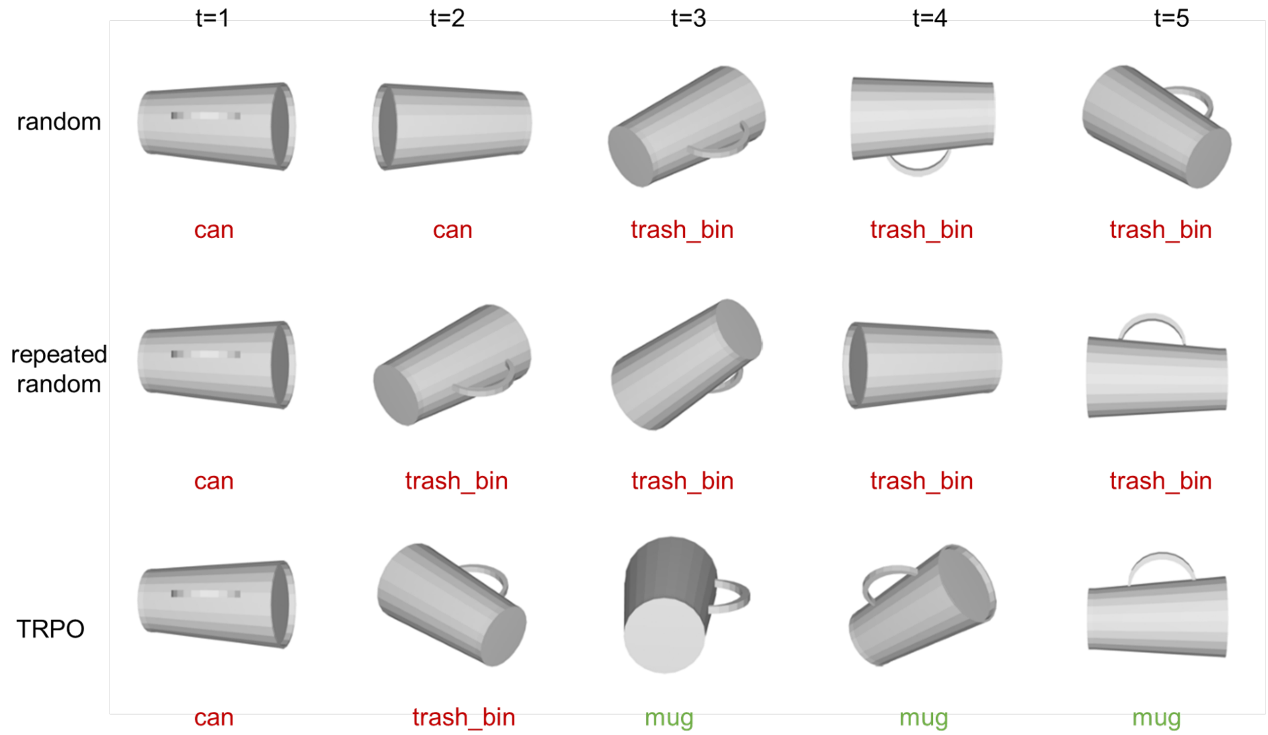
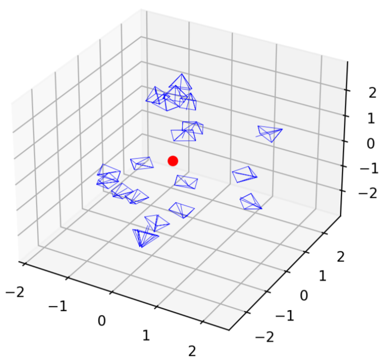
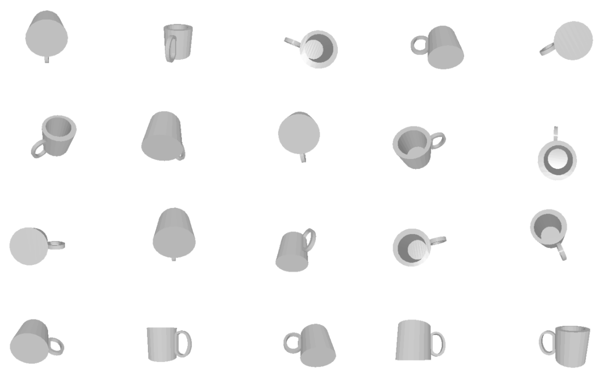


Fig. 4: View selection order for chair_000418 in ShapeNetCore55.



(a) camera viewpoints



(b) rendered images

Fig. 5: One example of camera viewpoints and rendered images of cup_0003 in the continuous view setting.

classifiers with datasets under fixed views. To do this, we randomly generated 20 pairs of coordinates on the sphere and made our free-view dataset based on 3D objects in ModelNet40. Pytorch3D is used as the renderer, the camera is always pointed at the center of the object, and plane reflections are used to render the 3D shape into a 2D image. One example of the camera viewpoint and rendered image are shown in Fig. 5.

T-step ResNet18 with the LSTM layer and the linear layer is trained with the free-view dataset. One single image is used as the input at each step, and the average loss over T is back-propagated. The resulting accuracy of ResNet18 with LSTM is 80.80%. For the view selection module, we chose to use DDPG [5], A2C [6], SAC [2], and PPO [7].

D. Experiments

1) Baselines: Besides Random baseline, Constraint Random is used in the continuous view setting. This baseline indicates a random selection limited by the action space range. Since the value ranges of action space (relative azimuth and relative elevation) in continuous view are $[10°, 72°]$ and $[10°, 36°]$ respectively, to make a fairer comparison, T relative angles are also extracted from the uniform distribution of these two intervals as actions.

