SUPPLEMENTARY MATERIALS OF LEARNING EFFICIENT MULTI-AGENT COOPERATIVE VISUAL EXPLORATION

Chao Yu[1], **Xinyi Yang**[1*], **Jiaxuan Gao**[2*], **Huazhong Yang**[1], **Yu Wang**[1], **Yi Wu**[23]
[1] Department of Electronic Engineering, Tsinghua University
[2] Institute for Interdisciplinary Information Sciences, Tsinghua University
[3] Shanghai Qi Zhi Institute

A APPENDIX

We would suggest to visit `https://sites.google.com/view/maans` for more information.

A.1 MAANS DETAILS

A.1.1 NEURAL SLAM MODULE AND THE LOCAL POLICY

We reuse the neural SLAM module and the local policy from origin ANS paper (Chaplot et al., 2020). We briefly explain the functionality of neural SLAM module and the local policy here, while for full detailed description, please refer to Chaplot et al. (2020).

The neural SLAM module takes in as input current RGB observation o_i, current and last noised sensor readings of pose $x'_{i-1:i}$, last pose estimation $\hat{x}_{i-1}$ and last map prediction $\hat{m}_{i-1}$, and outputs a map prediction $\hat{m}_i$ and a pose estimation $\hat{x}_i$. The neural SLAM module consists of two components, a mapper that first predicts the egocentric map and combines it with last map prediction $\hat{m}_{i-1}$ to obtain current map prediction $\hat{m}_i$, and a pose estimator that predicts the pose change $\hat{dx}_i$ and outputs current pose estimation $\hat{x}_i = \hat{x}_{i-1} + \hat{dx}_i$.

The local policy takes as input the relative angle and distance to the short-term goal and current RGB observations and then outputs an navigational action. Note that each agent itself maintains a reconstructed map based purely on its own previous observations, so at every environment step every agent independently runs the SLAM module to update his map.

A.1.2 SPATIAL COORDINATION PLANNER

Input Representation

Spatial Coordination Planner is the core component of MAANS. In SCP, each CNN-based feature extractor's input map, i.e. one feature extractor per agent, is a 240×240 map with 7 channels, including

- Obstacle channel: Each pixel value denotes the probability of being an obstacle.
- Explored region channel: A probability map for each pixel being explored.
- One-hot position channel: The only one none-zero grid denotes the position of the agent.
- Trajectory channel: This is used to represent the agent's history trace. To reflect time-passing, this channel is updated in an exponentially decaying weight manner. More precisely, an agent's trajectory channel V^t at timestep t is updated as following,

$$V^t_{x,y} = \begin{cases} 1 & \text{if agent is near } (x,y) \\ \varepsilon V^{t-1}_{x,y} & \text{otherwise} \end{cases}$$

  where the agent is regarded as near (x, y) when the grid-level distance between them is less than 3.

- One-hot global goal channel: This channel is a one-hot image demonstrating the position of last-step global goal.
- Goal history channel: This records all the previous global goals of the agent.
- Agent ID channel: This is a constant channel to reflect the agent's identity so as to distinguish among different agents. The value of every cell in the map is a normalized ID.

All mapping-related channels are transformed into world-view to save SCP from learning to align all agents' information, which might involve rotation and translation. To enhance agent identity, the value of the trajectory channel, the one-hot position channel, the one-hot global goal channel and the goal history channel are all multiplied by a normalized ID, just like the normalized one in the agent ID channel. The normalized ID of agent $k(1 \leq k \leq n)$ is defined as $\frac{k}{n(n+1)/2}$. The decay parameter ε is 0.9.

To ensure fully awareness of the decision agent k, that agent is always put in the *first* place, that is, his related input maps are always fed into the *first* feature extractor.

