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†], and Yi Wu[23†]

[1] Department of Electronic Engineering, Tsinghua University
[2] Institute for Interdisciplinary Information Sciences, Tsinghua University
[3] Shanghai Qi Zhi Institute
* Equal contribution
† Corresponding author
{zoeyuchao,jxwuyi}@gmail.com,yu-wang@tsinghua.edu.cn

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

A MAANS Details

Multi-Agent Active Neural SLAM (MAANS) consists of 4 modules (1) Neural SLAM; (2) Map Refiner and Map Merger; (3) Local Policy and Local Planner; (4) Multi-agent Spatial Planner (MSP). Here we describe each module in detail.

A.1 Neural SLAM

The Neural SLAM Module for map reconstruction and pose estimation and the Local Policy for action output in our work are directly derived from ANS [2]. Neural SLAM Module trained by supervised learning provides each agent an updated reconstructed map individually at every timestep. In order to recover a metric map with high accuracy, Neural SLAM Module takes as input current RGB observation o_t, current and last pose $x'_{t-1:t}$ from sensors, last pose estimation $\hat{x}_{t-1}$ and last map prediction $\hat{m}_{t-1}$, and outputs a map prediction $\hat{m}_t$ and a pose estimation $\hat{x}_t$, where t represents the current timestep. Note that noises are introduced in simulation to mimic realistic situations.

A.2 Map Refiner and Map Merger

For a better choice of cooperative global goals, we designed a Map Refiner for arranging all maps into the same coordinate system and a Map Merger for shared map reconstruction. More concretely, Map Refiner obtains the egocentric *global* map from a series of past egocentric *local* maps and unifies the global maps from all agents in the same coordinate system based on the pose estimates. Besides, there is a dilemma that the egocentric local map contains part of redundant space if an agent reaches the edge of the house, resulting in a large portion of invisible region around the explorable area. To promise an effective CNN feature extraction and more accurate global goal generation, we clip the unexplorable boundary and enlarge the explorable region.

Map Merger leverages all enlarged global maps from the Map Refiner to compose a shared map through max-pooling operator for each pixel location, which indicates the probability of being explored or the obstacle. As a result, the local planner produces the sub-goals on the merged global map, which is much more informative. Note that the merged map is merely employed in local planner to plan path, but not introduced in MSP, which only utilizes agents' egocentric global maps to infer global goals.

A.3 Local Planner and Local Policy

To effectively reach a global goal, the agent first plans a path to this long-term goal in a manually merged global map using Local Planner, which is mainly based on Fast Marching Method (FMM) [8], and generates a sequence of short-term sub-goals. The Local Policy learns to produce next action via imitation learning. The input of the Local Policy includes the relative angle and distance from the current position to the short-term goal as well as current RGB observations.

A.4 Multi-agent Spatial Planner

Input Representation The shared CNN Feature Extractor in Multi-agent Spatial Planner firstly takes in a 240×240 map with 6 channels as input, containing

- Obstacle channel: indicating the likelihood of being an obstacle of each pixel
- Explored region channel: denoting the probability of being explored of each pixel
- One-hot position channel: describing the position of the agent with an one-hot metric map.
- Trajectory channel: expressing the history trace of each agent with exponentially decaying weight to emphasize the direction of the trace:

$$V_{x,y}^{t} = \begin{cases} 1 & \text{current position} \\ \varepsilon V_{x,y}^{t-1} & \text{otherwise} \end{cases}$$

  where V^t denotes the trajectory channel at timestep t.
- One-hot global goal channel: demonstrating the position of the last global goal in an one-hot manner.
- One-hot Goal history channel: recording all the previous global goals of the agent.

Besides CNN spatial maps, we also introduce additional features, including agent-specific embeddings of its identity and current position and grid features, i.e., the embeddings of the relative coordinate of each grid to the agent position as well as the embedding of the previous global goal.

- Position Embedding: described as trainable $G \times G \times D$ parameters as part of the neural network. Two types of position embeddings are used to distinguish from the decision-making agent and it's partners. Note that G and D is respectively 8 and 128.
- Relative Coordinate Embedding: describing the relative position of the agent with the $G \times G$ coarse-grained maps.
- Previous Global Goal Embedding: expressing the relative position of last global goal with the $G \times G$ coarse-grained maps.

These embeddings are all concatenated with features outputted by CNN feature extractors and then fed into MSP.

Hierarchical Action Space MAANS adopts a hierarchical action space to represent global goals, where a global goal is consequently composed of a high-level discrete *region* $g = (g_x, g_y)$ and a low-level fine-grained continuous *point* $p = (p_x, p_y)$. To be more specific, the whole world-frame occupancy map is discretized into 8×8 uniform regions and a global goal (x_l, y_l) is decomposed into two levels,

$$x_l = \frac{g_x + p_x}{8}$$
$$y_l = \frac{g_y + p_y}{8}$$

In MSP, the region head outputs a 64-dim vector denoting the categorical distribution of g, while the point head outputs a bivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. Point p is obtained by applying Sigmoid to the Gaussian random variables, i.e.,

$$\hat{p} = (\hat{p}_x, \hat{p}_y) \sim \mathcal{N}(\mu, \Sigma)$$
$$p_x = \text{Sigmoid}(\hat{p}_x)$$
$$p_y = \text{Sigmoid}(\hat{p}_y)$$

Reward Function We design the reward in a team-based fashion, comprising of the coverage reward, the success reward, the overlap penalty and the time penalty. For a unified representation, $Ratio^t$ indicates the total coverage ratio at timestep t, $Area^t$ is the the total coverage area at timestep t, and $Area_k^t$ represents the explored area of agent k. The details of 4 kinds of reward gained by agent k at timestep t are listed as below.

- **Coverage Reward:** The coverage reward is a combination of the team coverage reward and the individual coverage reward. Team coverage reward illustrates the increment of explored area at timestep t, and is proportional to $\Delta Area^t = Area^t - Area^{t-1}$. For the consideration of individual contribution to the whole team exploration, the individual coverage reward is proportional to $\Delta Area_k^t = Area_k^t - Area_k^{t-1} \bigcap Area^{t-1}$. We remark that $\Delta Area_k^t \neq Area_k^t - Area_k^{t-1}$, which suppresses cooperation but leads to the individual exploration. The coefficient is 0.02.

- **Success Reward:** To encourage agent k achieves the target coverage ratio as much as possible, we give the bonus $1 \cdot \mathit{Ratio}^t$ to the agent if the total coverage rate is reached in 95%, and $0.5 \cdot \mathit{Ratio}^t$ when 90% coverage rate is realized.
- **Overlap Penalty:** The overlap penalty is applied to reduce repetitive exploration among agents so as to enhance cooperation capability. It is described as:

$$\begin{cases} -\Delta A^t_{overlap} \times 0.01,\ \mathit{Ratio}^t\ < 0.9 \\ -\Delta A^t_{overlap} \times 0.006, 0.9 \leq\ \mathit{Ratio}^t < 0.95 \\ 0,\ \mathit{Ratio}^t \geq 0.95 \end{cases}$$

  $\Delta A^t_{overlap} = A^t_{overlap} - A^{t-1}_{overlap}$ denotes the increment of the average overlapped explored area $A^t_{overlap}$ between each two agents at timestep t. In practical, the $A^t_{overlap_{k,u}} = \mathit{Area}^t_k \bigcap \mathit{Area}^{t-1}_u$ between agent k and agent u transforms into the sum of a one-hot map, where the grid on the map will be valued when its total value of two agents' explored probability is greater than 1.2.
- **Time Penalty:** For the sake of an efficient exploration, we propose the time penalty:

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

The linear combination of four parts is the final team-based reward. Note that all the explored and obstacle maps are in the scale of $5cm$ for each grid and the measurement unit of all the area is m^2.

