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

Abstract. We tackle the problem of cooperative visual exploration where multiple agents need to jointly explore unseen regions as fast as possible based on visual signals. Classical planning-based methods often suffer from expensive computation overhead at each step and a limited expressiveness of complex cooperation strategy. By contrast, reinforcement learning (RL) has recently become a popular paradigm for tackling this challenge due to its modeling capability of arbitrarily complex strategies and minimal inference overhead. In this paper, we propose a novel RL-based multi-agent planning module, *Multi-agent Spatial Planner* (MSP). MSP leverages a transformer-based architecture, *Spatial-TeamFormer*, which effectively captures spatial relations and intra-agent interactions via hierarchical spatial self-attentions. In addition, we also implement a few multi-agent enhancements to process local information from each agent for an aligned spatial representation and more precise planning. Finally, we perform policy distillation to extract a meta policy to significantly improve the generalization capability of final policy. We call this overall solution, *Multi-Agent Active Neural SLAM* (MAANS). MAANS substantially outperforms classical planning-based baselines for the first time in a photo-realistic 3D simulator, Habitat. Code and videos can be found at https://sites.google.com/view/maans.

Keywords: Multi-agent Reinforcement Learning, Visual Exploration

1 Introduction

Visual exploration [41] is an important task for building intelligent embodied agents and has been served as a fundamental building block for a wide range of

* This research is supported by NSFC (U20A20334, U19B2019 and M-0248), Tsinghua-Meituan Joint Institute for Digital Life, Tsinghua EE Independent Research Project, Beijing National Research Center for Information Science and Technology (BNRist), Beijing Innovation Center for Future Chips and 2030 Innovation Megaprojects of China (Programme on New Generation Artificial Intelligence) Grant No. 2021AAA0150000.

applications, such as scene reconstruction [1, 21], autonomous driving [3], disaster rescue [26] and planetary exploration [50]. In this paper, we consider a multi-agent exploration problem, where multiple homogeneous robots simultaneously explore an unknown spatial region via visual and sensory signals in a cooperative fashion. The existence of multiple agents enables complex cooperation strategies to effectively distribute the workload among different agents, which could lead to remarkably higher exploration efficiency than the single-agent counterpart.

Planning-based solutions have been widely adopted for navigation problems for both single-agent and multi-agent scenarios [4, 45, 53]. Planning-based methods require little training and can be directly applied to different scenarios. However, these methods often suffer from limited expressiveness capability on coordination strategies, require non-trivial hyper-parameter tuning for each test scenario, and are particularly time-consuming due to repeated re-planning at each decision step. By contrast, reinforcement learning (RL) has been promising solution for a wide range of decision-making problems [28, 33], including various visual navigation tasks [6, 9, 45]. Once a policy is well trained by an RL algorithm, the robot can capture arbitrarily complex strategies and produce real-time decisions with efficient inference computation (i.e., a single forward-pass of neural network).

However, training effective RL policies can be particularly challenging. This includes two folds: (1) learning a cooperative strategy over multiple agents in an end-to-end manner becomes substantially harder thanks to an exponentially large action space and observation space when tackling the exploration task based on visual signals; (2) RL policies often suffer from poor generalization ability to different scenarios or team sizes compared with classical planning-based approaches. Hence, most RL-based visual exploration methods focus on the single-agent case [6, 9, 45] or only consider a relatively simplified multi-agent setting (like maze or grid world [55]) of a fixed number of agents [30].

In this work, we develop *Multi-Agent Active Neural SLAM* (MAANS), the first RL-based solution for cooperative multi-agent exploration that substantially outperforms classical planning-based methods in a photo-realistic physical simulator, Habitat [46]. In MAANS, an agent consists of 4 components, a neural SLAM module, a planning-based local planner, a local policy for control, and the most critical one, a novel *Multi-agent Spatial Planner (MSP)*. which is an RL-trained planning module that can capture complex intra-agent interactions via a self-attention-based architecture, *Spatial-TeamFormer*, and produce effective navigation targets for a varying number of agents.

