Supplementary Materials of CSO: Constraint-Guided Space Optimization for Active Scene Mapping

1 Detailed Implementations

1.1 Implementation details

Inspired by [4, 12], we split the entire decision-making process into three parts: environment state generation, global policy, and local policy, as shown in figure 2 of the main paper. The environment state generation is used to generate the state input for the global policy, including the occupancy map, the distance map, and the frontiers-based entropy map. The global policy receives the state and predicts the long-term goal for the local policy. The local policy generates the specific actions for the robot. In our experiments, we set the global policy to be updated every 25 steps in the simulation environments, while in the real-world environments, we adjusted it to update every 15 steps based on practical considerations. During the local planning stage, the long-term goal does not change, and the local policy determines the action of each step according to it.

1.2 Data processing

Our simulation experiments are conducted on the Matterport3D dataset [3]. In addition to the standard train/val/test environments split of Matterport3D, we further split the dataset into small($< 30m^2$), middle($30 - 100m^2$), large($100 - 260m^2$), and super large($> 260m^2$) based on the traversable areas. The training set has 61 scenes, including 8 small scenes, 27 medium scenes, 16 large scenes, and 10 super large scenes. We evaluate our algorithm on both the test set and the val set, which contains 4 small scenes, 12 middle scenes, 10 large scenes, and 3 super large scenes. The splits are disjoint, therefore all evaluations are conducted in novel scenes.

1.3 Greedy-Dist and Greedy-Info

Based on [11], we further design two heuristic methods, Greedy-Info and Greedy-Dist, which select the long-term goal greedily based on the geodesic distance or frontiers-based entropy. Specifically, the Greedy-Info method selects the frontier with the maximum information entropy within the valid distance range as the long-term goal. When there are multiple frontiers with the same maximum entropy value at the same time, we select the frontier closest to the robot as the long-term goal. Similarly, the Greedy-Dist method chooses the closest frontier within the valid distance range as the long-term goal. When there are multiple frontiers with the same minimum distance value at the same time, we choose the frontier with the largest information entropy as the long-term goal. Both valid distance settings are the same as in our method.

1.4 Experiments in Real-World

In addition to simulated environments, we test our method in the real world. For experiments in the real world, we deploy the LIMO robot equipped with an ORBBEC Dabai Depth Camera and an EAI YDLIDAR X2L 360° 2D laser range lidar sensor to explore 3 distinct real-world scenes, as shown in figure 1. These scenes consist of 2 large scenes (a) and (b), alongside a super large scene (c), where (a) is composed of laboratories and corridors, (b) is a supermarket, and (c) is part of the student innovation and practice center. We leverage the gmapping algorithm [6], a particle filtering-based laser SLAM algorithm, to generate the robot's exploration map in the real world based on LiDAR and Odometry data. Simultaneously, we employ the ACML [1], an adaptive Monte Carlo localization algorithm to estimate the robot's pose while exploring accurately. More details of the real-world experiments can be found in the video.