Architecture CNN feature extractor is composed of 5 convolution layers as well as max pooling layers. Hyperparameters of these layers are listed in Table 1. Except the last layer, each layer is followed by a max pooling layer with kernel size 2.

The core part of MSP, Spatial-Teamformer, contains several blocks, each of which consists of two layers, i.e. an Individual Spatial Encoder and a Team Relation Encoder. Both Individual Spatial Encoder and Team Relation Encoder are self-attention layers with residual connection. We remark that the self-attention mechanism in MSP is exactly the same as transformer encoder in [4]. Hyperparameters of Spatial-Teamformer are listed in Table 2.

B Baselines

We implemented 6 classical planning-based methods, including 3 single-agent methods and 3 multi-agent baselines.

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

Table 1. CNN feature extractor hyperparameters

hyperparameters	value
# of attention heads	4
attention head dimension	32
attention hidden Size	128
# of Spatial-Teamformer blocks	2

Table 2. MSP hyperparameters

B.1 Single-agent Baselines

- **Nearest** [11] selects the nearest frontier as global goal [12] via breadth first search on the merged global map.
- **Utility** [6] chooses the frontier with the largest information gain [1].
- **RRT** [9] generates a collision-free random tree rooted at the agent's current location. After collecting enough tree nodes that lie on unexplored region, i.e. frontiers, RRT chooses the one with the highest utility $u(p) = IG(p) - N(p)$, where $IG(p)$ and $N(p)$ are respectively the normalized information gain and navigation cost of p. Pseudocode of RRT is shown in Algorithm 1. 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 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.

B.2 Multi-agent Baselines

- **APF** [14] first computes a potential field F based on explored occupancy map and current agent locations, and then follows the fastest descending direction of F to find a frontier as the global goal. Resistance force among agents is introduced in APF to avoid repetitive exploration. Pseudocode of APF is provided in Algorithm 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.

- **WMA-RRT** [7] WMA-RRT is a multi-agent variant of RRT. Pseudocode of WMA-RRT is provided in Algorithm 3. By maintaining a rooted tree together, agents share information to finish exploration. To impose cooperation using the shared tree, WMA-RRT uses a locking mechanism to avoid agents exploring same part of the tree and restricts agents to walking along the edge of the tree to ensure a strict system. Agents choose a node in the tree as a global goal and mark whether a subtree has been completedly searched. Although this is a multi-agent variant of RRT, we empirically found it perform much worse than RRT. We found this is because the locking mechanism actually restricts agent behaviors greatly and the algorithm itself is incompatible with active SLAM. For the former, agents are often forced not to explore large open areas which require multi-agent effort because locked by another agent. For the latter, WMA-RRT was originally designed for the case a ground-truth mapping is given and adding new nodes into the tree during exploration would cause mis-labeled completed subtree. Also, WMA-RRT do not perform value estimation over the nodes, making agents committed to branches that do not increase coverage much. Therefore, though WMA-RRT is guaranteed to reach fully coverage, it's inherently not suitable for the setting to maximize coverage ratio. As a comparison between RRT and WMA-RRT, WMA-RRT only utilizes RRT to expand tree while perform multi-agent planning using a strict tree-search procedure while RRT uses the random tree equipped with utility estimation to do planning. For detailed description of the algorithm, we encourage readers to check out [7].
- **Voronoi** [5] The voronoi-based method first partitions the map via voronoi partition and assigns components to agents so that each agent owns parts that are closest to it. Then each agent finds its own global goal by finding a frontier point with largest potential as in Utility within its own partition. In this way, duplicated exploration is be avoided. Pseudocode of Voronoi is provided in Algorithm 4.

Finally, we also evaluate the performance of random policies for references. To eliminate negative impact of visual blind area, the area within a distance of $2.5m$ to the agent is virtually marked as explored when choosing frontiers so that these baselines would output far enough global goals. The number of unexplored grids within a distance of $1.5m$ to a frontier p is defined as the information gain. All these baselines regenerate new global goals every 15 time steps, which is consistent with MSP. Case studies and failure modes of these methods are provided in Section F.3.

Algorithm 1 Rapid-exploring Random Tree (RRT)

```
Input : Map $M$ and agent location $loc$.
Output : Selected frontier goal
$NodeList \leftarrow \{loc\}, Targets \leftarrow \{\}$
$i \leftarrow 0$
while $i < T$ and $|Targets| < N_{target}$ do
  $i \leftarrow i + 1$
  $p \leftarrow$ a random point
  $s \leftarrow \arg\min_{u \in NodeList} ||u - p||_2$
  $t \leftarrow Steer(s, p, L)$
  if $No\_Collision(M, s, t)$ then
    if $t$ lies in unexplored area then
      $Targets \leftarrow Targets + \{t\}$
    else
      $NodeList \leftarrow NodeList + \{t\}$
    end if
  end if
end while
$C \leftarrow$ clusters of points in $Targets$.
$goal \leftarrow \arg\min_{c \in C} IG(c) - N(c)$
return $goal$
```

C Evaluation Metrics

We select 3 behavior statistics metric to show different characteristics of particular exploration methods.

- **Coverage**: *Coverage* represents the ratio of areas explored by the agents to the entire explorable space in the indoor scene at the end of the episode. Higher *Coverage* implies more effective exploration. A cell of $5cm \times 5cm$ is considered explored/covered when the 2D projection on the floor of some depth image in the exploration history covers this cell.
- **Steps**: *Steps* is the number of timesteps used by agents to achieve a coverage ratio of 90% within an episode. Fewer *Steps* implies faster exploration.
- **Mutual Overlap**: For effective collaboration, each agent should visit regions different from those explored by its teammates. We measure the average overlapping explored area over each pair of agents when the coverage ratio reaches 90%, which we call *Mutual Overlap. Mutual Overlap* denotes the normalized value of mutual overlap. Lower *Mutual Overlap* suggests better multi-agent coordination.

D Training Details

We adopt Multi-Agent Proximal Policy Optimization (MAPPO) [13], a multi-agent extension of PPO, to train MSP. The pseudocode of MAPPO is provided in Algorithm 5. Detailed hyper-parameters are listed in Table 3.