Hierarchical Action Space

Through SCP, every agent chooses a long-term goal (a point) from the whole space. A natural choice is to model the agent's policy as a multi-variable Gaussian distribution to select points from a plane. However, in our exploration setting, an agent's policy could be extremely multi-modal especially during early stage of exploration since many points could induce similar effects on the agent's path. To fix this issue, we adopt a hierarchical design. We first divide the whole map into 8×8 regions, from which the agent chooses a desired region. Then, similar to previous choice, a point in this region is selected as the long-term goal. Formally, the policy of agent k, could be described as,

$$\begin{aligned} g_r, g_c &\sim \text{Cat}(r_\theta) \\ x_l, y_l &\sim \mathcal{N}(\mu_\theta, \Sigma_\theta) \\ x_l' = \text{sigmoid}(x_l),&\ \ y_l' = \text{sigmoid}(y_l) \\ x_g = (g_r + x_l')/8,&\ \ y_g = (g_c + y_l')/8 \end{aligned}$$

where θ is the model parameter, Cat(r_θ) is the 8×8 categorical distribution for choosing the region, g_r, g_c are the row and column indexes of the sampled region, $\mu_\theta, \Sigma_\theta$ are the mean and covariance matrix of the Gaussian distribution to choose the local point within the region and (x_g, y_g) is the final sampled long-term goal.

A.1.3 Reward Function

We use 4 kinds of team-based reward, including a coverage reward, a success reward, an overlap penalty and a time penalty. In the following part, $Ratio^t$ denotes the total coverage ratio at timestep t. Let Exp^t be the merged explored map at timestep t and Exp_k^t be the explored map of agent k. Ideally both Exp^t and Exp_k^t can be considered as sets of explored points. Then define $\Delta Exp^t = Exp^t \backslash Exp^{t-1}$ as the newly discovered region at timestep t by the whole team with regard of the merged explored area. Specially, we model an individual's effort by $\Delta Exp_k^t = Exp_k^t \backslash Exp^{t-1}$, that is agent k's contribution at timestep k based on the whole team's previous exploration. Note that ΔExp_k^t is not defined based on the agent's previous exploration, i.e. $\Delta Exp_k^t \neq Exp_k^t \backslash Exp_k^{t-1}$. In practice, the exploration and obstacle maps are stored as real numbers, denoting probability of being explored and occupied. The reward gained by agent k at timestep t has 4 parts,

- **Coverage Reward:** The coverage reward consists of two parts, a team coverage reward and an individual coverage reward. The team coverage reward is proportional to area of the exploration increment ΔExp^t. The individual coverage reward, as the name suggests, is proportional to the individual contribution, i.e., the area of ΔExp_k^t. The coefficients for both parts are 0.02.
- **Success Reward:** Agent k gets a success reward of $1 \cdot Ratio^t$ when 95% coverage rate is reached and $0.5 \cdot Ratio^t$ when 90% coverage rate is achieved.

- **Overlap Penalty:** The overlap penalty $r_{overlap}$ is designed to encourage agents to reduce repetitive exploration and learn to cooperate with others. It is defined as

$$r_{overlap} = \begin{cases} -A_{overlap} \times 0.01, \ Ratio^t \ < 0.9 \\ -A_{overlap} \times 0.006, 0.9 \leq \ Ratio^t < 0.95 \\ 0, \ 0.95 \leq Ratio^t \end{cases}$$

  where $A_{overlap}$ is the increment of overlapped explored area between agent k and other agents. Ideally the overlapped area between agent k and agent u could be described as $Overlap^t_{k,u} = Exp^t_k \cap Exp^t_u$, while in practice the values of all explored maps are real numbers denoting the probability, hence in our implementation a grid is considered overlapped only when the sum of this grid's occupancy probabilities at the two agents' explored maps is greater than 1.2. We define $\Delta Overlap^t_{k,u} = Overlap^t_{k,u} \backslash Overlap^{t-1}_{k,u}$. Then $A_{overlap}$ is the sum of $\Delta Overlap^t_{k,u}$'s area over all other agents, i.e. $u \in \{1, \cdots, n\} \backslash \{k\}$.

- **Time Penalty:** The time penalty r_{time} is designed to encourage agents' exploration efficiency. It is defined as

$$r_{time} = \begin{cases} -0.002, \ Ratio^t \ < 0.9 \\ -0.001, 0.9 \leq \ Ratio^t < 0.95 \\ -0.0002, \ 0.95 \leq Ratio^t < 0.97 \end{cases}$$

The final team-based reward is simply the sum of all these terms. All the explored and obstacle maps are represented under discretization of $5cm$ and all the area computations are taken in m^2.

