2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

A Double Branch Next-Best-View Network and Novel Robot System for Active Object Reconstruction

Yiheng Han*, Irvin Haozhe Zhan*, Wang Zhao and Yong-Jin Liu†

Abstract— **Next best view (NBV) is a technology that finds the best view sequence for sensor to perform scanning based on partial information, which is the core part for robot active reconstruction. Traditional works are mostly based on the evaluation of candidate views through time-consuming volumetric transformation and ray casting, which heavily limits the applications of NBV. Recent deep learning based NBV methods aim to approximately learn the evaluation function by large-scale training, and improve both the effectiveness and efficiency of NBV. However, these methods force the network to regress the exact groundtruth value of each candidate view, which is much harder than simply ranking all the candidate views. Besides, most previous NBV works assume perfect sensing and perform in simulation environments, lacking real application abilities. In this paper, we propose a novel double branch NBV network, DB-NBV, to utilize the ranking process together with the evaluation process. We further design a real NBV robot and a pipeline to conduct real active reconstruction. Experiments on both simulation and real robot show that our method achieves the best performance and can be applied to real application with high accuracy and speed.**

I. INTRODUCTION

Three-dimensional (3D) object reconstruction is a long-standing important computer vision task. To reconstruct the object, passive vision methods try to extract as much information as possible from fixed perception input, while active vision methods, focus on obtaining more effective input information. With the advancement of robots, sensors and actuators, active vision has gradually draw more attentions, and many research works are proposed to handle active robotic reconstruction task. [1]–[6].

The core of active robotic reconstruction is to find the most effective information collection method, specifically, to find the best views to place the sensor [6]. The challenge of getting an updated sensor view sequence from current states is known as the next best view (NBV). In practice, the views are often limited by many factors such as robot workspace, the field of sensor view, and obstacles. An effective view planner for NBV can help to find optimal view sequences and avoid invalid and redundant views.

This work was partially supported by the Natural Science Foundation of China (61725204) , Tsinghua University Initiative Scientific Research Program (20211080093) and a Grant from Science and Technology Department of Jiangsu Province, China.

Y. Han, I.H. Zhan, W. Zhao and Y-J Liu are with BNRist, MOE-Key Laboratory of Pervasive Computing, Department of Computer Science and Technology, Tsinghua University, Beijing, China. hyh18@mails. zhanhz20@mails. zhao-w19@mails. and liuyongjin@}tsinghua.edu.cn

*Joint first authors

†Corresponding author

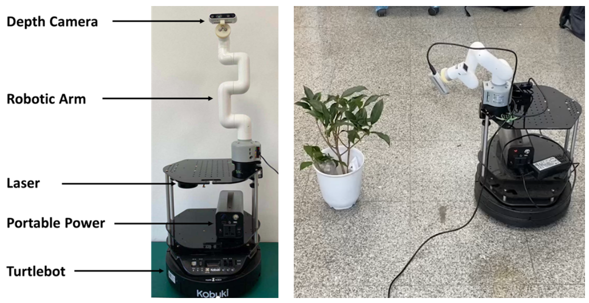


Fig. 1. Our novel designed NBV robot with turtlebot, robotic arm, portable power for powering robotic arm, laser and depth camera as the end effector of robotic arm.

It is hard to find an optimal discrete view sequence in continuous view space, as the search space is often too large to perform feasible optimization. To solve this problem, most of the existing works adopt the generate-and-test pipeline [5], which converts the optimization into the evaluation by defining a number of candidate views, evaluating them and selecting the best one.

Traditionally, the views are generally evaluated by converting the original point cloud from sensors into voxel representation and performing ray casting, which is very time-consuming and the results are not satisfying enough. With the rapid development of deep learning techniques, the NBV methods based on neural networks such as [7], [8] have made great improvements in terms of both effectiveness and efficiency. However, these deep learning based methods, which learn to approximate the evaluation function to score each view, force the network to regress the exact score for each view, according to the groundtruth evaluation scores. This is much harder compared to the pure ranking of all the candidate views, since the network only needs to know the relative rough scores, instead of the precise scores. The ranking process provides essential and simple information for the next-best-view, but not well studied in the learning based NBV methods.