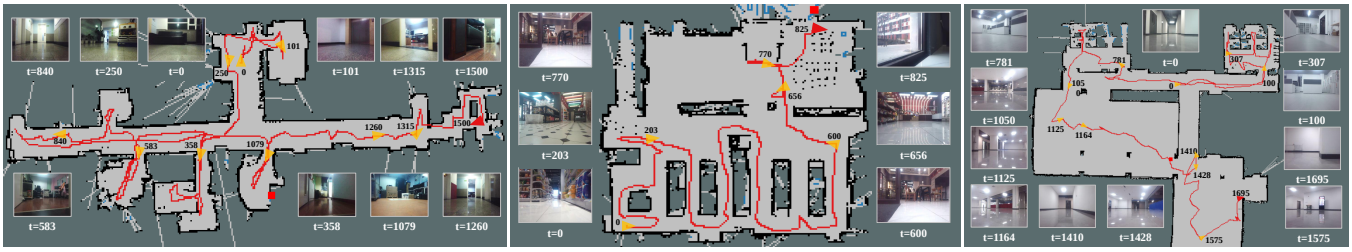
2 More Comparisons and Discussions

2.1 Comparison discussions

From the coverage curves in figure 4: (c),(f),(g),(h) of the main paper, it can be found that in the early stage of exploration, heuristics such as Greedy-Dist and Greedy-Info achieve a higher exploration efficiency than our method. However, in the later stage, our exploration curve will gradually be higher than others, either by being the first to finish exploring or by reaching the highest exploration completeness. This illustrates that although the greedy-based strategies can achieve higher exploration efficiency in some scenarios, it is easy to fall into local optimal and cannot maximize the long-term reward. In contrast, our approach uses frontiers-based entropy as the state input, while using action masks as the learning constraints to align the critic value space with the real-world geodesic distance space, thus achieving more efficient exploration and ensuring larger accumulated rewards.

As for other alternatives, ANS [4] uses the entire map as the policy input and outputs any position in the map as the next long-term goal without any constraints, which makes it easy to select goals that will cause long-distance round trips. NeuralCoMapping [12] and ARIADNE [2] enhance exploration performance by constraining the action space to frontiers or neighboring nodes around the robot instead of the entire map. However, the same problem is still inevitable because the distance from frontiers or neighboring nodes to the robot varies across different scenes due to different scene scales and dynamics while exploring. In addition, UPEN [5] uses RRT [8] for path planning based on the occupancy map uncertainty predicted by their model and selects the path that can maximize map uncertainty over candidate paths for exploration. Therefore, it depends heavily on the model predictions. At the same time, path planning based on RRT is also time-consuming.

It is worth noting that from figure 2, the planning of Greedy-Dist can also achieve an approximate performance to ours in some ideal scenarios, which further proves the alignment of the critic value space with the geodesic distance space in our method. Similar results can also be found in figure 6:(c)-6,(c)-7 of the main paper. However, in most scenarios, our method still outperforms the Greedy-Dist no matter in exploration efficiency, completeness, or path rationality. This is attributed to our method not solely relying on geodesic distance but integrating both frontiers-based entropy and geodesic distance to derive a set of action sequences that can maximize long-term rewards, thus achieving better performance.



(a) Laboratories and Corridors
$176.22m^2, 1500$ *steps*

(b) Supermarket
$170.44m^2, 825$ *steps*

(c) Student Innovation and Practice Center
$649.23m^2, 1695$ *steps*

Figure 1: We test our method in 3 real-world scenes using a LIMO robot, where the cumulative exploration area and steps are labeled below. For each exploration map, the gray areas are the explored regions, the black areas represent obstacles, the red line tracks the robot's exploration trajectory, the red triangle marks its current position and orientation, and the red square indicates its next long-term goal. Moreover, we marked some historical timesteps while exploring with the orange triangles, and the images around the exploration map are first-view RGB images of the robot at the corresponding timesteps.

Greedy Dist Ours

Figure 2: The similar exploration trajectories of ours and Greedy Dist in some ideal cases. The highly similar trajectories further demonstrate the alignment of the critic value space with the geodesic distance space in our method.

2.2 Comparison of Planning Time

Table 1: Comparison of planning time for different methods.

Method	Planning Time (*s*)
Greedy [11]	0.015
ANS [?]	0.004
UPEN(RRT) [5]	0.878
ARIADNE [2]	0.012
NeuralCoMapping [12]	0.019
Ours	0.024

Table 1 shows the planning time of different methods while exploring. Compared to the time-consuming traditional planning method RRT, our planning time is only 2.73% of it. At the same time, we can achieve a more efficient and complete exploration. It is worth noting that our method results in a longer planning time due to the use of frontiers-based entropy and action mask operations. Despite all of this, our method remains at a similar level while achieving better performance compared to other learning-based methods and simple greedy strategies.

3 Network Architecture Details

We use the off-policy learning approach Proximal Policy Optimization (PPO) [9] as the policy optimizer. The detailed network architecture is shown in figure 3.