A.1.4 Architecture

Our models are trained and implemented using Pytorch (Paszke et al., 2017). We reuse the neural SLAM module and local policy from Chaplot et al. (2020) and we briefly summarize their architectures here. Neural SLAM module has two components, a Mapper and a Pose Estimator. The Mapper is composed of ResNet18 convolutional layers, 2 fully-connected layers, and 3 deconvolutional layers. The Pose Estimator consists of 3 convolutional layers and 3 fully connected layers. Similarly, the local policy has Resnet18 convolutional layers, fully-connected layers and a recurrent GRU layer.

Table 1: CNN Block Hyperparameter

Layer	Out Channels	Kernel Size	Stride	Padding
1	32	3	1	1
2	64	3	1	1
3	128	3	1	1
4	64	3	1	1
5	32	3	2	1

The Spatial Coordination Planner (SCP) has three main components, including CNN-based feature extractors, a transformer-based relation encoder and a spatial action decoder.

1. Each CNN-based feature extractor contains 5 consecutive CNN blocks. Their corresponding parameters are shown in Tab. 1. We use ReLU as the activation function. After each of the front four CNN blocks, we attach a 2D max pooling layer with 2 kernel size.
2. The transformer-based relation encoder consists of embeddings for agent-specific position and an attention layer. Embeddings are used to better capture spatial information. The attention layer has 4 heads, with 32 dimension size for each head.
3. The spatial action decoder simply uses a CNN projector and linear transformations to turn the feature map output from the transformer-based relation encoder to corresponding logits for Categorical distribution (region head) and means and standard deviations of the Gaussian distribution (point heads).

The critic also utilizes a similar architecture as SCP, except replacing the spatial action decoder with fully-connected layers to output value predictions. For full details about the architecture, please refer to the open-source code.

Table 2: MAPPO hyperparameters

common hyperparameters	value
gradient clip norm	10.0
GAE lambda	0.95
gamma	0.99
value loss	huber loss
huber delta	10.0
mini batch size	batch size / mini-batch
optimizer	Adam
optimizer epsilon	1e-5
weight decay	0
network initialization	Orthogonal
use reward normalization	True
use feature normalization	True
learning rate	2.5e-5
episode length	300
number of local steps	15

A.2 TRAINING DETAILS

Algorithm 1 MAPPO

```
Initialize $\theta$, the parameters for policy $\pi$ and $\phi$, the parameters for critic $V$, using Orthogonal
initialization (Hu et al., 2020)
Set learning rate $\alpha$
while $step \leq step_{\max}$ do
  set data buffer $D = \{\}$
  for $i = 1$ to $num\_rollouts$ do
    $\tau = []$ empty list
    for $t = 1$ to $T$ do
      for all agents $a$ do
        $p_t^{(a)} = \pi(o_t^{(a)}; \theta)$
        $u_t^{(a)} \sim p_t^{(a)}$
        $v_t^{(a)} = V(s_t^{(a)}; \phi)$
      end for
      Execute actions $\boldsymbol{u_t}$, observe $r_t, s_{t+1}, \boldsymbol{o_{t+1}}$
      $\tau \mathrel{+}= [s_t, \boldsymbol{o_t}, \boldsymbol{u_t}, r_t, s_{t+1}, \boldsymbol{o_{t+1}}]$
    end for
    Compute advantage estimate $\hat{A}$ via GAE on $\tau$
    Compute reward-to-go $\hat{R}$ on $\tau$ and normalize
    $D = D \cup \tau$
  end for
  for epoch $k = 1, \ldots, K$ do
    $b \leftarrow$ sequence of random mini-batches from D with all agent data
    for batch $c$ in $b$ do
      Adam update $\theta$ on $L(\theta)$ with batch $c$
      Adam update $\phi$ on $L(\phi)$ with batch $c$
    end for
  end for
end while
```

We use the neural SLAM module and the local policy and directly use the trained model provided in origin ANS paper (Chaplot et al., 2020).

The multi-agent reinforcement learning (MARL) framework for SCP training is Multi-Agent PPO (MAPPO) (Yu et al., 2021a), which is an extension of PPO (Schulman et al., 2017) to multi-agent scenarios. The pseudocode of MAPPO is provided in Algo. 1. Hyperparameters are shown in Tab. 2. Note that global goals are chosen every 15 steps, yielding 20 global planning steps in one episode.