Algorithm 2 Artificial Potential Field (APF)

```
Input : Map $M$, number of agents $n$ and agent locations $loc_1 \ldots loc_n$.
Output : Selected goals for each agent
$P \leftarrow$ frontiers in $M$
$C \leftarrow$ clusters of frontiers $P$
$goals \leftarrow$ an empty list
for $i = 1 \rightarrow n$ do
    $F \leftarrow$ zero potential field
    // Compute Resistance force
    for $j = 1 \rightarrow n$ do
        for unoccupied grid $p \in M$ do
            if $j \neq i$ and $||p - loc_j||_2 < D$ then
                $F_p \leftarrow F_p + k_D \cdot (D - ||p - loc_j||_2)$
            end if
        end for
    end for
    for $c \in C$ do
        Run breadth-first search to compute distance map $dis$ starting from $c$
        $F \leftarrow F - dis^{-1} \cdot w_c$
    end for
    $u \leftarrow loc_i, cnt \leftarrow 0$
    while $u \notin M$ and $F_u$ is not a local minima and $cnt < T$ do
        $cnt \leftarrow cnt + 1$
        $F_u \leftarrow F_u + C_{repeat}$
        $u \leftarrow \arg\min_{v \in Neigh(u)} F_v$
    end while
    append $u$ to the end of $goals$
end for
return $goals$
```

Algorithm 3 Weighted Multi-Agent RRT

```
Find_Next_Point($a$):
if Node $a$ is a leaf node then
    return $a$
end if
for edge $e = (a, b) \in Child(a)$ in clock-wise order do
    if $e$ is not locked and subtree rooted at node $b$ is not completed then
        Lock edge $e$ for $LockTime$ timesteps
        return Find_Next_Point($b$)
    end if
end for
Mark subtree rooted at $a$ as completed
return Parent node of $a$

Main():
Reset the environment $env$
PHASE ← "Gather Stage"
Initialize rooted tree $T = \emptyset$
while $env$ is not done do
    if agents are close enough then
        $RootLoc$ ← mean coordination of all agents
        Initialize root node $Root = Node(RootLoc)$
        for all agents $a = 1 \to n$ do
            $T = T \cup \{(Root, Node(AgentLocation[a]))\}$
        end for
        PHASE ← "Exploration Stage"
    end if
    for all agents $a = 1 \to n$ do
        if PHASE = "Gather Stage" then
            $goal_a$ ← mean coordination of all agents
        else
            $goal_a$ ← Find_Next_Goal($AgentNode[a]$)
        end if
    end for
    Compute directional action via $goal_1, \cdots, goal_N$
    Move one step and receiver observations
    Update global merged map
    if PHASE = "Exploration Stage" then
        Add new nodes into $T$ given new global merged map using standard RRT
    end if
end while
```

Algorithm 4 Voronoi-based method

```
Input : Map $M$ and all agents' locations $loc_1, \cdots, loc_n$.
Output : Selected frontier goals for all agents
Partition the map $M$ via voronoi parition into triangle blocks $B = \{b_1, \cdots, b_m\}$
Initialize $P = [[], \cdots, []]$, i.e. $n$ empty lists
for all blocks $i = 1 \rightarrow m$ do
    $dis_1, dis_2, \cdots, dis_n \leftarrow$ distance of all agents to block $b_i$.
    $a \leftarrow \arg\min_a dis_a$
    $P_a \leftarrow P_a \cup \{b_i\}$
end for
$goal \leftarrow \emptyset$
for all agents $a = 1 \rightarrow n$  do
    $goal_a \leftarrow$ frontier point within $P_a$ that has largest information gain.
end for
return  $goal$
```