We also implement a map refiner to align the spatial representation of each agent's local map, and a map merger, which enables the local planner to perform more precise sub-goal generation over a manually combined approximate 2D map. Finally, instead of directly running multi-task RL over all the training scenes, we first train a single policy on each individual scene and then use policy distillation to extract a meta policy, leading to a much improved generalization capability. We compare MAANS with a collection of classical planning-based methods and RL-based variants. Empirical results show that MAANS has a 20.56% and 7.99% higher exploration efficiency on training and testing scenes than the best planning-based competitor. The learned policy can further generalize to novel team sizes in a zero-shot manner as well.

2 Related Work

2.1 Visual Exploration

In classical visual exploration solutions, a search-based planning algorithm could be adopted to generate valid exploration trajectories. Representative variants include frontier-based methods [53, 62, 65], which always choose navigation targets from the explored region, and sampling-based methods [27], which generate goals via a stochastic process. In addition to the expensive search computation for planning, these methods do not involve learning and thus have limited representation capabilities for particularly challenging scenarios. Hence, RL-based methods have been increasingly popular for their training flexibility and strong expressiveness power. Early methods simply train navigation policies in a purely end-to-end fashion [9, 22] while recent works start to incorporate the inductive bias of a spatial map structure into policy representation by introducing a differentiable spatial memory [16, 34, 37], semantic prior knowledge [30] or learning a topological scene graph [2, 8, 63].

The Active Neural SLAM (ANS) method [6] is the state-of-the-art framework for single-agent visual exploration(details in Sec. 3.2). There are also follow-up enhancements based on the ANS framework, such as improving map reconstruction with occupancy anticipation [40] and incorporating semantic signals into the reconstructed map for semantic exploration [7]. Our MAANS can be viewed as a multi-agent extension of ANS with a few multi-agent-specific components.

2.2 Multi-agent Cooperative Exploration

There have been works extending planning-based visual exploration solutions to the multi-agent setting by introducing handcraft planning heuristics over a shared reconstructed 2D map [5, 11, 12, 18, 35, 38, 60]. However, due to the lack of learning, these methods may have the limited potential of capturing non-trivial multi-agent interactions in challenging domains. By contrast, multi-agent reinforcement learning (MARL) has shown its strong performances in a wide range of domains [36], so many works have been adopting MARL to solve challenging cooperative problems. Representative works include value decomposition for approximating credit assignment [42, 49], learning intrinsic rewards to tackle sparse rewards [20, 29, 57] and curriculum learning [31, 58].

However, jointly optimizing multiple policies makes multi-agent RL training remarkably more challenging than its single-agent counterpart. Hence, these end-to-end RL methods either focus on much simplified domains, like grid world or particle world [55], or still produce poor exploration efficiency compared with classical planning-based solutions. Our MAANS framework adopts a modular design and is the first RL-based solution that significantly outperforms classical planning-based baselines in a photo-realistic physical environment.

Finally, we remark that MAANS utilizes a centralized global planner MSP, which assumes perfect communication between agents. There are also works on multi-agent cooperation with limited or constrained communication [24, 39, 48, 15, 22, 56, 69], which are parallel to our focus.

2.3 Size-Invariant Representation Learning

There has been rich literature in deep learning studying representation learning over an arbitrary number of input entities in deep learning [67, 68]. In MARL, the self-attention mechanism [54] has been the most popular policy architecture to tackle varying input sizes [14, 23, 44, 59] or capture high-order relations [19, 32, 63, 66]. A concurrent work [56] also considers the zero-shot team-size adaptation in the photo-realistic environment by learning a simple attention-based communication channel between agents. By contrast, our works develop a much expressive network architecture, Spatial-TeamFormer, which adopts a hierarchical self-attention-based architecture to capture both intra-agent and spatial relationships and results in substantially better empirical performance (see Section 5.4). Besides, parameter sharing is another commonly used technique in MARL for team-size generalization, which has been also shown to help reduce nonstationarity and accelerate training [10, 52]. Our work follows this paradigm as well.