To further improve our solution's generalization ability, we use a final training-and-distillation solution. To be more explicit, we first train scene-specific teachers over all training scenes and then train a student model *MAANS-TD* by running policy distillation over these teachers. Suggested by Czarnecki et al. (2019), our training procedure is on-policy manner. During each episode, we parallelly collect data in all the training environments using the student and then run behavior cloning between the student and the teachers using these on-policy data. We use KL divergence loss for the region head and MSE loss for the point heads. Formally speaking, we try to minimize following loss,

$$L(\theta) = \sum_t \mathbb{E}_{\pi_{\theta_s}}[D_{KL}(r_{\theta_s}||r_{\theta_t}) + ||\mu_{\theta_s} - \mu_{\theta_t}||^2|t]$$

where θ_s is the parameter of the student model, θ_t is the parameter of the teacher model, $r_{\theta_s}, r_{\theta_t}$ are the 8×8 categorical distributions for region selection, μ_{θ_s} and μ_{θ_t} are the mean of the Gaussian distribution for point selection. Here the expectation is conditional on t because we have an expert for every training scene, conditioning on t means the training environment is the corresponding scene.

For each episode, the epoch size is 4. Note that we only need to train the actor of the student since we do not need a critic to produce value estimations. We use an Adam optimizer with a learning rate of 0.000025.

A.3 Planning-based Baselines

We demonstrate some details about the four planning-based baselines here.

Utility: A method that always chooses frontier that maximizes information gain (Burgard et al., 2005).

Nearest: A method that always chooses the nearest frontier as long-term goal (Yamauchi, 1997). The distance to a frontier is computed using breadth first search on the occupancy map.

APF: Artificial Potential Field (APF) (Yu et al., 2021b) plans a path for each agent based on a computed potential field. The end of the path, which is a frontier, is the selected goal. For every agent, an artificial potential field F is computed in the discretized map, with consideration of distance to frontiers, presence of obstacles and potential exploration reward. APF also introduces resistance force as a simple mechanism. Finally the path is generated along the fastest decreasing direction of F, starting from the agent's current position.

RRT: This baseline is adopted from Umari & Mukhopadhyay (2017). Rapid-exploring Random Tree (RRT) is originally a path planning algorithm based on random sampling and is used as a frontier detector in Umari & Mukhopadhyay (2017). After collecting enough frontiers through random exploration, RRT chooses frontier p with the largest utility $u(p) = IG(p) - N(p)$, where $IG(p)$ and $N(p)$ are respectively the normalized information gain and navigation cost of p.

To avoid visual blind area and ensure that selected frontiers are far enough, the area within $2.5m$ from each agent is considered explored when making global planning. The information gain of a frontier p is computed as the number of unexplored grids within $1.5m$ to p. All these baselines do re-planning every 15 environment steps, which is consistent to SCP.

Pseudocode of APF is shown in Algo. 2. Line 6-12 computes the resistance force between every pair of agents where D is the influence radius. In line 13-18, distance maps starting from cluster centers are computed and the corresponding reciprocals are added into the potential field so as one agent approaches the frontier, the potential drops. Here w_c is the weight of cluster c, which is the number of targets in this cluster. Consequently an agent would prefer to seek for frontiers that are closer and with more neighboring frontiers. Line 20-25 shows the process to find the fastest potential descending path, at each iteration the agent moves to the cell with the smallest potential among all neighboring ones. T is the maximum number of iterations and C_{repeat} is repeat penalty to avoid agents wandering around cells with same potentials.

Pseudocode of RRT is shown in Algo. 3. In each iteration, a random point p is draw and a new node t is generated by expanding from s to p with distance L, where s is the closest tree node to

Algorithm 2 Artificial Potential Field (APF)

```
Input : Map $M$, number of agents $n$ and agent locations $loc_1 \dots loc_n$.
Output : Selected goals
 1: $P \leftarrow$ frontiers in $M$
 2: $C \leftarrow$ clusters of frontiers $P$
 3: $goals \leftarrow$ an empty list
 4: for $i = 1 \rightarrow n$ do
 5:     $F \leftarrow$ zero potential field, i.e., a 2d array
 6:     for $j = 1 \rightarrow n$ do
 7:         for empty grid $p \in M$ do
 8:             if $j \neq i$ and $||p - loc_j||_2 < D$ then
 9:                 $F_p \leftarrow F_p + k_D \cdot (D - ||p - loc_j||_2)$
10:             end if
11:         end for
12:     end for
13:     for $c \in C$ do
14:         Run breadth-first search to compute distance map $dis$ starting from $c$
15:         for empty grid $p \in M$ do
16:             $F_p \leftarrow F_p - dis_p^{-1} \cdot w_c$
17:         end for
18:     end for
19:     $u \leftarrow loc_i, cnt \leftarrow 0$
20:     while $u \notin M$ and $F_u$ is not a local minima and $cnt < T$ do
21:         $cnt \leftarrow cnt + 1$
22:         $F_u \leftarrow F_u + C_{repeat}$
23:         $u \leftarrow \arg\min_{v \in Neigh(u)} F_v$
24:     end while
25:     append $u$ to the end of $goals$
26: end for
27: return $goals$
```