Algorithm 5 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
```

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
parallel environment threads	10
number of local steps	15

Table 3. MAPPO Hyperparameters

As for policy distillation, an expert network $\pi(g, x, y|s, \theta_t^{(i)})$ is trained, where g is the region head, (x, y) is the point head, s is the state, for each training scenes and team size of 2. After that, a student network is trained $\pi(g, x, y|s, \hat{\theta})$ via performing behavior cloning from these expert networks. The student network is trained in dagger style: for each episode, we first collect data using $\pi(g, x, y|s, \hat{\theta})$, and then run several updates to optimize the output of student using past experience. The objective function to minimize is sum of $L_g(\hat{\theta})$, which is the KL divergence between the student region distribution $\pi_g(s, \hat{\theta})$ and teacher region distribution $\pi_g(s, \theta_t^{(i)})$, and $L_{x,y}(\hat{\theta})$, which is the square error loss between point head of student and that of teacher. Policy distillation uses Adam optimizer with $2.5e-5$ learning rate.

E Experimental Setting

E.1 Datasets

We follow the dataset used in *Active Neural SLAM* (ANS) [2]. The original Gibson Challenge dataset [10], which could be used with Habitat Simulator, provides 72 training and 14 validation scenes. Note that Gibson testing set is not public but rather held on an online evaluation server for the PointGoal task, so the validation set is used as the testing instead of hyper-parameter tuning. We have made substantial efforts to check every single scene from the Gibson Challenge dataset and have to exclude a large portion of scenes not suitable for multi-agent exploration, including (i) scenes that have large disconnected regions; and (ii) scenes that have multiple floors (agents can not go upstairs) so that the agents are not possible to reach 90% coverage of the entire house. Note that disconnected region is not an issue for semantic or point navigation tasks.

We categorize the remaining scenes into 23 training scenes, including 9 small scenes, 9 middle scenes and 5 large scenes based on explorable area, as well as 10 testing scenes, which including 5 small scenes, 4 middle scenes and 1 large scene. Note that the validation set of the original Gibson Challenge dataset, i.e. the testing set, has 14 scenes, of which 8 scenes are eligible for our task, including 5 small scenes, 1 middle scene and 1 large scene. Since most scenes in the testing set are too small for multi-agent exploration task, we additionally add 3 middle scenes to the testing set. The common training paradigm for visual exploration is to randomly sample training scenes or team sizes at each training episode [3]. However, we empirically observe that different Habitat scenes and team sizes may lead to drastically different exploration difficulties. During training, gradients from different configurations may negatively impact each other. Hence, our solution is to train a separate policy on each map and use policy distillation to extract a meta policy to tackle this problem.

E.2 Assumption of Birth Position

We assume the agents has access to the **birth location** of each other while the locations during an episode are estimated using Neural SLAM module. The merged global map is fused using the estimated locations and the birth place.

E.3 Episode Length

First, we empirically found that as the number of agents grows, even random exploration can be particularly competitive (as shown in Table 6), which, we believe, is due to the limited explorable space of Gibson scenes. Hence, we only test up to 4 agents and argue that the 2-agent case is the most challenging. Regarding the *episode length*, it is estimated according to the number of timesteps when the strongest single-agent planning-based method, RRT, achieves 95% coverage. Note that if the horizon is too long (e.g., 1000, which is used in ANS), almost all the methods will have the same the final coverage rate. In addition, the *Mutual Overlap* and *Steps* metrics are all estimated before the agents reach a 90% coverage, which do not depend on the episode length. The choice of a higher re-planning frequency (e.g., re-plan per 15 timestep) is also due to a shorter episode horizon[4]. All the baselines use the same planning frequency.

F Additional Experiment Results

F.1 Fixed Team Size

Trained with Team size = 2 We first report the performance of MAANS and all baseline methods with a fixed team size of $N = 2$ agents on both 9 representative training scenes and unseen testing scenes in Table 4 and Table 5.

[4] All the agents will re-generate their personal long-term global goals using MSP in a synchronous manner every 15 timesteps while ANS [2] re-plans every 25 timesteps.

MAANS outperforms all the planning-based baselines with a clear margin in every evaluation metric, particularly the *Steps*, on both training and testing scenes. We can observe that *Steps* of testing scenes is overall fewer than training scenes since we use 9 middle scenes for training while the testing set has more small scenes. And as the scene size grows, the performance of planning-based methods degrades a lot. We directly apply the policies trained with $N = 2$ agents to the scenes of $N = 3, 4$ agents respectively. The zero-shot generalization performance of MAANS compared with all the baselines on 9 representative training scenes and testing scenes is shown in Table 6 and Table 7. We can observe that MAANS still outperforms planning-based methods on training scenes. And MAANS could even achieve comparable performance on testing scenes despite MAANS having neither seen the testing scenes nor the novel team sizes.

Sce.	Metrics	Random	Nearest	Utility	RRT	MAANS
Training Sce.	*Mut. Over.* ↓	0.66(0.01)	0.53(0.02)	0.68(0.01)	0.53(0.02)	**0.42(0.01)**
	Steps ↓	273.56(1.38)	246.79(3.90)	236.15(3.61)	199.59(3.27)	**158.55(2.25)**
	Coverage ↑	0.86(0.00)	0.91(0.01)	0.92(0.01)	0.96(0.00)	**0.97(0.00)**
Testing Sce.	*Mut. Over.* ↓	0.66(0.02)	0.58(0.01)	0.69(0.01)	0.57(0.02)	**0.54(0.02)**