3 Preliminary

3.1 Task Setup

We consider a multi-agent coordination indoor active SLAM problem, in which a team of agents needs to cooperatively explore an unknown indoor scene as fast as possible. At each timestep, each agent performs an action among *Turn Left*, *Turn Right* and *Forward*, and then receives an RGB image through a camera and noised pose change through a sensor, which is provided from the Habitat environment. We consider a decision-making setting by assuming perfect communication between agents. The objective of the task is to maximize the accumulative explored area within a limited time horizon.

3.2 Active Neural SLAM

The ANS framework [6] consists of 4 parts: a neural SLAM module, a RL-based global planner, a planning-based local planner and a local policy. The neural SLAM module takes an RGB image, the pose sensory signals, and its past outputs as inputs, and outputs an updated 2D reconstructed map and a current pose estimation. Note that in ANS, the output 2D map only covers a neighboring region of the agent location and always keeps the agent at the egocentric position. For clarification, we call this raw output map from the SLAM module a *agent-centric local map*. The global planner in ANS takes in an augmented agent-centric *local map* as its input, and outputs two real numbers from two Gaussian distributions denoting the coordinate of the long-term goal. The local planner performs classical planning, i.e., Fast Marching Method (FMM) [47], over the agent-centric local map towards a given long-term goal, and outputs a trajectory of short-term sub-goals. Finally, the local policy produces actions given an RGB image and a sub-goal and is trained by imitation learning.

With the advantage of the modeling capability of arbitrarily complex strategy in RL, an RL-based global planner which determines the global goals encourages

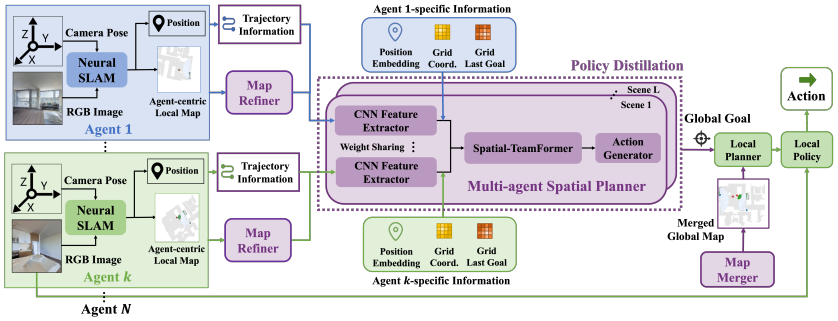

Fig. 1. Overview of *Multi-Agent Active Neural SLAM* (MAANS).

exploration faster. To apply RL training, we model the problem as a decentralized partially observable Markov decision process (Dec-POMDP). Dec-POMDP is parameterized by $\langle S, A, O, R, P, n, \gamma, h\rangle$. n is the number of agents. S is the state space, A is the joint action space. $o^{(i)} = O(s; i)$ is agent i's observations at state s. $P(s'|s, a)$ defines the transition probability from state s to state s' via joint action a. $R(s, A)$ is the shared reward function. γ is the discount factor. The objective function is $J(\theta) = \mathbb{E}_{a,s}[\sum_t \gamma^t R(s^t, a^t)]$. In this task, the policy π_θ generates a global goal for each agent every decision-making step. The shared reward function is defined as the accumulative environment reward every global goal planning step.