Algorithm 3 Rapid-exploring Random Tree (RRT)

```
Input : Map $M$ and agent location $loc$.
Output : Selected frontier goal
 1: $NodeList \leftarrow \{loc\}, Targets \leftarrow \{\}$
 2: $i \leftarrow 0$
 3: while $i < T$ and $|Targets| < N_{target}$ do
 4:     $i \leftarrow i + 1$
 5:     $p \leftarrow$ a random point
 6:     $s \leftarrow \arg\min_{u \in NodeList} ||u - p||_2$
 7:     $t \leftarrow Steer(s, p, L)$
 8:     if $No\_Collision(M, s, t)$ then
 9:         if $t$ lies in unexplored area then
10:             $Targets \leftarrow Targets + \{t\}$
11:         else
12:             $NodeList \leftarrow NodeList + \{t\}$
13:         end if
14:     end if
15: end while
16: $C \leftarrow$ clusters of points in $Targets$.
17: $goal \leftarrow \arg\min_{c \in C} IG(c) - N(c)$
18: return $goal$
```

p. If segment (s, t) has no collision with obstacles in M, t is inserted into the target list or the tree according to whether t is in unexplored area or not. Finally, the goal is chosen from the target list with the largest utility $u(c) = IG(c) - N(c)$ where $IG(c)$ is the information gain and $N(c)$ is the navigation cost. $IG(c)$ is computed by the number of unexplored grids within $1.5m$ to c, as mentioned above. $N(c)$ is computed as the euclidean distance between the agent location and point c. To keep these two values at the same scale, we normalize $IG(\cdot)$ and $N(\cdot)$ to $[0, 1]$ w.r.t all cluster centers.

A.4 Evaluation Metric

A.4.1 Accumulative Coverage Score (ACS)

we proposed ***Accumulative Coverage Score*** (ACS) as our performance metric. let $Ratio^t$ denote the coverage rate, i.e., the ratio of explored region to the total explorable area, at timestep t for an episode. The ACS number at timestep k, ACS_k, is computed by $ACS_k = \sum_{t=0}^{k} Ratio^t$. A higher ACS number implies faster exploration. We take the ACS number at timestep 200 in all our experiments.

A.4.2 Behavior Statistics

we also consider 3 additional behavior statistics measurement to capture different characteristics of a particular exploration strategy. Let $Overlap_{i,j}^t$ denote the ratio of the overlapped area explored by agent i and j to total explorable area at timestep t.

- **Coverage Ratio**: The *final* ratio of explored area to total area when an episode terminates, i.e., $Ratio^T$ where T is the episode length.
- **Steps**: The timesteps that the agents use to reach a 90% coverage, which is defined as $\min\{t | Ratio^t \geq 90\%\}$.
- **Overlap Ratio**: The ratio of average overlapped area explored by each pair of agents to the current explored area when 90% coverage is reached. Formally, the overlap ratio metric is defined as $\frac{1}{n(n-1)/2} \sum_{i<j} Overlap_{i,j}^{\tilde{t}} / Ratio^{\tilde{t}}$ where $\tilde{t} = \min\{t | Ratio^t \geq 90\%\}$.

A.5 Additional Experiment Results

A.5.1 Training Performance

Fig. 1 shows the within-episode exploration efficiency and ACS performance of 2 agents and 3 agents on training maps by different trained policies.