	Steps ↓	193.83(2.80)	166.23(3.96)	161.28(2.32)	157.29(2.59)	**144.16(2.52)**
	Coverage ↑	0.93(0.00)	0.95(0.00)	0.95(0.00)	0.95(0.01)	**0.96(0.00)**

Table 4. Performance of MAANS and *single-agent baselines* with a fixed size of $N = 2$ agents on both training and testing scenes.

Sce.	Metrics	Random	APF	WMA-RRT	Voronoi.	MAANS
Training Sce.	*Mut. Over.* ↓	0.66(0.01)	0.61(0.01)	0.61(0.01)	0.44(0.01)	**0.42(0.01)**
	Steps ↓	273.56(1.38)	251.41(3.15)	268.20(2.24)	237.04(2.95)	**158.55(2.25)**
	Coverage ↑	0.86(0.00)	0.90(0.01)	0.87(0.01)	0.93(0.00)	**0.97(0.00)**
Testing Sce.	*Mut. Over.* ↓	0.66(0.02)	0.57(0.01)	0.64(0.01)	**0.51(0.01)**	0.54(0.02)
	Steps ↓	193.83(2.80)	181.18(4.17)	198.92(3.83)	156.68(3.21)	**144.16(2.52)**
	Coverage ↑	0.93(0.00)	0.93(0.01)	0.91(0.01)	**0.96(0.01)**	**0.96(0.00)**

Table 5. Performance of MAANS and *multi-agent baselines* with a fixed size of $N = 2$ agents on both training and testing scenes.

Trained with Team size = 3 Here we additionally report the performance of all the baseline methods and MAANS trained with a fixed team size of $N = 3$ agents on 9 representative training scenes and testing scenes in Table 8 and Table 9. Except for comparable performance to Voronoi on testing scenes, MAANS consistently outperforms other planning-based baselines in all metrics on both training scenes and testing scenes. When training with team size 3, MAANS also exhibits surprising zero-shot transfer ability to various team sizes on training and testing scenes as shown in Table 10 and Table 11. More concretely, MAANS

# Agent	Metrics	Random	Nearest	Utility	RRT	MAANS
			Training Scenes			
3	*Mut. Over.* ↓	0.54(0.01)	0.46(0.01)	0.58(0.00)	0.44(0.01)	**0.42(0.01)**
	Steps ↓	221.29(1.80)	188.58(2.02)	180.82(2.25)	155.13(3.26)	**127.88(1.91)**
	Coverage ↑	0.82(0.01)	0.91(0.00)	0.94(0.00)	0.95(0.01)	**0.97(0.00)**
4	*Mut. Over.* ↓	0.49(0.01)	0.43(0.01)	0.52(0.01)	**0.36(0.01)**	0.42(0.01)
	Steps ↓	163.11(0.77)	154.75(2.16)	151.30(3.03)	140.57(1.78)	**114.75(1.69)**
	Coverage ↑	0.87(0.00)	0.88(0.01)	0.91(0.01)	0.92(0.01)	**0.96(0.00)**
			Testing Scenes			
3	*Mut. Over.* ↓	0.55(0.01)	0.51(0.01)	0.59(0.01)	**0.45(0.01)**	0.53(0.01)
	Steps ↓	145.90(2.80)	131.04(3.53)	128.46(3.04)	128.33(1.66)	**122.48(2.22)**
	Coverage ↑	0.94(0.00)	0.95(0.00)	**0.96(0.00)**	0.95(0.01)	**0.96(0.00)**
4	*Mut. Over.* ↓	0.48(0.01)	0.46(0.01)	0.54(0.01)	**0.41(0.01)**	0.50(0.01)
	Steps ↓	116.94(1.61)	**108.23(1.24)**	110.14(1.93)	111.30(1.58)	109.07(2.02)
	Coverage ↑	0.93(0.01)	**0.94(0.00)**	**0.94(0.01)**	0.93(0.01)	**0.94(0.00)**

Table 6. Zero-shot generalization performance of MAANS trained with a fixed team size $N = 2$ and *single-agent methods* to novel team sizes on training and testing scenes.

# Agent	Metrics	Random	APF	WMA-RRT	Voronoi	MAANS
			Training Scenes			
3	*Mut. Over.* ↓	0.54(0.01)	0.45(0.01)	0.54(0.01)	**0.37(0.01)**	0.42(0.01)
	Steps ↓	221.29(1.80)	207.20(2.41)	210.01(2.68)	180.27(2.51)	**127.88(1.91)**
	Coverage ↑	0.82(0.01)	0.87(0.01)	0.87(0.01)	0.95(0.00)	**0.97(0.00)**
4	*Mut. Over.* ↓	0.49(0.01)	0.35(0.01)	0.49(0.01)	**0.34(0.01)**	0.42(0.01)
	Steps ↓	163.11(0.77)	170.59(1.06)	168.07(1.41)	147.01(2.38)	**114.75(1.69)**
	Coverage ↑	0.87(0.00)	0.79(0.01)	0.82(0.01)	0.93(0.00)	**0.96(0.00)**
			Testing Scenes			
3	*Mut. Over.* ↓	0.55(0.01)	**0.40(0.01)**	0.56(0.01)	0.43(0.01)	0.53(0.01)
	Steps ↓	145.90(2.80)	152.62(3.71)	161.59(3.60)	**119.98(2.31)**	122.48(2.22)
	Coverage ↑	0.94(0.00)	0.92(0.01)	0.89(0.02)	**0.96(0.00)**	**0.96(0.00)**
4	*Mut. Over.* ↓	0.48(0.01)	**0.30(0.01)**	0.52(0.01)	0.39(0.01)	0.50(0.01)
	Steps ↓	116.94(1.61)	133.68(1.35)	136.88(3.08)	**101.90(2.36)**	109.07(2.02)
	Coverage ↑	0.93(0.01)	0.88(0.01)	0.84(0.02)	**0.95(0.00)**	0.94(0.00)

Table 7. Zero-shot generalization performance of MAANS trained with a fixed team size $N = 2$ and *multi-agent methods* to novel team sizes on training and testing scenes.

trained with a fixed team size $N = 3$ shows much better performance than all planning-based methods on training scenes and comparable performance on testing scenes. Besides, we can observe that MAANS trained with a fixed team size $N = 2$ and MAANS trained with a fixed team size $N = 3$ are better than each other with corresponding training team size. MAANS trained with a fixed team size $N = 3$ performs better in 4-agent case, showing 0.07 less $MutualOverlap$ and 8 fewer *Steps* on training scenes and 0.04 less $MutualOverlap$ and 6 fewer *Steps* on testing scenes.

Sce.	Metrics	Random	Nearest	Utility	RRT	MAANS
Training Sce.	*Mut. Over.* ↓	0.54(0.01)	0.46(0.01)	0.58(0.00)	0.44(0.01)	**0.33(0.01)**
	Steps ↓	221.29(1.80)	188.58(2.02)	180.82(2.25)	155.13(3.26)	**121.99(1.91)**
	Coverage ↑	0.82(0.01)	0.91(0.00)	0.94(0.00)	0.95(0.01)	**0.97(0.00)**
Testing Sce.	*Mut. Over.* ↓	0.55(0.01)	0.51(0.01)	0.59(0.01)	**0.45(0.01)**	0.48(0.01)
	Steps ↓	145.90(2.80)	131.04(3.53)	128.46(3.04)	128.33(1.66)	**121.62(1.96)**
	Coverage ↑	0.94(0.00)	0.95(0.00)	**0.96(0.00)**	0.95(0.01)	**0.96(0.00)**

Table 8. Performance of MAANS and *single-agent baselines* with a fixed size of $N = 3$ agents on both training and testing scenes.

Sce.	Metrics	Random	APF	WMA-RRT	Voronoi	MAANS
Training Sce.	*Mut. Over.* ↓	0.54(0.01)	0.45(0.01)	0.54(0.01)	0.37(0.01)	**0.33(0.01)**
	Steps ↓	221.29(1.80)	207.20(2.41)	210.01(2.68))	180.27(2.51)	**121.99(1.91)**
	Coverage ↑	0.82(0.01)	0.87(0.01)	0.87(0.01)	0.95(0.00)	**0.97(0.00)**
Testing Sce.	*Mut. Over.* ↓	0.55(0.01)	0.40(0.01)	0.56(0.01)	**0.43(0.01)**	0.48(0.01)
	Steps ↓	145.90(2.80)	152.62(3.71)	161.59(3.60)	**119.98(2.31)**	121.62(1.96)
	Coverage ↑	0.94(0.00)	0.92(0.01)	0.89(0.02)	**0.96(0.00)**	**0.96(0.00)**

Table 9. Performance of MAANS and *multi-agent baselines* with a fixed size of $N = 3$ agents on both training and testing scenes.

F.2 Varying Team Size