In our paper, we propose a novel double branch NBV network, called DB-NBV, which elegantly combines view ranking task with view evaluation to better decide the NBV. Furthermore, we design an NBV robot with an Automated Guided Vehicle (AGV), robotic arm, laser, portable power for powering the robotic arm, and depth camera as the end effector of the robotic arm. A constrained NBV robot planner trained by our DB-NBV on ShapeNet dataset [9] is introduced to conduct the reconstruction of 3D objects. We validate the DB-NBV and our NBV system both in simulation and real world. Experiments on ShapeNet and selected complex object models from Stanford 3D Scanning Repository[1] and MIT CSAIL Textured Models Database[2] show that our DB-NBV achieves state-of-the-art performance compared to strong baselines. Real-world examples prove that DB-NBV can be applied to our real-time NBV robot and maintain the high performance. The main contributions of our work are as follows:

- A novel DB-NBV method that effectively combines the ranking and evaluation process.
- A real NBV robot with our trained DB-NBV planner, which achieves real-time reconstruction.
- Extensive experiments show that our DB-NBV achieves the best reconstruction among existing methods.

II. Related Work

A. Next best view

NBV has been one of the key points in the field of active vision for many years [3]. We distinguish it between synthesis method, which is designed for a specific task, fixed constraints and repetitive work [6], and generate-and-test method, which generate many candidate views and select by evaluating them [4], [5], [10]. Synthesis method has a lower cost, but it relies too much on subjective experience and has almost no generalization. When the environment and goals change, the synthesis method will be totally invalid. In comparison, the generate-and-test method takes into account both time and efficiency, and has high stability. Therefore, most works on NBV choose the generate-and-test method, including ours.

B. Data Representation

The data representation used to store, express, and calculate the spatial information and position will fundamentally change the algorithm of NBV. At first, the 3D model of NBV is represented by triangular mesh [11], [12]. However, on the one hand, it has too much redundant information, which leads to high calculation and storage costs. On the other hand, its evaluation is not intuitive and clear. Later, voxel is widely used in NBV due to simpler and more intuitive representations and more diversified evaluation methods. Connolly [4] evaluate the candidate view via the information gain in the voxel space. Some other works use ray casting to evaluate the sensor views through simulation [13]–[18]. Object probability [19] and saliency detection by point cloud segmentation [20] are also proposed to score the candidate views. Although the efficiency of voxel is greatly improved compared to mesh, the conversion from the point cloud collected by depth camera to voxel and ray casting are still very time-consuming, which leads to the real-time performance cannot be guaranteed. Recently, with the proposal of various point cloud feature extraction methods [10], [21] and the widespread of its applications [22]–[26], point cloud began to be used directly in NBV [27]. Our DB-NBV also falls in the point cloud representation.

[1]graphics.stanford.edu/data/3Dscanrep/
[2]people.csail.mit.edu/tmertens/textransfer/data/

C. Deep learning for NBV

With the expansion of the scope of applications and the establishment of databases [9], deep learning begins to be used in NBV. Wu et al. [28] design a 3D-shape net to complete a 3D model and find the uncertain area to guide the next best view. Another work [29] takes the same plan and applies it to a multi-view scenario with plant phenotyping. Johns et al. [30] train a convolutional neural network to find the next best view by both depth and greyscale image sequence. In the work [31], a 3D convolutional neural network is proposed to learn utility functions and evaluate the views for a volumetric scene. Mendoza et al. [7] use voxel data to train their NBV-net and find the next best view to reconstructing 3D objects. Collander et al. [32] combine NBV with reinforcement learning by topological features from point cloud. Zeng et al. [8] directly uses point cloud as input for their PC-NBV network and has made a great improvement in performance.

III. Problem Statement

Limited by the workspace of the robotic arm and the relative position of the robot and reconstructed object as shown in Figure. 1, we control the NBV robot to move on a fixed-size circle around the reconstructed object to include as many candidate views as possible. During making a circle, we will iterate the view calculation and sensor acquisition n times at even intervals, deciding a serials of views $V_{plan} = \{v_i | i = 0, 1, 2, ..., n\} \subset R^3 \times SO(3)$ based on the input partial point clouds $X = \{x_0, x_1, ..., x_m\} \subset R^3$ captured by depth camera. Existing work has proved that finding the best V_{plan} with prior knowledge is an NP-C problem [27].

IV. DB-NBV

To the best of the authors' knowledge, all the existing work about NBV is to accurately evaluate views and pay no attention to keep the rank of views. But obviously, to select the "best", the rank of the views may be a more intuitive and important metric. Therefore, we propose a novel double branch NBV network (DB-NBV) that includes both rank branch and evaluation branch. In this section, we first introduce the generation of training data with our NBV robot constraints. Then a novel point-wise point cloud input fused with view state and point weight is also described in this part. Finally, we split our network architecture into a rank branch and evaluation branch to explain in detail.