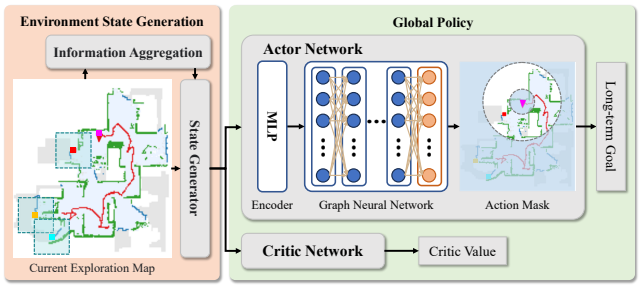


Figure 3: Network Architecture Details

The actor network is trained to learn a policy network that outputs the probability of each action being chosen. In our experiment, a graph neural network(GNN) [7] is adopted as the actor network to perform feature extraction and fusion based on the state input $s(\omega_t)$ generated by the state generator. Specifically, the actor network consists of two parts, the encoder and the GNN module. The encoder is composed of a multi-layer perception (MLP). We first construct a graph $G(F_t, \Omega_t)$ based on the frontiers F_t and the explored path $\Omega_t = \{\omega_0, ..., \omega_t\}$ to represent the context of the current scene. It establishes the correspondences between the robot and frontier nodes extracted from the constructed occupancy map M_t. We distribute the information given by state $s(\omega_t)$ into nodes and edges of $G(F_t, \omega_t)$. For each node n_i, the input feature $f(n_i) \in \mathbb{R}^5$ includes the (x, y) coordinates in M_t, the semantic label that indicates $n_i \in F_t$ or $n_i \in \Omega_t$, the history label that indicates n_i is ω_t or explored nodes $n_i \in \{\omega_0, ..., \omega_{t-1}\}$, the proposed frontiers-based entropy $I_{n_i}(M_t)$, and finally obtain a 32-dimensional feature. The edge feature $f(n_i, n_j) \in \mathbb{R}^{32}$ is also given by a MLP with $l_{ij} \in \mathbb{R}^1$ indicates the geodesic distance from node n_j to node n_i. Then we feed these node and edge features into the GNN network consisting of 3 GNN layers for feature transfer and output a set of scores.

Based on these scores, we compute the probability $\Pi_{\text{mask}}(f|s(\omega_t))$ of each action, followed by action mask-based action sampling.

The critic network trains a state-value network that predicts the state value $V(s(\omega_t))$ to indicate how much reward is earned from the current state, which is adopted to train the actor network. In our experiment, the state input of the critic network is a vector of $[6, w, h]$, where $[w, h]$ is the size of the exploration map, and the 6 channels are obstacle map, frontiers map, robot current position map, robot historical trajectory map, explored map, and explorable map respectively. The initial $[w, h]$ is set to $[480, 480]$, and before feeding into the critic network, we use a maxpool operation to compress it to $[120, 120]$ for ease of calculation. The critic network consists of 5 convolutional layers and 3 linear layers. A flatten operation is inserted in the middle as a separation to flatten the features into one dimension.

Based on the robot's position, the estimated long-term goal, and the exploration map, the local policy generates a moving trajectory from the robot to the long-term goal. We adopt the Fast Marching Method (FMM) [10] to achieve this purpose.

References

[1] Omur Arslan and Daniel E Koditschek. 2016. Voronoi-based coverage control of heterogeneous disk-shaped robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4259–4266.

[2] Yuhong Cao, Tianxiang Hou, Yizhuo Wang, Xian Yi, and Guillaume Sartoretti. 2023. Ariadne: A reinforcement learning approach using attention-based deep networks for exploration. *arXiv preprint arXiv:2301.11575* (2023).

[3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158* (2017).

[4] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. 2020. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155* (2020).

[5] Georgios Georgakis, Bernadette Bucher, Anton Arapin, Karl Schmeckpeper, Nikolai Matni, and Kostas Daniilidis. 2022. Uncertainty-driven planner for exploration and navigation. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 11295–11302.

[6] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. 2007. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics* 23, 1 (2007), 34–46.

[7] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[8] Steven M. LaValle. 1998. Rapidly-exploring random trees: a new tool for path planning. *The annual research report* (1998).

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[10] James A Sethian. 1996. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences* 93, 4 (1996), 1591–1595.

[11] Brian Yamauchi. 1997. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 146–151.

[12] Kai Ye, Siyan Dong, Qingnan Fan, He Wang, Li Yi, Fei Xia, Jue Wang, and Baoquan Chen. 2022. Multi-Robot Active Mapping via Neural Bipartite Graph Matching. In *CVPR*.