ScanBot: Autonomous Reconstruction via Deep Reinforcement Learning

HEZHI CAO[*] and XI XIA[*], University of Science and Technology of China, China
GUAN WU, University of Science and Technology of China, China
RUIZHEN HU[†], Shenzhen University, China
LIGANG LIU, University of Science and Technology of China, China

CCS Concepts: • **Computing methodologies**; • **Shape analysis**;

Additional Key Words and Phrases: Indoor scene reconstruction, autonomous reconstruction, hierarchical reinforcement learning, auxiliary learning tasks

ACM Reference Format:
Hezhi Cao, Xi Xia, Guan Wu, Ruizhen Hu, and Ligang Liu. 2018. ScanBot: Autonomous Reconstruction via Deep Reinforcement Learning. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 METHOD DETAILS

1.1 Basic components

Algorithm. Starting from the initial location p^r, the robot tries to explore and scan the unknown environment with quality reconstruction $\mathcal{S}$. Then, a ROI with insufficient exploration or incomplete objects is generated by the global scanning policy $\Pi^{(g)}$ after taking in the 2D quality map $\mathcal{M}$ built from $\mathcal{S}$. The ROI is first represented by a goal point $a^{(g)}$, accessed through the path $\mathcal{P}$ generated by D* lite and ends up to be p^r, where the robot actually arrived. All unscanned objects in the designated ROI are progressively selected with group of k and completed by scanning the viewpoint $a^{(l)}$ dedicated by the masked local policy $\Pi^{(l)}$. The local policy takes the current voxelized ROI region $\mathcal{V}^r$ and selected objects $\mathcal{V}^o$ as input to generate the viewpoint for completion until it is content with the reconstruction and chooses STOP action, or no feasible candidate viewpoints left. The robot iterates the above steps until the whole scene has been completely explored and scanned. The entire autoscanning process is shown in Algorithm 1, where $\mathcal{B}_u$ and $\mathcal{B}_s$ denotes unscanned objects bank in designated ROI and global scanned objects bank, respectively.

[*]Both authors contributed equally to this research.
[†]Corresponding author: Ruizhen Hu (ruizhen.hu@gmail.com)

Authors' addresses: Hezhi Cao; Xi Xia, University of Science and Technology of China, Hefei, Anhui, China; Guan Wu, University of Science and Technology of China, Hefei, China; Ruizhen Hu, Shenzhen University, Shenzhen, China; Ligang Liu, University of Science and Technology of China, Hefei, China.

ALGORITHM 1: DRL-based autoscanning for scene reconstruction

```
Input: Initial pose of the robot $p^r$
Output: Reconstructed scene $\mathcal{S}$
$\mathcal{S} \leftarrow$ InitiallyScan($p^r$);
$\mathcal{B}_s = \emptyset$;
repeat                                  /* Global scanning loop */
    $\mathcal{M} \leftarrow$ ConstructQualityMap($\mathcal{S}$);
    $a^{(g)} \leftarrow \Pi^{(g)}(\mathcal{M})$;
    $\mathcal{P} \leftarrow$ PlanPath($a^{(g)}$);
    $p^r \leftarrow$ FollowPath($\mathcal{P}$);
    $\mathcal{S} \leftarrow$ ScanAlongPath($\mathcal{P}$);
    $\mathcal{B}_u \leftarrow$ ObtainUnscannedObjectsInROI($\mathcal{S}$, $p^r$, $\mathcal{B}_s$);
    repeat                              /* Local scanning loop */
        $idx \leftarrow$ SelectObjects($\mathcal{B}_u$, $k$);
        $\mathcal{B}_u, \mathcal{B}_s \leftarrow$ UpdateObjectsBank($idx$);
        $\mathcal{V}^o, \mathcal{V}^r \leftarrow$ ConstructVoxelGrid($\mathcal{S}$, $idx$);
        repeat                          /* Object scanning loop */
            $a^{(l)} \leftarrow \Pi^{(l)}(\mathcal{V}^o, \mathcal{V}^r)$;
            $\mathcal{S} \leftarrow$ ScanNBV($a^{(l)}$);
            $\mathcal{B}_u \leftarrow$ RecordNewObjectsInROI($\mathcal{S}$);
            $\mathcal{V}^o, \mathcal{V}^r \leftarrow$ UpdateVoxelGrid($\mathcal{S}$);
        until No viewpoint left or $a^{(l)}$ = STOP;
    until $\mathcal{B}_u == \emptyset$;
until Time runs out;
```