A. Data generation with NBV robot constraints

1) NBV robot constraints: As shown in Figure. 1, our NBV robot can only reach part of the views at a fixed position. To reconstruct objects as efficiently and completely as possible, we control the NBV robot to move on a fixed-size circle around the reconstructed object and iteratively find the best view n times at even intervals. An NBV process is finished when the NBV robot completes a circle, evenly collects n depth images, and returns to the origin. We first uniformly sampling views on a sphere centered on the object. Then according to the limitations of the robot arm workspace, the size of the circle, and the height of the NBV robot, each candidate views selected will be limited to the workspace reachable by the robotic arm. The constraints are used in the generation of the training data.

2) Data generation: What the NBV network needs to achieve is the ability to accurately evaluate views and select the best. Specifically, for a point cloud input and a set of views, NBV network should accurately evaluate their information gain and find the best view. In theory, any accurate evaluation can be used as ground truth to help neural networks better achieve this function. We can easily simulate the reconstruction process on synthetic 3D datasets likes ShapeNet[19] and get an accurate evaluation. However, traversing all the viewpoints and selection order is extremely time-consuming and space-consuming. A random generation will have poor effect due to the low sampling rate. As a trade-off, we use the greedy method and constrained view state to generate training data based on the complete model as prior knowledge. The training data generated in this way can avoid samples with low efficiency and cover most of the efficient samples for training. The detail is shown in Algorithm. 1.

To generate data by simulation, we need the mesh model O to get the point cloud from selected view v^i by operation P, the complete point cloud P_c to calculate the coverage rate as the label, and the constrained candidate set $A = \{A_i | i = 1, 2, ..., n\}$ to divide the entire candidate view set V into the iteration number n sets with overlap. The same size of sensor reachable space makes each element of A contain fixed m candidate views. The overlap occurs because the reachable space of the sensor is larger than the interval angle of uniform collection on the fixed circle.

To facilitate constrained view selection, we define the v_{state} as follow:

$$v_{state} = \begin{cases} 1, & v \in V_{selected} \\ 2, & v \in A_{iter} \ and \ v \notin V_{selected} \\ 0, & otherwise \end{cases} \tag{1}$$

Where $V_{selected}$ is the set of selected sensor views. For each view selection, only the views with $v_{state} = 2$ can be selected.

We follow the definition of coverage rate and newly added point cloud in [8]. For any partial point cloud of the model obtained from a selected view, we can use this equation to evaluate its completeness and label the view.

$$C(P, v_s) = \begin{cases} \frac{1}{|P_c|} \sum\limits_{p_c \in P_c} U(\min\limits_{p \in P} ||p - p_c||_2 - \epsilon), \ v_s = 2 \\ 0, \ otherwise \end{cases} \tag{2}$$

Where U is the Heaviside step function, and ϵ is a distance threshold. Besides. We find that for constrained NBV, the importance of each point in the point cloud is different as the constraint changes. For a simple example, when our NBV robot is on the front of an object, the point cloud on the back of the object has little effect on NBV calculations. Inspired by this, we design a point-wise weight feature W as follow:

$$W = \frac{\mathbf{p}_{xy} \cdot \mathbf{p}_{robot}}{2} \tag{3}$$

After the point is projected onto the circular surface, $\mathbf{p}_{xy}$ is the vector from the point to the centroid of the object. $\mathbf{p}_{robot}$ is the vector from the current position of the NBV robot to the centroid of the object.

Algorithm 1: NBV training data preparation

```
Input: $O$, $P_c$, $V_c$, $max_{iter}$, $A$.
1  foreach $v \in V_c$ do
2      $v_{state} \leftarrow 0$
3  end
4  $W \leftarrow \varnothing$
5  $iter \leftarrow 0$
6  $P_{part} \leftarrow \varnothing$
7  $i \leftarrow Random(1:m)$
8  while $iter < max_{iter}$ do
9      $v \leftarrow A_{iter}$
10     $v^i_{state} \leftarrow 1$
11     $P_{part}.append(P(v^i, O))$
12     $max \leftarrow 0$
13     foreach $j \leftarrow 1:m$ do
14         if $C(P^j_{new}, v^j_{state}) > max$ then
15             $max \leftarrow C(P^j_{new}, v^j_{state})$
16             $i \leftarrow j$
17         end
18     end
19     $\mathbf{p}_{robot} \leftarrow FindCenterPointXY(A_{iter})$
20     foreach $\mathbf{p} \in P_{part}$ do
21         $\mathbf{p}_{xy} \leftarrow CoordXY(\mathbf{p})$
22         $W.append((\mathbf{p}_{xy} \cdot \mathbf{p}_{robot} + 1)/2)$
23     end
24     $Save(P_{part}, V_{state}, C(P_{new}, v_{state}), W)$
25     $iter++$
26 end
```