A.5.2 Comparison with Other ANS Variants

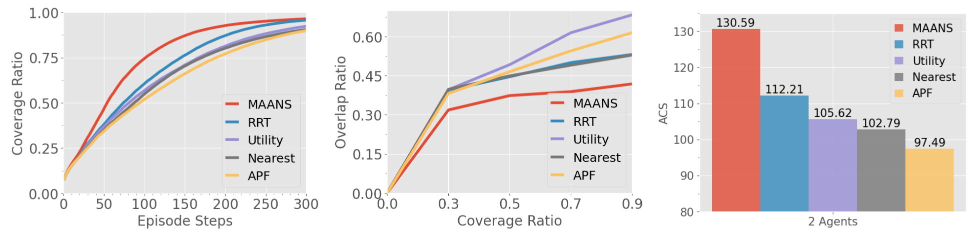
We measure the *ACS*, *Steps* and the *Overlap Ratio* metric over these 3 maps and demonstrate the training curves in Fig. 2.

A.5.3 Ablation Study on SCP

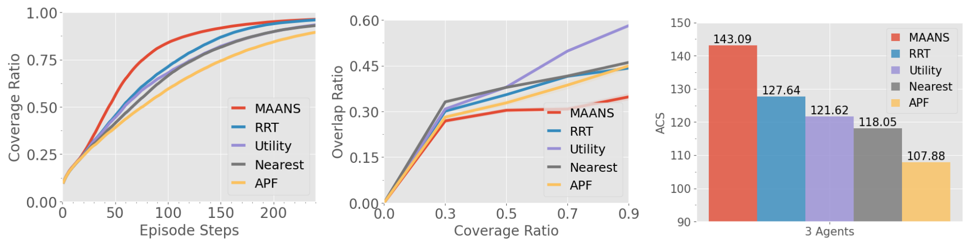
Fig.3 shows ablation studies on *SCP* vs. *SCP w.o. AE.* on 3 representative training maps.

Fig.4 shows ablation studies on *SCP* vs. *SCP w.o. RE.* on 2 representative training maps.

Fig.5 shows ablation studies on *SCP* vs. *SCP-merge* on 3 representative training maps.

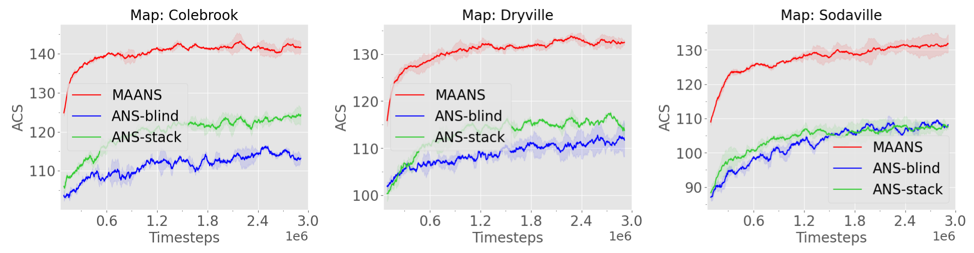


(a) 2 agents

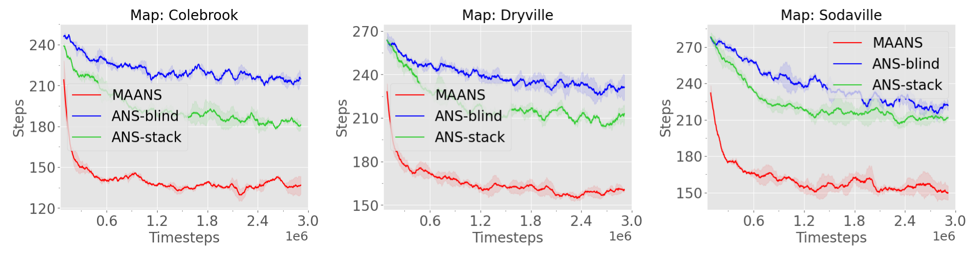


(b) 3 agents

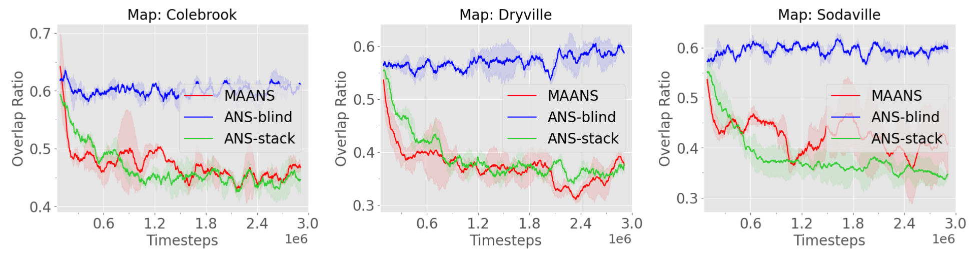
Figure 1: The left two figures show within-episode exploration efficiency on training maps by different trained policies. We measure the coverage rate w.r.t. episode step (left, higher the better) and overlap ratio w.r.t. coverage ratio (right, lower the better). As exploration proceeds, agents by MAANS cover explorable space much faster with a significantly lower overlap ratio. The rightmost figure shows the comparison of ACS performance between MAANS and other planning-based methods.