We further consider the setting where the team size varies within an episode. We summarize the zero-shot generalization performance of MAANS compared with the planning-based baselines on training scenes in Table 12 and Table 13. We use "$N_1 \Rightarrow N_2$" to denote that each episode starts with N_1 agents and the team size immediately switches to N_2 after 90 timesteps. More concretely, in an episode, max(N_1, N_2) agents are set in the beginning, and N_2-N_1 agents are unable to move until timesteps 90 reaches in the increased team size scenarios. The setting is reversed in the decreased ones. We remark that MAANS trained by fixed team size $N = 2$ or $N = 3$ is separately presented in experiments, which is called MAANS(N=2) or MAANS(N=3).

# Agent	Metrics	Random	Nearest	Utility	RRT	MAANS
			Training Scenes			
2	*Mut. Over.* ↓	0.66(0.01)	0.53(0.02)	0.68(0.01)	0.53(0.02)	**0.33(0.01)**
	Steps ↓	273.56(1.38)	246.79(3.90)	236.15(3.61)	199.59(3.27)	**167.24(2.12)**
	Coverage ↑	0.86(0.00)	0.91(0.01)	0.92(0.01)	0.96(0.00)	**0.96(0.00)**
4	*Mut. Over.* ↓	0.49(0.01)	0.43(0.01)	0.52(0.01)	0.36(0.01)	**0.34(0.01)**
	Steps ↓	163.11(0.77)	154.75(2.16)	151.30(3.03)	140.57(1.78)	**106.12(2.19)**
	Coverage ↑	0.87(0.00)	0.88(0.01)	0.91(0.01)	0.92(0.01)	**0.96(0.00)**
			Testing Scenes			
2	*Mut. Over.* ↓	0.66(0.02)	0.58(0.01)	0.69(0.01)	0.57(0.02)	**0.52(0.01)**
	Steps ↓	193.83(2.80)	166.23(3.96)	161.28(2.32)	157.29(2.59)	**154.95(2.95)**
	Coverage ↑	0.93(0.00)	0.95(0.00)	0.95(0.00)	0.95(0.01)	**0.96(0.00)**
4	*Mut. Over.* ↓	0.48(0.01)	0.46(0.01)	0.54(0.01)	**0.41(0.01)**	0.46(0.01)
	Steps ↓	116.94(1.61)	108.23(1.24)	110.14(1.93)	111.30(1.58)	**103.52(1.98)**
	Coverage ↑	0.93(0.01)	0.94(0.00)	0.94(0.01)	0.93(0.01)	**0.95(0.00)**

Table 10. Zero-shot generalization performance of MAANS trained with a fixed team size $N = 3$ and *single-agent methods* to novel team sizes on training and testing scenes.

# Agent	Metrics	Random	APF	WMA-RRT	Voronoi	MAANS
			Training Scenes			
2	*Mut. Over.* ↓	0.66(0.01)	0.61(0.01)	0.61(0.01)	0.44(0.01)	**0.33(0.01)**
	Steps ↓	273.56(1.38)	251.41(3.15)	268.20(2.24)	237.04(2.95)	**167.24(2.12)**
	Coverage ↑	0.86(0.00)	0.90(0.01)	0.87(0.01)	0.93(0.00)	**0.96(0.00)**
4	*Mut. Over.* ↓	0.49(0.01)	0.35(0.01)	0.49(0.01)	**0.34(0.01)**	**0.34(0.01)**
	Steps ↓	163.11(0.77)	170.59(1.06)	168.07(1.41)	147.01(2.38)	**106.12(2.19)**
	Coverage ↑	0.87(0.00)	0.79(0.01)	0.82(0.01)	0.93(0.00)	**0.96(0.00)**
			Testing Scenes			
2	*Mut. Over.* ↓	0.66(0.02)	0.57(0.01)	0.64(0.01)	**0.51(0.02)**	0.52(0.01)
	Steps ↓	193.83(2.80)	181.18(4.17)	198.92(3.83)	156.68(3.21)	**154.95(2.95)**
	Coverage ↑	0.93(0.00)	0.93(0.01)	0.91(0.01)	**0.96(0.01)**	**0.96(0.00)**
4	*Mut. Over.* ↓	0.48(0.01)	**0.30(0.01)**	0.52(0.01)	0.39(0.01)	0.46(0.01)
	Steps ↓	116.94(1.61)	133.68(1.35)	136.88(3.08)	**101.90(2.36)**	103.52(1.98)
	Coverage ↑	0.93(0.01)	0.88(0.01)	0.84(0.02)	**0.95(0.00)**	**0.95(0.00)**

Table 11. Zero-shot generalization performance of MAANS trained with a fixed team size $N = 3$ and *multi-agent methods* to novel team sizes on training and testing scenes.

In scenarios where the team size increases, though RRT still performs the best among the planning-based baselines, MAANS outperforms RRT with a clear margin for 25 fewer *Steps* in $2 \Rightarrow 4$ setting and 35 fewer *Steps* in others. While as the team size decreases, the performance between MAANS and classical methods varies more widely, which shows that MAANS has a 35 fewer *Steps* in the comparison of the best baseline, RRT. Besides, MAANS has the best result in the metrics of mutual overlap ratio and coverage. We consider the case is more challenging where the team size decreases and the share information gain becomes less shapely, therefore the baselines could not adjust the strategy immediately.

When we compare MAANS$_{(\text{N}=2)}$ with MAANS$_{(\text{N}=3)}$, MAANS$_{(\text{N}=3)}$ has a comparable performance with a lower *Mutual Overlap* ratio to the other. It indicates that training with a fixed team size $N = 3$ helps MAANS grasp the capability of cooperation better so that the strategy is more stable and inflexible in a varying team size situation.

We summarize the zero-shot generalization performance of MAANS compared with the planning-based baselines on testing scenes in Table 14 and Table 15. In cases where the team size increases, MAANS still produces substantially better performances, especially *Steps*, which suggests that MAANS has the capability to adaptively adjust its strategy. Regarding the cases where the team size decreases, MAANS shows slightly worse performance than the best single-agent baseline, RRT, and the best multi-agent baseline, Voronoi. We remark that decreasing the team size is particularly challenging since the absence of some agents might immediately leave a large part of the house unexplored and consequently, the team should immediately update their original plan with drastically different goal assignments.

# Agent	Metrics	Random	Nearest	Utility	RRT	MAANS (N=2)	MAANS (N=3)
Increase Number of Agents							
$2 \Rightarrow 3$	*Mut. Over.* ↓	0.54(0.01)	0.43(0.01)	0.56(0.01)	0.36(0.01)	0.30(0.01)	**0.25(0.01)**
	Steps ↓	223.63(2.16)	211.73(1.96)	210.88(1.30)	185.94(1.83)	148.82(2.01)	**146.34(1.76)**
	Coverage ↑	0.86(0.01)	0.89(0.01)	0.90(0.00)	0.94(0.00)	**0.96(0.00)**	**0.96(0.00)**
$2 \Rightarrow 4$	*Mut. Over.* ↓	0.42(0.01)	0.37(0.01)	0.47(0.00)	0.31(0.01)	0.26(0.01)	**0.22(0.01)**
	Steps ↓	175.89(0.64)	174.06(0.72)	174.55(0.69)	165.43(0.97)	142.96(1.24)	**138.22(1.36)**
	Coverage ↑	0.82(0.01)	0.81(0.01)	0.81(0.00)	0.88(0.01)	**0.94(0.00)**	**0.94(0.00)**
$3 \Rightarrow 4$	*Mut. Over.* ↓	0.45(0.01)	0.38(0.01)	0.49(0.01)	0.26(0.01)	0.29(0.01)	**0.24(0.01)**
	Steps ↓	170.90(1.19)	165.85(0.80)	165.82(0.72)	155.15(2.18)	125.26(1.46)	**119.03(1.36)**
	Coverage ↑	0.85(0.01)	0.85(0.00)	0.87(0.01)	0.90(0.01)	0.95(0.00)	**0.96(0.00)**
Decrease Number of Agents							
$3 \Rightarrow 2$	*Mut. Over.* ↓	0.48(0.01)	0.39(0.01)	0.48(0.01)	0.35(0.01)	0.41(0.01)	**0.33(0.01)**
	Steps ↓	225.51(2.57)	213.16(1.65)	209.87(1.75)	187.93(1.98)	145.14(2.83)	**142.09(1.98)**
	Coverage ↑	0.84(0.01)	0.86(0.01)	0.88(0.00)	0.91(0.00)	**0.95(0.00)**	**0.95(0.00)**
$4 \Rightarrow 2$	*Mut. Over.* ↓	0.41(0.01)	0.36(0.01)	0.42(0.01)	**0.32(0.01)**	0.40(0.01)	0.33(0.01)