B. Network Architectures

1) Rank branch: Since the raw point cloud is hard to use as input directly, we followed the existing feature extraction module [26] and add additional processing to our novel W feature. In the extraction block 1, after the feature extraction unit, we can get the point the point-wise feature F_0. Then, we get the reinforced feature F_1 by concatenating the F_0, duplicated max-pooling feature G_0, and point weight W. Because we concatenate our novel point weight W to differentiate the effectiveness of each point, we use an additional self-attention module [25] to remap the attention according to W and get the weight mapped feature F_2'. After concatenates with the duplicated V_{state}, we input the F_3' to extraction block 2. The self-attention module in extraction block 2 aims to get the view state mapped feature F_4. Finally, followed by the shared MLP, max pooling, and MLP, rank scores R are obtained.

Inspired by RankNet [33] and probabilistic cost function, the loss function of our rank branch L_R is set as follow:

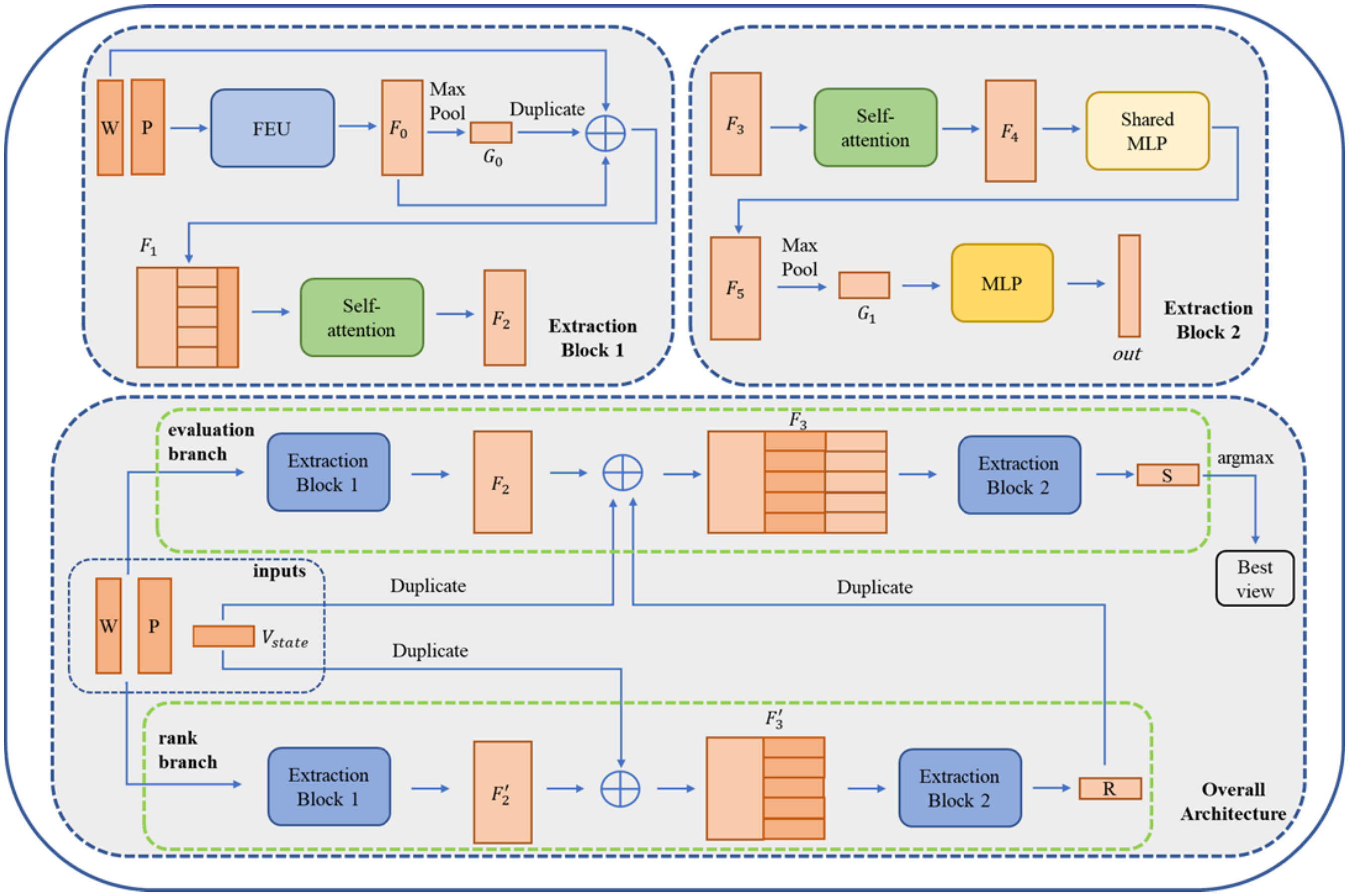


Fig. 2. Network Architectures of our DB-NBV.