(a) ACS

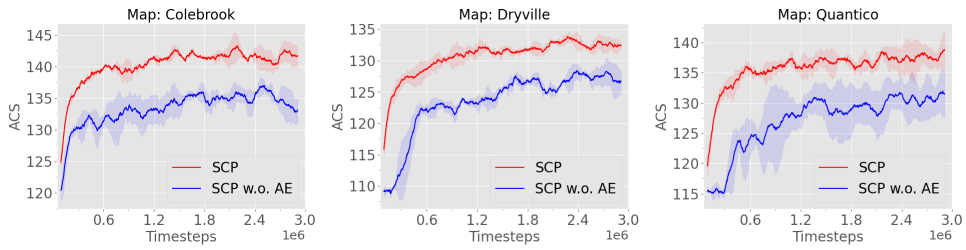


(b) Steps

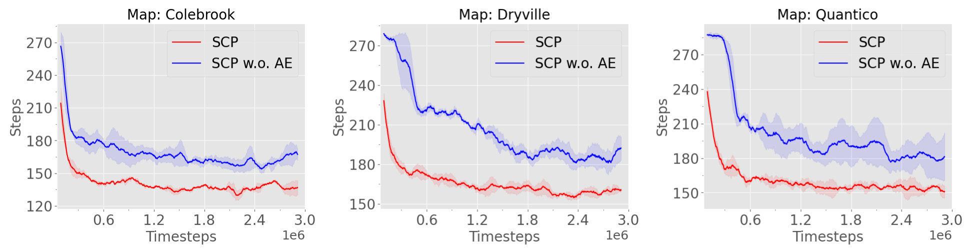


(c) Overlap Ratio

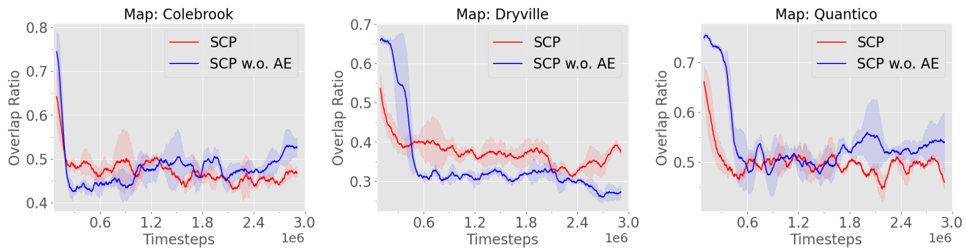
Figure 2: Comparison between MAANS and other ANS variants on 3 representative training maps.



(a) ACS



(b) Steps



(c) Overlap Ratio

Figure 3: Ablation Studies on SCP vs. SCP w.o. AE. on 3 representative training maps.

REFERENCES

Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations*. ICLR, 2020.

Wojciech Marian Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M. Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. *CoRR*, abs/1902.02186, 2019. URL `http://arxiv.org/abs/1902.02186`.

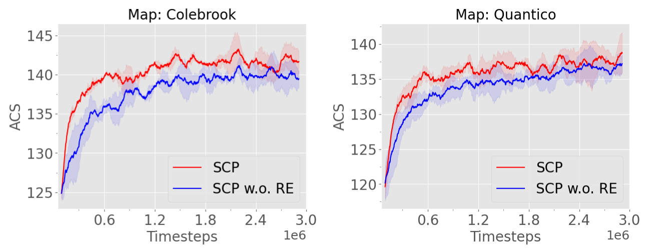
Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

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

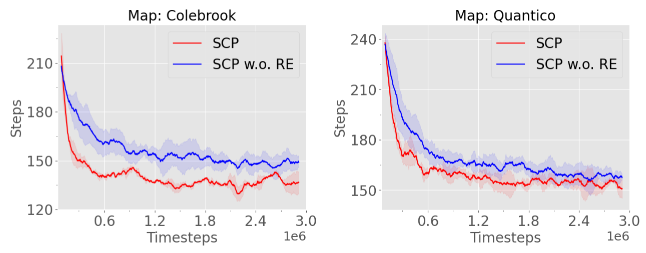
Hassan Umari and Shayok Mukhopadhyay. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1396–1402, 2017. doi: 10.1109/IROS.2017.8202319.