Semantic SLAM framework. After the robot enters an unknown indoor environment, it begins acquiring a stream of raw RGB-D images on the fly as input, which are processed by an underlying semantic dense reconstruction SLAM framework with frame-to-frame tracking, segmentation, and fusion operations [Runz et al. 2018]. The SLAM framework estimates camera pose at current step and updates the current scene reconstruction $\mathcal{S}$ with newly acquired points and instance labels in real-time. Note that the MaskRCNN module to obtain semantic labels is pretrained on the MS-COCO dataset and fine-tuned by the samples collected from Gibson and MP3D, considering the discrepancy between the classes in MS-COCO and our used datasets. Apart from semantic masks from the MaskRCNN module, geometric boundaries produced by the geometric segmentation are also incorporated to acquire a more accurate instance segmentation of objects. After obtaining the labeled points, the robot maintains an object-centric dense map $\mathcal{S} = \{s_k = (\mathbf{x}_k, l_k)\}_k$ with each 3D point associated with space position $\mathbf{x}_k$ and instance label l_k. Moreover, we would like to highlight that how to obtain accurate camera poses is not the focus of our proposed method but the SLAM module. Our method aims at planning the path of scanner to complete the partial objects based on current reconstruction S.

The DRL framework. We utilize the Decentralized Distributed Proximal Policy Optimization (DD-PPO) [Wijmans et al. 2019], which belongs to the actor-critic structure, as our main DRL framework. In an actor-critic based approach, an actor and a critic module are updated iteratively. Given the current environment state s_t, the actor serves as a stochastic policy $\pi(a_t|s_t)$ and is trained to output either the mean $\mu(s_t)$ and standard deviation $\sigma(s_t)$ of a Gaussian distribution for continuous actions or the probability of each discrete action. At the same time, the critic is trained to predict the future cumulative discounted reward $V(s_t)$ (i.e. state-value function) and can be used to estimate the relative advantage of an action via Generalized Advantage Estimation (GAE) [Schulman et al. 2015]. DD-PPO scales the original PPO method very well in distributed and decentralized settings and is suitable for training in resource-intensive simulations such as photorealistic 3D environments.

1.2 2D quality map

The construction of exploration, obstacle, and semantic channels in 2D quality map is detailed as follows:

Exploration channel. The exploration channel is used to reflect where the agent has visited. Specifically, without removing the floor plane, current acquired 3D point cloud $\mathcal{S}$ is directly projected to form a 0-1 grid (i.e. elements of value 0 denote unseen region and value 1 denote explored locations).

Obstacle channel. The obstacle channel is constructed by counting all remaining 3D points belonging to each grid cell and normalized by a non-linear transformation $clip((c/c_{\max})^2, 0, 1)$, where $c_{\max}$ is a threshold considered as fully non-traversable. Hence, the values in this channel are appropriate approximations of the degree to which the robot cannot safely pass through these cells.

Semantic channels. We select C_s functional object categories to represent the semantic channels of the global map without overwhelming the agent with negligible semantic signals (please refer to Section 2 for selected categories). Any remained classes are absorbed into an "Others" channel, including objects on the table or ornaments. Each element in these channels simply represents the occupancy of the corresponding category.

Trajectory channel. The trajectory channel memorizes the agent's positions of the last n_{traj} steps with a linearly decaying scheme to establish a time-varying trajectory.

1.3 Off-Policy Correction

The fact that zeroing out the probabilities of infeasible actions turns the local policy learning into an off-policy update. In other words, the agent now behaves according to the masked policy $\pi^{\text{mask}}(a|s)$ (i.e., behavioral policy) but the PPO algorithm uses the collection of rollouts to update the original unmasked policy $\pi^{(l)}(a|s)$ (i.e., target policy). This distribution inconsistency breaks the hypothesis of the policy gradient method to which the PPO belongs and can lead to instability of training, and hence must be fixed. To be more specific, we incorporate the V-trace algorithm [Espeholt et al. 2018]