$$P_{ij} = \frac{e^{R(i)-R(j)}}{1+e^{R(i)-R(j)}} \tag{4}$$

Where i,j are the indexes of view.

$$C_{ij} = -\bar{P_{ij}} \log P_{ij} - (1-\bar{P_{ij}}) \log(1-P_{ij}) \tag{5}$$

$\bar{P_{ij}}$ is calculated by the ground truth R_{gt} and P_{ij} is calculated by R.

$$L_R = \sum_{i,j=1}^{n} C_{ij} \tag{6}$$

n is the number of candidate views. The score of rank branch will be fused to the evaluation branch.

2) Evaluation branch: The feature extraction blocks of the evaluation branch are the same as that of the rank branch. The score of the rank branch will be fused with F_2 and obtain the F_3 with an additional rank feature. Mean squared error (MSE) is applied as the loss function L_E, which calculated by S and S_{gt}. S is the coverage rate defined in Equation. 2. The index of the max value in S will be the best view selected by the network.

V. Experiments

In this section, we train our DB-NBV with the data generated by our Algorithm. 1. on a random selection subset of ShapeNet [9]. We use objects of the same type as the training set to generate Familiar objects testing dataset, different types of objects to generate Unfamiliar objects testing dataset, and find six complex models to generate Complex objects testing dataset. Our DB-NBV compares with volumetric methods AreaFactor [15], voxel-based deep learning method NBV-net [7] and state-of-the-art point-cloud-based deep learning method PC-NBV [8] on all test sets.

A. Training details

PC information. All experiments are deployed on a PC with Intel E5-2640-v4 CPU and NVIDIA RTX2080Ti GPU.

Dataset details. ShapeNet is a richly annotated three-dimensional CAD model of an object, a large-scale shape storage library, and has indexed more than 3 million models, of which 220,000 models are divided into 3135 categories. It is so large that we have to choose some representative models from it. Similar to PC-NBV, we use a randomly selected subset of ShapeNet as training dataset. we randomly pick 4,000 models from the 8 categories (vessel, table, sofa, lamp, chair, car, cabinet and airplane) as training data, 400 models as validation data, 800 models as Familiar objects testing dataset and 800 models from another unseen 16 categories (bathtub, bed, bicycle, camera, clock, display, faucet, helmet, piano, pistol, printer, rifle, rocket, stove, tower, train) as Unfamiliar objects testing dataset. The categories are shown in Figure. 3. We also use another six complex models

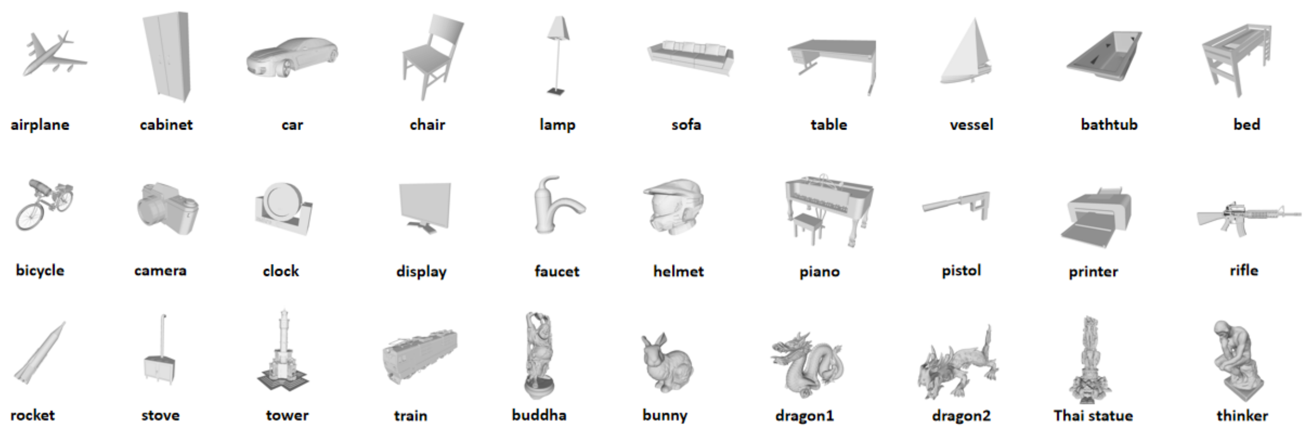


Fig. 3. All the test models from three test dataset. The first 8 models similar to training models are from the Familiar objects testing dataset, the next 16 models are from the Unfamiliar objects testing dataset and the last 6 models with rich surface details are from Complex object testing dataset.