4 Methodology

The overview of MAANS is demonstrated in Fig. 1, where each agent is presented in a modular structure. The *Neural SLAM* module corrects the sensor error and performs SLAM in order to build a top-down 2D occupancy map. Then we use a *Map Refiner* to rotate each agent's egocentric local map to a global coordinate system. We augment these refined maps with each agent's trajectory information and feed these spatial inputs along with other agent-specific information to our core planning module, *Multi-agent Spatial Planner (MSP)* to generate a global goal as the long-term navigation target for each individual agent. We remark that only estimated geometric information is utilized in this map fusion process. To effectively reach a global goal, the agent first plans a path to this long-term goal in a manually merged global map using FMM and generates a sequence of short-term sub-goals. Finally, given a short-term sub-goal, a *Local Policy* outputs the final navigation action based on the visual input and the relative spatial distance as well as the relative angle to the sub-goal. Note that the Neural SLAM module and the Local Policy do not involve multi-agent interactions, so we directly reuse these two modules from ANS [45]. We fix these modules throughout training and only train the planning module MSP using the MAPPO algorithm [64].Hence, the actual action space for training MSP is the spatial location of the global goal.

4.1 Multi-agent Spatial Planner

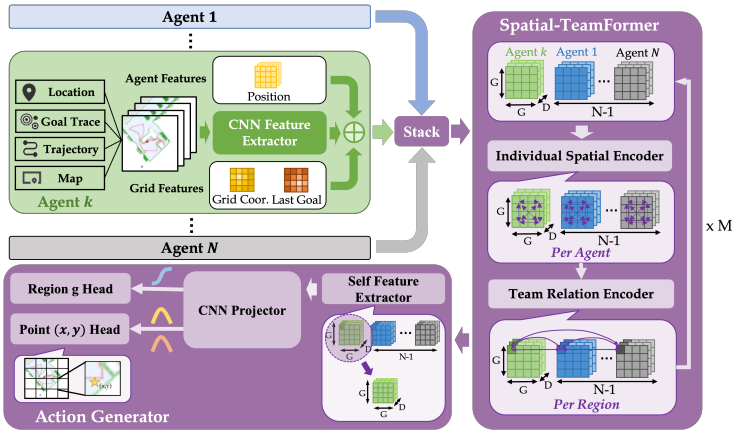

Fig. 2. Workflow of *Multi-agent Spatial Planner* (MSP), including a CNN-based Feature Extractor, a Spatial-TeamFormer for representation learning and an Action Generator.

Multi-agent Spatial Planner (MSP) is the core component in MAANS, which could perform planning for an arbitrary number of agents. The full workflow of MSP is shown in Fig. 2. MSP first applies a weight-shared CNN feature extractor to extract spatial feature maps from each agent's local navigation trajectory and then fuses team-wise information with hierarchical transformer-based network architecture, Spatial-TeamFormer. Finally, an action generator will generate a spatial global goal based on the features from Spatial-TeamFormer. Suppose there are a total of N agents and the current decision-making agent has ID k. We will describe how agent k generates its long-term goal via the 3 parts in MSP in the following content. Note that due to space constraints, we only present the main ideas while more computation details can be found in Appendix A.4.

(1) CNN Feature Extractor For every single agent, we use its current location, movement trajectory, previous goal, goal history, self-occupancy map and obstacle map as inputs and convert them to a 480×480 2D map with 6 channels over a global coordinate system. We adopt a weight-shared CNN network with 5 layers to process each agent's input map, which produces a $G \times G$ feature map with $D = 32$ channels. G corresponds to the discretization level of the scene. We choose $G = 8$ in our work, leading to G^2 grids corresponding to different spatial regions in the underlying indoor scene.

Besides CNN spatial maps, we also introduce additional features, including agent-specific embeddings of its 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.

(2) Spatial-TeamFormer With a total of N extracted $G \times G$ feature maps, we aim to learn a team-size-invariant spatial representation over all the

agents. Transformer has been a particularly popular candidate for learning invariant representations, but it may not be trivially applied in this case. Standard Transformer model in NLP [54] tackles 1-dimensional text inputs, which ignores the spatial structure of input features. Visual transformers [13] capture spatial relations well by performing spatial self-attention. However, we have a total of N spatial inputs from the entire team.