	Steps ↓	173.36(1.08)	168.89(1.12)	167.54(1.30)	157.40(2.56)	127.35(2.08)	**118.93(2.30)**
	Coverage ↑	0.81(0.00)	0.82(0.01)	0.83(0.01)	0.86(0.01)	**0.93(0.00)**	**0.93(0.00)**
$4 \Rightarrow 3$	*Mut. Over.* ↓	0.44(0.01)	0.39(0.01)	0.47(0.00)	**0.33(0.01)**	0.41(0.01)	**0.33(0.01)**
	Steps ↓	168.32(1.24)	161.87(1.08)	159.96(1.48)	147.61(1.78)	119.59(2.31)	**111.24(1.54)**
	Coverage ↑	0.85(0.00)	0.85(0.00)	0.88(0.01)	0.90(0.01)	**0.95(0.00)**	**0.95(0.00)**

Table 12. Performance of MAANS and *single-agent baselines* with a varying team size on training scenes.

# Agent	Metrics	Random	APF	WMA-RRT	Voronoi	MAANS (N=2)	MAANS (N=3)
Increase Number of Agents							
$2 \Rightarrow 3$	*Mut. Over.* ↓	0.54(0.01)	0.42(0.00)	0.49(0.00)	0.35(0.01)	0.30(0.01)	**0.25(0.01)**
	Steps ↓	223.63(2.16)	225.35(0.97)	221.85(0.00)	200.91(2.32)	148.82(2.01)	**146.34(1.76)**
	Coverage ↑	0.86(0.01)	0.82(0.01)	0.85(0.00)	0.92(0.00)	**0.96(0.00)**	**0.96(0.00)**
$2 \Rightarrow 4$	*Mut. Over.* ↓	0.42(0.01)	0.31(0.01)	0.44(0.01)	0.30(0.01)	0.26(0.01)	**0.22(0.01)**
	Steps ↓	175.89(0.64)	178.77(0.13)	178.91(0.33)	170.50(0.88)	142.96(1.24)	**138.22(1.36)**
	Coverage ↑	0.82(0.01)	0.68(0.01)	0.67(0.01)	0.85(0.01)	**0.94(0.00)**	**0.94(0.00)**
$3 \Rightarrow 4$	*Mut. Over.* ↓	0.45(0.01)	0.32(0.01)	0.45(0.00)	0.31(0.01)	0.29(0.01)	**0.24(0.01)**
	Steps ↓	170.90(1.19)	175.35(0.56)	173.64(0.41)	159.23(1.50)	125.26(1.46)	**119.03(1.36)**
	Coverage ↑	0.85(0.01)	0.74(0.01)	0.79(0.00)	0.90(0.00)	0.95(0.00)	**0.96(0.00)**
Decrease Number of Agents							
$3 \Rightarrow 2$	*Mut. Over.* ↓	0.48(0.01)	0.38(0.01)	0.44(0.00)	**0.33(0.01)**	0.41(0.01)	**0.33(0.01)**
	Steps ↓	225.51(2.57)	225.51(1.32)	226.14(0.00)	206.94(2.50)	145.14(2.83)	**142.09(1.98)**
	Coverage ↑	0.84(0.01)	0.78(0.01)	0.81(0.00)	0.89(0.01)	**0.95(0.00)**	**0.95(0.00)**
$4 \Rightarrow 2$	*Mut. Over.* ↓	0.41(0.01)	0.32(0.01)	0.40(0.00)	**0.30(0.01)**	0.40(0.01)	0.33(0.01)
	Steps ↓	173.36(1.08)	176.79(0.65)	176.38(0.00)	165.23(2.80)	127.35(2.08)	**118.93(2.30)**
	Coverage ↑	0.81(0.00)	0.68(0.01)	0.73(0.00)	0.83(0.01)	**0.93(0.00)**	**0.93(0.00)**
$4 \Rightarrow 3$	*Mut. Over.* ↓	0.44(0.01)	0.33(0.01)	0.44(0.00)	**0.32(0.01)**	0.41(0.01)	0.33(0.01)
	Steps ↓	168.32(1.24)	173.94(0.81)	171.85(0.00)	155.65(2.79)	119.59(2.31)	**111.24(1.54)**
	Coverage ↑	0.85(0.00)	0.73(0.01)	0.78(0.00)	0.89(0.01)	**0.95(0.00)**	**0.95(0.00)**

Table 13. Performance of MAANS and *multi-agent baselines* with a varying team size on training scenes.

# Agent	Metrics	Random	Nearest	Utility	RRT	MAANS (N=2)	MAANS (N=3)
Increase Number of Agents							
2 ⇒ 3	*Mut. Over.* ↓	0.52(0.01)	0.47(0.01)	0.55(0.02)	0.43(0.01)	0.46(0.01)	**0.41(0.01)**
	Steps ↓	159.85(3.60)	144.60(3.45)	143.14(1.93)	136.42(2.41)	134.11(2.88)	**131.96(1.80)**
	Coverage ↑	0.93(0.01)	0.95(0.01)	0.95(0.00)	**0.96(0.01)**	**0.96(0.00)**	**0.96(0.00)**
2 ⇒ 4	*Mut. Over.* ↓	0.43(0.01)	0.40(0.01)	0.47(0.01)	0.38(0.01)	0.43(0.01)	**0.37(0.01)**
	Steps ↓	134.57(0.63)	129.13(1.86)	126.92(1.67)	122.42(1.85)	122.09(1.99)	**116.66(4.51)**
	Coverage ↑	0.90(0.00)	0.91(0.01)	0.92(0.01)	0.92(0.00)	**0.93(0.01)**	**0.93(0.01)**
3 ⇒ 4	*Mut. Over.* ↓	0.46(0.01)	0.42(0.01)	0.49(0.01)	**0.37(0.01)**	0.45(0.01)	0.39(0.00)
	Steps ↓	125.08(1.35)	116.96(1.42)	115.44(3.38)	119.02(1.32)	114.84(1.56)	**110.04(0.54)**
	Coverage ↑	0.92(0.01)	0.93(0.00)	0.93(0.01)	0.92(0.01)	**0.94(0.00)**	**0.94(0.00)**
Decrease Number of Agents							
3 ⇒ 2	*Mut. Over.* ↓	0.45(0.01)	0.41(0.01)	0.46(0.01)	**0.39(0.01)**	0.43(0.01)	0.40(0.00)
	Steps ↓	167.98(2.06)	149.41(2.59)	146.73(3.44)	**139.52(3.74)**	145.43(3.44)	146.93(2.93)
	Coverage ↑	0.91(0.00)	**0.94(0.00)**	0.93(0.01)	**0.94(0.01)**	**0.94(0.01)**	**0.94(0.00)**
4 ⇒ 2	*Mut. Over.* ↓	0.36(0.01)	0.33(0.01)	0.37(0.01)	**0.31(0.00)**	0.37(0.01)	0.34(0.00)
	Steps ↓	140.37(1.80)	128.11(1.72)	127.40(0.49)	**125.84(0.83)**	129.73(2.73)	127.30(1.55)
	Coverage ↑	0.88(0.01)	**0.90(0.01)**	**0.90(0.01)**	**0.90(0.00)**	**0.90(0.01)**	**0.90(0.00)**
4 ⇒ 3	*Mut. Over.* ↓	0.43(0.01)	0.39(0.00)	0.44(0.01)	**0.35(0.01)**	0.43(0.01)	0.39(0.01)
	Steps ↓	127.46(1.64)	117.44(0.89)	114.56(2.28)	119.70(1.88)	116.74(1.38)	**112.72(2.13)**
	Coverage ↑	0.91(0.01)	**0.93(0.00)**	**0.93(0.01)**	0.91(0.00)	**0.93(0.01)**	**0.93(0.00)**

Table 14. Performance of MAANS and *single-agent baselines* with a varying team size on testing scenes.

# Agent	Metrics	Random	APF	WMA-RRT	Voronoi	MAANS (N=2)	MAANS (N=3)
Increase Number of Agents							
2 ⇒ 3	*Mut. Over.* ↓	0.52(0.01)	0.38(0.02)	0.44(0.01)	**0.32(0.02)**	0.46(0.01)	0.41(0.01)
	Steps ↓	159.85(3.60)	167.87(2.85)	176.89(5.70)	148.12(6.69)	134.11(2.88)	**131.96(1.80)**
	Coverage ↑	0.93(0.01)	0.89(0.01)	0.88(0.01)	0.92(0.05)	**0.96(0.00)**	**0.96(0.00)**
2 ⇒ 4	*Mut. Over.* ↓	0.43(0.01)	0.26(0.01)	0.33(0.01)	**0.24(0.01)**	0.43(0.01)	0.37(0.01)
	Steps ↓	134.57(0.63)	148.48(1.01)	157.48(1.50)	130.49(1.81)	122.09(1.99)	**116.66(4.51)**
	Coverage ↑	0.90(0.00)	0.78(0.01)	0.69(0.02)	0.90(0.01)	**0.93(0.01)**	**0.93(0.01)**
3 ⇒ 4	*Mut. Over.* ↓	0.46(0.01)	**0.28(0.01)**	0.40(0.00)	0.30(0.00)	0.45(0.01)	0.39(0.00)
	Steps ↓	125.08(1.35)	139.82(1.77)	141.80(1.53)	114.99(1.04)	114.84(1.56)	**110.04(0.54)**
	Coverage ↑	0.92(0.01)	0.85(0.01)	0.83(0.01)	**0.94(0.00)**	**0.94(0.00)**	**0.94(0.00)**
Decrease Number of Agents							
3 ⇒ 2	*Mut. Over.* ↓	0.45(0.01)	**0.33(0.01)**	0.51(0.01)	0.42(0.01)	0.43(0.01)	0.40(0.00)
	Steps ↓	167.98(2.06)	178.43(2.25)	171.87(2.53)	**133.77(2.83)**	145.43(3.44)	146.93(2.93)
	Coverage ↑	0.91(0.00)	0.86(0.01)	0.87(0.01)	**0.95(0.01)**	0.94(0.01)	0.94(0.00)
4 ⇒ 2	*Mut. Over.* ↓	0.36(0.01)	**0.23(0.00)**	0.47(0.01)	0.38(0.00)	0.37(0.01)	0.34(0.00)
	Steps ↓	140.37(1.80)	151.69(1.60)	144.66(1.46)	**115.97(1.93)**	129.73(2.73)	127.30(1.55)
	Coverage ↑	0.88(0.01)	0.78(0.01)	0.81(0.01)	**0.93(0.00)**	0.90(0.01)	0.90(0.00)
4 ⇒ 3	*Mut. Over.* ↓	0.43(0.01)	**0.27(0.01)**	0.48(0.00)	0.39(0.01)	0.43(0.01)	0.39(0.01)
	Steps ↓	127.46(1.64)	142.86(2.31)	140.68(0.17)	**106.92(2.78)**	116.74(1.38)	112.72(2.13)
	Coverage ↑	0.91(0.01)	0.84(0.01)	0.82(0.00)	**0.94(0.01)**	0.93(0.01)	0.93(0.00)