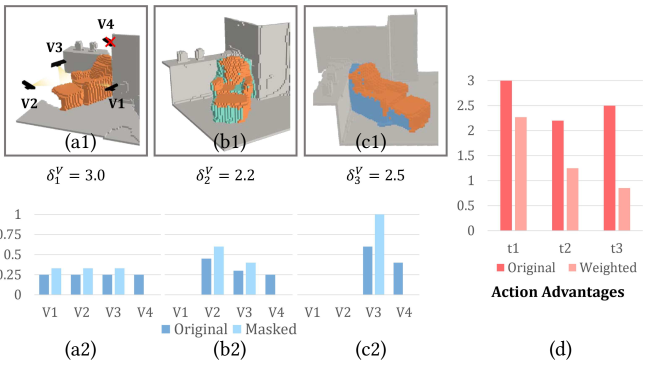

Fig. 1. An example of the off-policy correction. At each step, the agent can choose one action from V1-V4 (a1). The V4 is infeasible due to the obstruction from the wall. The agent then sequentially selects V1, V2, and V3 with the newly reconstructed voxels highlighted in different colors (a1-c1). For each step, The action probabilities before and after applying the feasibility mask are juxtaposed in (a2)-(c2). In (d), we show the original and importance weighted action advantages to illuminate the fact that the actual advantages may be much lower due to the increased frequencies of unmasked actions.

to replace the GAE-based [Schulman et al. 2015] return

$$v_t = V(s_t) + \sum_{l=0}^{n-1} (\gamma\lambda)^l \delta_{t+l}^V \tag{1}$$

with importance weighted one

$$v_t = V(s_t) + \sum_{l=0}^{n-1} (\gamma\lambda)^l \Big(\prod_{i=0}^{l-1} c_{t+i}\Big) \hat{\delta}_{t+l}^V \tag{2}$$

where $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD error for V and $\hat{\delta}_t^V = \rho_t \delta_t^V$ is the weighted TD error with $\rho_t = min(\bar{\rho}, \frac{\pi^{(l)}(a_t|s_t)}{\pi^{\text{mask}}(a_t|s_t)})$, and $c_i = min(\bar{c}, \frac{\pi^{(l)}(a_i|s_i)}{\pi^{\text{mask}}(a_i|s_i)})$ is another truncated importance sampling weights (Figure 1). In plain words, c_i and ρ_t reduce the importance of the rewards obtained by initially less frequent actions since some infeasible actions have been removed ($\frac{\pi^{(l)}(a_i|s_i)}{\pi^{\text{mask}}(a_i|s_i)} \leq 1$). By the same token, we also replace policy gradient ratio $r_t = \frac{\pi^{(l)}(a_t|s_t)}{\pi^{(l)}_{\text{old}}(a_t|s_t)}$ with $\frac{\pi^{(l)}(a_t|s_t)}{\pi^{\text{mask}}_{\text{old}}(a_t|s_t)}$ to correct the policy gradient estimation. The reader is encouraged to refer to [Espeholt et al. 2018] for more details.

2 EXPERIMENT DETAILS

Habitat tasks. The global exploratory scanning task is devised to train our global scanning policy, where the agent is randomly placed in a room and asked to completely explore and scan the simulated 3D environment with depth and semantic sensors. For this task, we only keep our 2D quality map as the agent's internal spatial representation and integrate single-frame point clouds directly into it. This simplification can reduce memory consumption (by not keeping the whole 3D reconstruction result) when training, yet generalize well in our experiments.

On the contrary, the local object reconstruction task mimics the scenario when the agent enters the ROI and tries to sense it. To this end, we randomly generate a pair of start and end points for each episode. The end point is constrained to be within a small radius around an object, while the start point is not. At the start of each episode, the agent is steered to the end point from the start point and begins scanning the selected objects in detail with the initially acquired point cloud. Unlike the global task, we keep a full-blown 3D reconstructed point cloud in the local one to provide accurate voxel grids to the agent.

Parameter setting. When constructing a 2D quality map, we evenly divide 2π into $C_q = 8$ intervals, hence yielding 8 quality-related channels, and compute quality score with $d_{\text{best}} = 1.5$ and $d_{\max} = 3.0$. In addition, we choose a set of $C_s = 10$ functional object categories to form the semantic channels: *chair*, *table*, *cabinet*, *sofa*, *bed*, *sink*, *stairs*, *bathtub*, *counter*, and *others* (including all unlisted categories). The resolution of the map is $5cm$ for each cell and total $M = 720$ cells, yielding an effective spatial coverage of $36m \times 36m$. The local egocentric map of size $12m \times 12m$ is cropped to build the final semantic map of size $240 \times 240 \times 42$. When computing NBV, we construct voxel grids with $L = 64$ for selected objects and $L' = 128$ for environment points in $5m \times 5m \times 2.5m$ cubic area surrounding the selected objects. The reward coefficients are $w_e = 0.05$, $w_q = 0.1$, $w_s = -0.01$ (with linear scheduling from 0 to -0.01 as the training proceeds), and $w_g = -0.01$ for global policy and $w_v = 0.005$, $w_s = -0.05$, and $w_d = -0.03$ for local policy. Another set of hyperparameters balancing the losses is set to $\alpha_{\text{critic}} = 0.5$, $\alpha_{\text{entropy}} = 0.01$, $\alpha_{\text{PCP}} = 0.1$, $\alpha_{\text{SPR}} = 0.1$ and $\alpha_{\text{mask}} = 0.5$, respectively.