Hence, we present a specialized architecture to jointly leverage intra-agent and spatial relationships in a hierarchical manner, which we call *Spatial-TeamFormer*. A Spatial-Teamformer block consists of two layers, i.e., an *Individual Spatial Encoder* for capturing spatial features for each agent, and a *Team Relation Encoder* for reasoning cross agents. Similar to visual transformer [13], *Individual Spatial Encoder* focuses only on spatial information by performing a spatial self-attention over each agent's own $G \times G$ spatial map without any cross-agent computation. By contrast, *Team Relation Encoder* completely focuses on capturing team-wise interactions without leveraging any spatial information. In particular, for each of the $G \times G$ grid, *Team Relation Encoder* extracts the features w.r.t. that grid from the N agents and performs a standard transformer over these N features. We can further stack multiple Spatial-TeamFormer blocks for even richer interaction representations.

We remark that another possible alternative to Spatial-TeamFormer is to simply use a big transformer over the aggregated $N \times G \times G$ features. Such a naive solution is substantially more expensive to compute ($O(N^2G^4)$ time complexity) than Spatial-TeamFormer ($O(N^2G^2 + NG^4)$ time complexity), which may also incur significant learning difficulty in practice (see Section 5.4).

(3) Action Generator The Action Generator is the final part of MSP, which outputs a long-term global goal over the reconstructed map. Since spatial-TeamFormer produces a total of N rich spatial representation, which can be denoted as $N \times G \times G$, we take the first $G \times G$ grid, which is the feature map of the current agent, to derive a single team-size-invariant representation.

In order to produce accurate global goals, we adopt a spatial action space with two separate action heads, i.e., a discrete region head for choosing a region g from the $G \times G$ discretized grids, and a continuous point head for outputing a coordinate (x, y), indicating the relative position of the global goal within the selected region g. To compute the action probability for g, we compute a spatial softmax operator over all the grids while to ensure the scale of (x, y) is bounded between 0 and 1, we apply a sigmoid function before outputting the value of (x, y). We remark that such a spatial design of action space is beneficial since it alleviates the problem of multi-modal issue of modeling potential "good" goals, which could not be simply represented by a simple normal distribution as used in [9] (see Section 5.4).

4.2 Map Refiner for Aligned 2D Maps

We develop a map refiner to ensure all the maps from the neural SLAM module are within the same coordinate system. The workflow is shown as the blue and green part in Fig. 3. The map refiner first composes all the past *agent-centric local maps* to recover the agent-centric *global* map. Then, we transform the coordinate

system based on the pose estimates to normalize the global maps from all the agents w.r.t. the same coordinate system. To ensure the feature extractor in MSP concentrates only on the viable part and also induce a more focused spatial action space, we crop the unexplorable boundary of the normalized map and enlarge the house region as our final refined map.

4.3 Map Merger for Improved Local Planning

The local planner from ANS plans sub-goals on the agent-centric local map, while in our setting, we can also leverage the information from other agents to plan over a more accurate map. The diagram of map merger is shown in Fig. 3. After obtaining N enlarged global maps via the map refiner, the map merger simply integrates all these maps by applying a max-pooling operator for each pixel location. We remark that the artificial merged global map is only utilized in the local planner, but not in the global planner MSP. We empirically observe that having a coarse merged map produces better short-term local goal while such an artificial map is not sufficient for accurate global planning. (see Section 5.4)

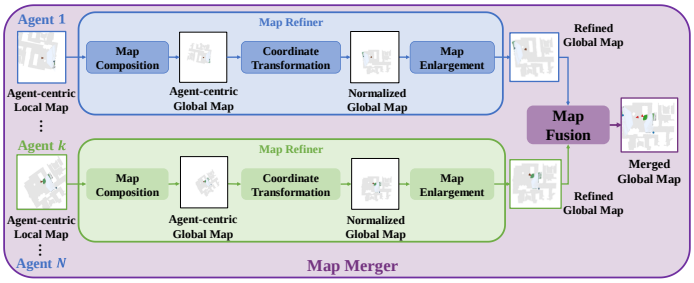

Fig. 3. Computation workflow of *map refiner* (blue and green) and *map merger* (purple).