Table 15. Performance of MAANS and *multi-agent baselines* with a varying team size on testing scenes.

F.3 Case Studies

We implemented several planning-based methods as baselines, including Nearest, Utility, APF, RRT and additionally a voronoi-based method plus a multi-agent variant of RRT, WMA-RRT. While these methods are respectively tested under environments with ideal assumptions, we empirically found some of them do not work well under our setting with realistic perception and noises.

- **Sensitive to realistic noise.** Nearest, Utility, APF and the voronoi-based method all apply cell-level goal searching. We empirically found that they share a same failure mode, that is, trying to approach a cell that is wrongly estimated as explorable, even when the mapping only has minor cell-level error. The planning-based baseline RRT do not perform cell-level planning but in a geometric way, thus it's robust to mapping noise.
- **Naive estimation of future value.** Both Utility and RRT select candidate frontier with highest utility, which is computed as the number of unexplored cells within a small distance. Such estimation of future values is very short-sighted. In contrast, TANS, by using neural network, could perform more complicated inference about utility of exploring one part of the map.
- **Strict restriction over robot behaviors.** Among the planning-based baselines, both APF and WMA-RRT adopt a formally-designed cooperation system. However their cooperation schemes both imposes great restriction to agents' behaviors. APF includes resistance force among agents to avoid duplicated exploration. In WMA-RRT, agents share a same tree and follow a locking-and-searching method to do cooperation. However, in cases where it's better for agents to go through the same place simultaneously, which is pretty common, the resistance force in APF and the locking mechanism in WMA-RRT prevent agents from the optimal strategy. Meanwhile, WMA-RRT restricts agents to follow paths in the shared tree, losing the benefit of accidental coverage brought by random search.

  We provide case studies with corresponding graphics in Fig. 1-6. The red angles indicate the agents and their direction. The blue points indicate the global goals selected by the agents.

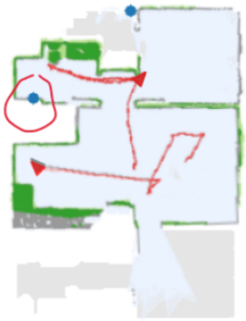

Fig. 1. Sensitive to Realistic Noise. Utility, Nearest, APF and the voronoi-based method performs cell-level goal searching, the above picture demonstrates a case when the agent selects a cell that is estimated as "unexplored" because of mapping error.

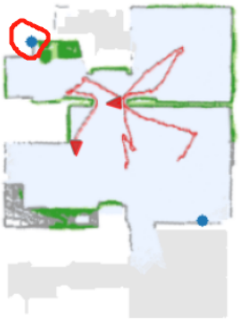

Fig. 2. Faulty Estimation of Value. In the above situation, Utility selects a cell at the corner of the room as a global goal since it's nearby space is unexplored. However, with prior knowledge about building structures, one should infer that such point should not be chosen.

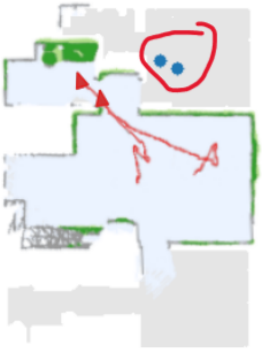

Fig. 3. Poor Cooperation. RRT performs planning independently, it's possible that agents choose close frontiers as goals, leading to duplicated exploration.

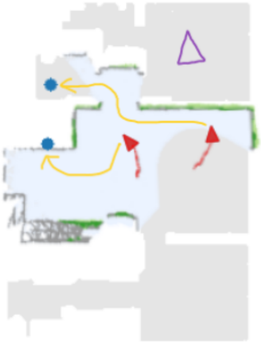

Fig. 4. APF Resistance Force. APF achieves cooperation by introducing resistance force among agents. In above situation, the yellow lines demonstrate agents' paths to their corresponding global goal. Due to the resistance force, the agent on the left is forced to choose a sub-optimal global goal instead of frontiers in the more promising part indicated by the purple triangle.

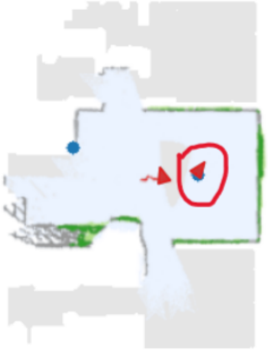

Fig. 5. WMA-RRT Locking Mechanism. In WMA-RRT, agents cooperatively maintain a tree and use a locking mechanism to avoid duplicated exploration. The above picture shows a case where agent on the right has to stay where it is since the path is locked by another agent.

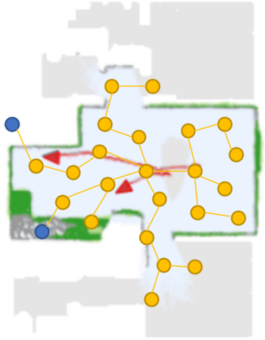

Fig. 6. Incompatible with Active Mapping. Specially, WMA-RRT algorithm is inherently incompatible with the active mapping process. When new cells are classified as obstacles, some nodes and edges in the tree would become invalid and WMA-RRT could not adapt to such change. In the above picture, agents still try to approach selected points even the tree edges and nodes are no longer valid.

References

1. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Transactions on robotics **21**(3), 376–386 (2005)
2. Chaplot, D.S., Gandhi, D., Gupta, S., Gupta, A., Salakhutdinov, R.: Learning to explore using active neural slam. In: International Conference on Learning Representations. ICLR (2020)
3. Chen, T., Gupta, S., Gupta, A.: Learning exploration policies for navigation. In: International Conference on Learning Representations. ICLR (2019)
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
5. Hu, J., Niu, H., Carrasco, J., Lennox, B., Arvin, F.: Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. IEEE Transactions on Vehicular Technology **69**(12), 14413–14423 (2020)
6. Juliá, M., Gil, A., Reinoso, O.: A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. Autonomous Robots **33**(4), 427–444 (2012)
7. Nazif, A.N., Davoodi, A., Pasquier, P.: Multi-agent area coverage using a single query roadmap: A swarm intelligence approach. In: Advances in practical multi-agent systems, pp. 95–112. Springer (2010)
8. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proceedings of the National Academy of Sciences **93**(4), 1591–1595 (1996)
9. Umari, H., Mukhopadhyay, S.: 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). https://doi.org/10.1109/IROS.2017.8202319
10. Xia, F., R. Zamir, A., He, Z., Sax, A., Malik, J., Savarese, S.: Gibson Env: real-world perception for embodied agents. In: Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on. IEEE (2018)
11. Yamauchi, B.: 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)
12. Yamauchi, B.: 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)
13. Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of mappo in cooperative, multi-agent games. arXiv preprint arXiv:2103.01955 (2021)
14. Yu, J., Tong, J., Xu, Y., Xu, Z., Dong, H., Yang, T., Wang, Y.: 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) (2021)