Brian Yamauchi. 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'*, pp. 146–151. IEEE, 1997.

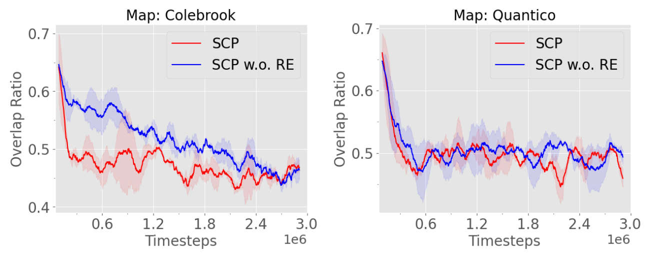
Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021a.



(a) ACS



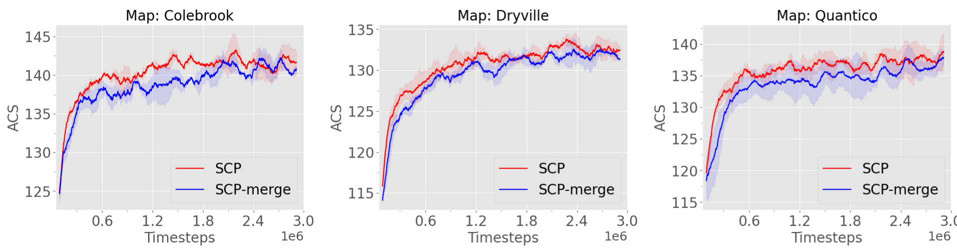
(b) Steps



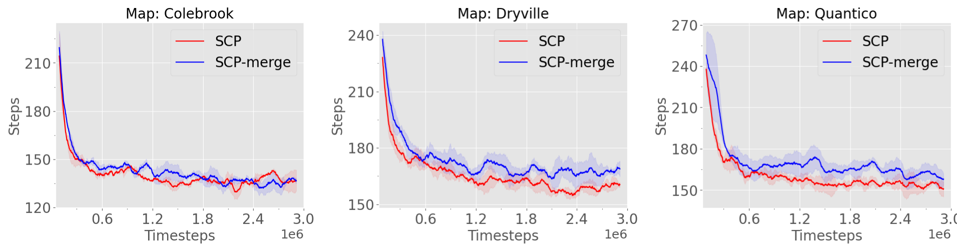
(c) Overlap Ratio

Figure 4: Ablation Studies on SCP vs. SCP w.o. RE. on 2 representative training maps.

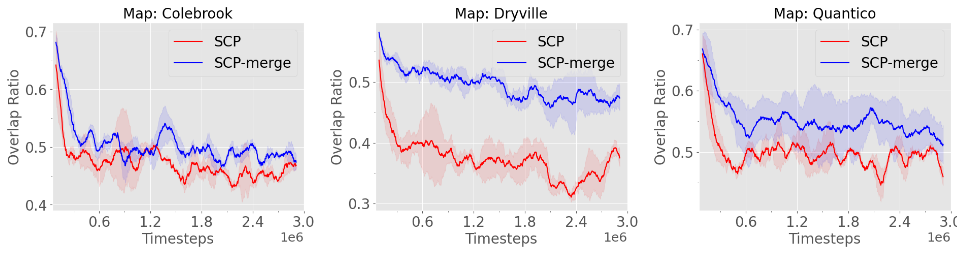
Jincheng Yu, Jianming Tong, Yuanfan Xu, Zhilin Xu, Haolin Dong, Tianxiang Yang, and Yu Wang. Smmr-explore: Submap-based multi-robot exploration system with multi-robot multi-target potential field exploration method. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021b.



(a) ACS



(b) Steps



(c) Overlap Ratio

Figure 5: Ablation Studies on SCP vs. SCP-merge on 3 representative training maps.