2) Quantitative Results: Fig. 7 shows the average accuracy of baselines and our AOR system under continuous view setting.

Although the unrestricted Random baseline has the highest

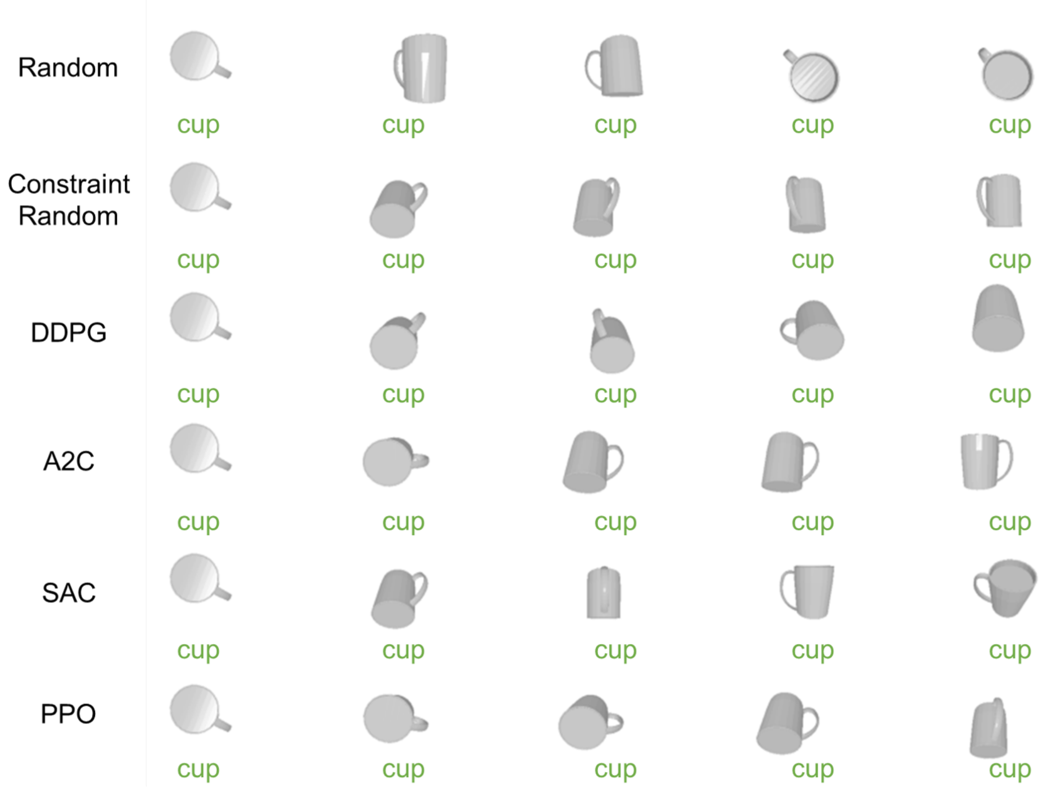


Fig. 6: View selection results of *cup* under continuous views.

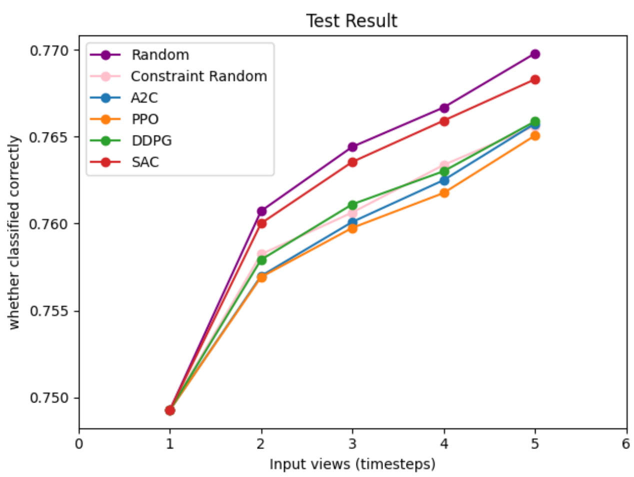


Fig. 7: Accuracy of continuous views with ModelNet40.

accuracy, the performance of A2C, DDPG, and SAC is better than the Constraint Random baseline with the same action space restriction. But the accuracy difference between the worst-performance PPO and Random baseline is only 0.47%, almost no difference. It can be considered that this is because ResNet18 of higher learning capacities overfits all possible views during training, that is, rewards are always provided no matter what action is taken.

As can be seen from Fig. 6, views of Random baseline are more diverse, so it can provide more favorable information for high-ability classifiers, and SAC is the agent with the most varied view, so its effect is closest to unrestricted Random baseline, while other reinforcement learning actions are more conservative. However, if the action space range of AOR agents is not limited, the advantage that continuous view is more in line with the smooth robot vision in reality will be lost, and the difference with discrete view will be less and the research significance may disappear.

So, in the future, we need to think about how to improve accuracy more efficiently using only small view changes. Or, we can consider forming an end-to-end pipeline using differentiable renderers and employing a combination of stochastic gradient descent and REINFORCE [8], as in [4], so that losses can be backpropagated and the gradient of object recognition accuracy in the direction of camera motion can be utilized. Then, jointly training the object recognition and view selection modules can be used to ensure that the newly selected view can improve accuracy.

References

[1] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[2] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1238–1247, 2018.
[5] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
[6] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.
[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
[8] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.