4.4 Policy Distillation for Improved Generalization

The common training paradigm for visual exploration is multi-task learning, i.e., at each training episode, a random training scene or team size is sampled and all collected samples are aggregated for policy optimization [6, 9]. 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. Similar observations have been also reported in the existing literature [17, 51]. We use policy distillation to tackle this problem. Therefore, we adopt a two-phase distillation-based solution: in the first phase, we train separate policies for representative training scenes with a fixed team size, i.e., we choose $N = 2$ in our experiments. in the second phase, we learn another policy with $N = 2$ agents to distill the collection of pretrained

policies over different training scenes and directly measure the generalization ability of this distillation policy to novel scenes and different team sizes.

More specifically, for the i-th training scene, we first learn a specialized teacher policy $\pi(g, x, y|s, \theta_i)$ given state s with parameter θ_i, where g denotes the region output and (x, y) is the point head output. Then we train another distillation policy $\pi(g, x, y|s, \theta)$ by simply running a dagger-style [43] imitation learning, i.e., randomly rollout trajectories w.r.t. the distillation policy $\pi(s, \theta)$ and imitate the output from the specific teacher policy.

Since the region action g is discrete, we adopt a KL-divergence-based loss function while for the continuous point action (x, y), a squared difference loss between the teacher policy and distillation policy is optimized.

5 Experiment Results

5.1 Experiment Setting

We adopt scene data from the Gibson Challenge dataset [61] while the visual signals and dynamics are simulated by the Habitat simulator [46]. Although Gibson Challenge dataset provides 72 training and 14 validation scenes, we discard scenes that are not appropriate for our task, such as scenes that have large disconnected regions or multiple floors so that the agents are not possible to achieve 90% coverage of the entire house. Then we categorize the remaining scenes into 23 training scenes and 10 testing scenes. We consider $N = 2, 3, 4$ agents in our experiments. Every RL training is performed with 10^4 training episodes over 3 random seeds. Each evaluation score is expressed in the format of "mean (standard deviation)", which is averaged over a total of 300 testing episodes, i.e., 100 episodes per random seed. More details are deferred to Appendix E.

5.2 Evaluation Metrics

We take 3 metrics to examine the exploration efficiency:

1. **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.
2. **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.
3. **Mutual Overlap**: For effective collaboration, each agent should visit regions different from those explored by its teammates. We report the average overlapping explored area over each pair of agents when the coverage ratio reaches 90%. *Mutual Overlap* denotes the normalized value of this metric. Lower *Mutual Overlap* suggests better multi-agent coordination.

5.3 Baselines

We first adapt 3 single-agent planning-based methods, namely *Nearest* [62], *Utility* [25], and *RRT* [53], to our problems by planning on the merged global

map. The 3 planning-based baselines are frontier-based, i.e., they choose long-term navigation goals from the boundary between currently explored and unexplored area using different heuristics. Note that though these are originally single-agent methods and are adapted to multi-agent settings by planning on the merged global map. When choosing global goals, each agent performs computation based on the merged global map, its current position and its past trajectory.

For multi-agent baselines, we compare MAANS with 3 planning-based methods, namely *Voronoi* [18], *APF* [65] and *WMA-RRT* [35]. *APF* [65] computes artificial potential field over clustered frontiers and plans a potential-descending path with maximum information gain. APF introduces resistance force among multiple agents to avoid repetitive exploration. *WMA-RRT* [35] is a multi-agent variant of RRT, in which agents cooperatively maintain a single tree and follow a formal locking-and-search scheme. *Voronoi*-based method partitions the map into different parts using a voronoi partition and each agent only searches unexplored area in its own partition. More details can be found in Appendix B.