Implementation details. To train the system, we deploy our framework on a Cloud Server equipped with an Intel Xeon Platinum 8163 CPU (2.50GHz×96), 756GB RAM, and 8 Nvidia Tesla V100 GPUs each with 32GB RAM. We use the off-the-shelf episodes from the Habitat PointNav challenge for our global task, which specify the simulation scene, start position and orientation of this episode. For the local task, we generate about 2.5M episodes for network training and 2000 episodes for testing. The training takes about 5 days for global policy and 2 days for local policy on this machine, respectively. In the early stage of the training, the global policy first uses a heuristic local policy that selects the five nearest views for scanning when reaching the ROI. The local policy then replaces this heuristic policy after being successfully trained. When conducting the experiments, we ran the online semantic reconstruction with the three trained networks on a desktop with an AMD Ryzen 9 5950X CPU (3.4GHz×16), 64GB RAM, and an Nvidia GeForce RTX 3090 GPU with 24GB RAM.

3 ADDITIONAL ABLATION STUDIES

No exploration or quality reward. In our global scanning module, we carefully devise two dedicated reward terms to juggle scanning quality improvement with scene exploration to accomplish sufficient coverage with high-quality reconstruction. To evaluate the necessity of both of them, an experiment is conducted to train two more agents, each of which only receives one reward term as its positive stimulus (the other two negative terms keep the same).

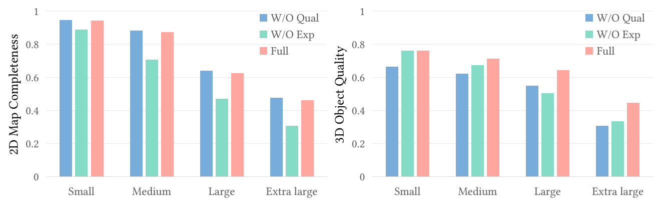

Fig. 2. Ablation study on the effect of exploration and quality terms of the reward on the performance of scene exploration and object reconstruction quality. Two more agents are trained with only exploration (Exp-only) or quality (Qual-only) terms for this experiment.

Table 1. Ablation study on the effect of semantic trimming on the performance of 3D scene completeness and 3D object quality.

Split	SC		OQ	
	Trim-sem	Full-sem	Trim-sem	Full-sem
Small	0.935	0.936	0.759	0.760
Medium	0.887	0.889	0.711	0.714
Large	0.665	0.663	0.642	0.642
Extra large	0.502	0.505	0.444	0.446

Table 2. Average inference time and memory usage of different modules in our method.

Component	SLAM framework	Global policy	Local policy
Time (ms)	167	12	10

As shown in Figure 2, the exploration-only agent spends most of its time loafing around and fails to acquire reconstruction with satisfactory quality. On the contrary, the quality-only agent exhibits a shortsighted strategy, which omits to cover enough space and objects in large environments.

No semantic trimming. In all experiments, we only preserve channels of the most prominent object categories in the 2D quality map and represent all excluded objects by *others* channel. The basic assumption is that the unobtrusive objects have little influence when predicting the scene layout and may induce extra computational costs. To verify our assumption, we compare the results of the trimmed semantic (Trim-sem) and full semantic (Full-sem) settings. As demonstrated in Table 1, the trimmed semantic setting shares similar performance with the full semantic but takes about 50% less time to train.

4 RUNTIME ANALYSIS

For runtime analysis of our system, we evaluate the inference time per frame of major components of our method on the desktop setting. Processing each frame includes a Habitat simulation with an average speed of 20 ms, an updating of the SLAM framework, and an inference of global policy or local policy since our method alternates between global ROI planning and local NBV planning. As shown in Table 2, the largest computational bottleneck and time-consuming operation is the SLAM framework, and both global and local policies are lightweight thanks to the elaborated 2D-3D representation and neat policy network. Moreover, a more advanced SLAM module can be adopted to improve the already acceptable real-time performance.

REFERENCES

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. 2018. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*. PMLR, 1407–1416.

Martin Runz, Maud Buffier, and Lourdes Agapito. 2018. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 10–20.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. 2019. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357* (2019).