TABLE I

Comparison on Unfamiliar objects testing dataset. Coverage results after 10 iterations are reported on each category. DB-NBV achieves the best performance on all categories.

	bathtub	bed	bicycle	camera	clock	display	faucet	helmet	piano	pistol	printer	rifle	rocket	stove	tower	train	Average
Random	0.695	0.521	0.788	0.573	0.728	0.814	0.643	0.489	0.556	0.485	0.552	0.614	0.625	0.530	0.574	0.515	0.606
Area Factor [15]	0.853	0.763	0.963	0.816	0.906	0.918	0.900	0.794	0.827	0.770	0.832	0.898	0.875	0.770	0.838	0.761	0.842
NBV-Net [7]	0.875	0.721	0.948	0.789	0.880	0.913	0.857	0.748	0.762	0.746	0.772	0.888	0.888	0.735	0.781	0.789	0.818
PC-NBV [8]	0.868	0.780	0.960	0.833	0.901	0.925	0.887	0.804	0.857	0.771	0.824	0.901	0.909	0.765	0.822	0.823	0.852
DB-NBV	**0.902**	**0.788**	**0.973**	**0.849**	**0.910**	**0.932**	**0.907**	**0.810**	**0.860**	**0.788**	**0.857**	**0.913**	**0.921**	**0.779**	**0.839**	**0.832**	**0.866**

TABLE II

Comparison on Familiar objects testing dataset. Coverage results after 10 iterations are reported on each category. DB-NBV achieves the best performance on all categories.

	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Average
Random	0.648	0.524	0.446	0.662	0.702	0.526	0.653	0.607	0.596
AF [15]	0.899	0.719	0.710	0.890	0.911	0.749	0.878	0.836	0.824
NBV-Net [7]	0.895	0.694	0.684	0.872	0.900	0.746	0.882	0.815	0.811
PC-NBV [8]	0.913	0.724	0.745	0.902	0.913	0.772	0.922	0.843	0.842
DB-NBV	**0.924**	**0.732**	**0.760**	**0.915**	**0.926**	**0.772**	**0.922**	**0.865**	**0.852**

TABLE III

Comparison on Complex objects testing dataset. Coverage results after 10 iterations are reported on each category. DB-NBV achieves the best performance on all categories.

	buddha	bunny	dragon1	dragon2	Thai statue	thinker	Average
Random	0.648	0.615	0.564	0.638	0.631	0.608	0.617
AF [15]	0.937	0.982	0.905	0.930	0.988	0.742	0.914
NBV-Net [7]	0.883	0.842	0.874	0.944	0.886	0.916	0.891
PC-NBV [8]	0.901	0.998	0.966	0.979	0.986	0.967	0.966
DB-NBV	**0.949**	**0.999**	**0.983**	**0.995**	**0.992**	**0.982**	**0.983**

(thinker, Thai statue, dragon1, dragon2, bunny, buddha) to generate Complex objects testing dataset.

Parameter details. For each P_c in Equation. 2, we sample 16,384 points from object model O and set the distance threshold ϵ to 0.00707m. The candidate views are generated uniformly in the reachable space for robotic arm with a constraint on z axis: $z_{axis} \in [-30°, 60°]$. The candidate view number $m = 240$ and the $max_{iter} = 10$. m usually takes an integer multiple of max_{iter}, otherwise for the NBV robot, the candidate view space will lose its rotation symmetry. Batch size should be higher than 32 and the learning rate is set to 0.0001. To ensure real-time, point clouds are down-sampled to 512 for training and 1024 for testing. The learning rate is set to decrease by 0.7 every 50,000 iterations. L_2 loss is used by our network with 0.0005 for rank branch and 0.0001 for evaluation branch. To make the NBV robot traverse all candidate views, it must complete ten iterations for each reconstruction. Therefore, whether it is training or testing, we have completed 10 iterations without premature termination.

B. Comparison on testing dataset

Our DB-NBV compares with random sample, volumetric methods AreaFactor [15], voxel-based deep learning method NBV-net [7] and state-of-the-art point-cloud-based deep learning method PC-NBV [8] on three testing datasets defined in dataset detail. Since these models are used on our NBV robot that collects ten times uniformly each reconstruction, unlike other work [8], [15] that focuses on AUC, we pay more attention to the reconstruction coverage after a complete acquisition. Therefore, we use the 10^{th} round coverage rate to measure the performance. The results of three testing datasets are shown in Table. I, Table. II and Table. III.

According to the result, our DB-NBV has the best performance in all categories. All test categories in Table. I are randomly selected. Our method has not only the best overall effect but also the best in all individual categories. We believe that this is because the increased forecast uncertainty of our double branch network and constrained datasets make the choice of NBV more inclined to choose a globally optimal strategy to achieve the best results. To further confirm, we draw the AUC curve of the entire Familiar objects testing dataset and Unfamiliar objects testing dataset as Figure. 4.

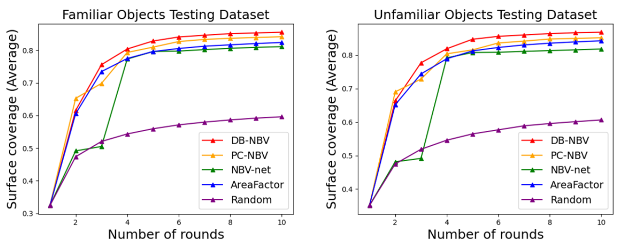


Fig. 4. AUC curve of Familiar objects testing dataset and Unfamiliar objects testing dataset.

In the first two rounds of the two testing dataset, the performance of our DB-NBV is lower than PC-NBV and volumetric methods. However, from the third round to the end, our method has always been better than others. Due to the low interpretability of neural networks, it is hard to directly establish a mathematical model to analyze the reasons. But obviously, our dual-branch NBV network makes our NBV model more inclined to collect points with higher global value and can find the part of point clouds that is difficult to collect.

Since our network structure and constraints are more complex than PC-NBV and our NBV robot has to achieve real-time, we test the inference time of all NBV methods in our paper.

TABLE IV

COMPARISON OF INFERENCE TIME FOR EACH NBV PREDICTION.

	AF	NBV-net	PC-NBV	Ours
Time(s)	11.66	0.913	0.063	0.098

As shown in Table. IV, due to our complex network structure and constraints, DB-NBV is slightly slower than PC-NBV, but still much higher than traditional methods, which can ensure real-time performance for our NBV robot.

C. Our NBV robot

The overall design of our NBV robot is shown in Figure. 1. To achieve efficient collection and full coverage of candidate views, the NBV robot will make a circle around the reconstruction object clockwise and collect point clouds data ten times evenly during the movement. Laser is used to find the initial position for the NBV robot by keeping a fixed distance to the reconstructed object. The robotic arm is placed on the left side of the turtlebot to cover more candidate views.

We sample candidate views uniformly in workspace, but the reachable views of the robotic arm cannot be directly obtained. To get the constrained candidate set A defined in Section.IV.A, we plan the motion for each candidate view and save the trajectory for reachable views. As described in Section. V.A, the candidate view space has rotational symmetry about the z axis by every 36 degrees. Therefore, we only need to calculate the subset A_1 for initial position of the NBV robot, and the entire set A can be obtained by simply rotating A_1. According to motion planning, 40 candidate views can be reached with the candidate view number $m = 240$ for the initial position.

We choose plants as the reconstruction objects, because the surface and occlusion of the plants are very complicated, which makes it difficult to reconstruct. A snapshot is shown in Figure. 5 and the entire NBV demo can be found in our video.

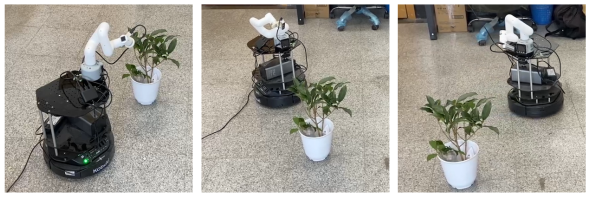
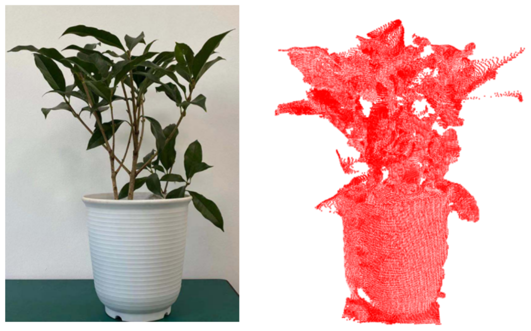


Fig. 5. A snapshot for our NBV process.

An example reconstruction result is given in Figure. 6. We collect point clouds from ten candidate viewpoints and simply stitch them together according to the camera pose. The error between the reconstruction result and the reconstructed object comes from the 3D perspective and the errors of the camera pose which is caused by the accuracy of the robotic arm and turtlebot. In future work, we will try to change the robot with higher accuracy or use some additional registration operations to further solve this problem.



(a) Reconstruction object (b) Reconstruction result

Fig. 6. An example reconsturction result for our NBV robot.

VI. Conclusions

In this paper, we design a novel NBV robot and propose a point-clouds-based double branch NBV network (DB-NBV) that combines the ranking and evaluation process. Simulation experiment results show that our DB-NBV outperforms all the existing NBV methods for both trained and untrained models, and has extremely high stability and global optimization performance. Real-world reconstruction experiments verify the ability and effectiveness of our method for real applications.

REFERENCES

[1] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International journal of computer vision*, vol. 1, no. 4, pp. 333–356, 1988.

[2] J. Aloimonos, "Purposive and qualitative active vision," in *[1990] Proceedings. 10th International Conference on Pattern Recognition*, vol. 1. IEEE, 1990, pp. 346–360.

[3] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.

[4] C. Connolly, "The determination of next best views," in *Proceedings. 1985 IEEE international conference on robotics and automation*, vol. 2. IEEE, 1985, pp. 432–435.

[5] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai, "A survey of sensor planning in computer vision," *IEEE transactions on Robotics and Automation*, vol. 11, no. 1, pp. 86–104, 1995.

[6] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Computing Surveys (CSUR)*, vol. 35, no. 1, pp. 64–96, 2003.

[7] M. Mendoza, J. I. Vasquez-Gomez, H. Taud, L. E. Sucar, and C. Reta, "Supervised learning of the next-best-view for 3d object reconstruction," *Pattern Recognition Letters*, vol. 133, pp. 224–231, 2020.

[8] R. Zeng, W. Zhao, and Y.-J. Liu, "Pc-nbv: A point cloud based deep network for efficient next best view planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7050–7057.

[9] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[11] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.

[12] S. Chen and Y. Li, "Vision sensor planning for 3-d model acquisition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 5, pp. 894–904, 2005.

[13] N. A. Massios, R. B. Fisher, *et al.*, *A best next view selection algorithm incorporating a quality criterion*. Department of Artificial Intelligence, University of Edinburgh, 1998, vol. 2.

[14] C. Potthast and G. S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 148–164, 2014.

[15] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, "Volumetric next-best-view planning for 3d object reconstruction with positioning error," *International Journal of Advanced Robotic Systems*, vol. 11, no. 10, p. 159, 2014.

[16] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, "View/state planning for three-dimensional object reconstruction under uncertainty," *Autonomous Robots*, vol. 41, no. 1, pp. 89–109, 2017.

[17] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects," *Journal of Real-Time Image Processing*, vol. 10, no. 4, pp. 611–631, 2015.

[18] F. Bissmarck, M. Svensson, and G. Tolt, "Efficient algorithms for next best view evaluation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5876–5883.

[19] J. Daudelin and M. Campbell, "An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-d objects," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1540–1547, 2017.

[20] R. Monica and J. Aleotti, "Contour-based next-best view planning from point cloud segmentation of unknown objects," *Autonomous Robots*, vol. 42, no. 2, pp. 443–458, 2018.

[21] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[22] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 605–613.

[23] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 728–737.

[24] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7203–7212.

[25] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[26] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3d point set upsampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5958–5967.

[27] G. H. Tarbox and S. N. Gottschlich, "Planning for complete sensor coverage in inspection," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 84–111, 1995.

[28] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[29] C. Wu, R. Zeng, J. Pan, C. C. Wang, and Y.-J. Liu, "Plant phenotyping by deep-learning-based planner for multi-robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3113–3120, 2019.

[30] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3813–3822.

[31] B. Hepp, D. Dey, S. N. Sinha, A. Kapoor, N. Joshi, and O. Hilliges, "Learn-to-score: Efficient 3d scene exploration by predicting view utility," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 437–452.

[32] C. Collander, W. J. Beksi, and M. Huber, "Learning the next best view for 3d point clouds via topological features," *arXiv preprint arXiv:2103.02789*, 2021.

[33] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 89–96.