5.4 Ablation Study

We report the training performances of multiple RL variants on 2 selected scenes, *Colebrook* and *Dryville*, and measure the *Steps* and the *Mutual Overlap* over these 2 scenes.

(1) Comparison with ANS variants We first consider 2 ANS variants:

1. **ANS-idv** We train N ANS agents to explore individually, i.e., without any communication, in the environment.
2. **ANS-stack** We directly stack all the agent-centric local maps from the neural SLAM module as the input representation to the global planner, and retrain the ANS global planner under our multi-agent task setting.

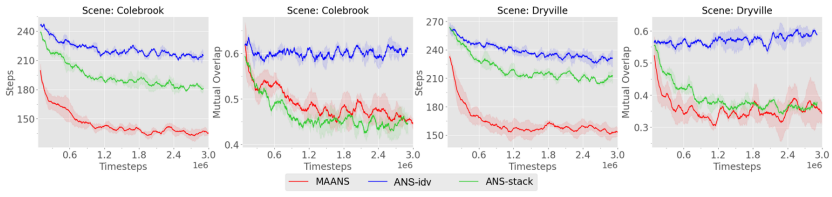

Fig. 4. Comparison between MAANS (red) and other ANS variants.

We demonstrate the training curves in Fig. 4. Regarding *Steps*, both ANS variants perform consistently worse than MAANS on each map. Regarding *Mutual Overlap*, the idv variant fails to cooperate completely while the stack variant produces comparable *Mutual Overlap* to MAANS despite its low exploration efficiency. We remark that ANS-stack performs global and local planning completely on the agent-centric *local* map while the local map is a narrow sub-region over the entire house, which naturally leads to a much conservative exploration strategy and accordingly helps produce a lower *Mutual Overlap*.

(2) Ablation Study on MSP We consider 3 additional MSP variants:

1. **MSP w.o. TeamFormer**: We completely substitute Spatial-TeamFormer with a simple average pooling layer over the extracted spatial features from CNN extractors.
2. **MSP w.o. Act. Gen.** We remove the region head from the spatial action generator, so that the global goal is directly generated over the entire refined global map via two Gaussian action distributions. We remark that such an action space design follows the original ANS paper [45].
3. **MSP-merge** We consider another MSP variant that applies a single CNN feature extractor over the manually merged global map from the map merger, instead of forcing the network to learn to fuse each agent's information.

As shown in Fig. 5, the full MSP module produces the lowest *Steps* and *Mutual Overlap*. Among all the MSP variants, *MSP w.o. Act. Gen.* produces the highest *Steps*. This suggests that a simple Gaussian representation of actions may not be able to fully capture the distribution of good long-term goals, which can be highly multi-modal in the early exploration stage. In scene *Dryville*, *MSP w.o. TeamFormer* performs much worse and shows larger training unstability than the full model, showing the importance of jointly leveraging intra-agent and spatial relationships in a hierarchical manner. In addition, *MSP-merge* produces a very high *Mutual Overlap* in scene *Dryville*. We hypothesis that this is due to the fact that many agent-specific information are lost in the manually merged maps while MSP can learn to utilize these features implicitly.

(3) Ablation Study on Spatial-TeamFormer We consider the following variants of MAANS by altering the components of Spatial-TeamFormer as follows:

1. **No Ind. Spatial Enc.**: Individual Spatial Encoder is removed from Spatial-TeamFormer
2. **No Team Rel. Enc.**: Similarly, this variant removes Team Relation Encoder while only keeps Individual Spatial Encoder.
3. **Unified**: This variant applies a single unified transformer over the spatial features from all the agents instead of the hierarchical design in Spatial-TeamFormer. In particular, we directly feed all the $N \times G \times G$ features into a big transformer model to generate an invariant representation.
4. **Flattened**: In this variant, we do not keep the spatial structure of feature maps. Instead, we first convert the CNN extracted feature into a flatten vector for each agent and then simply feed these N flattened vectors to a standard transformer model for feature learning. We remark that this variant is exactly the same as [56].

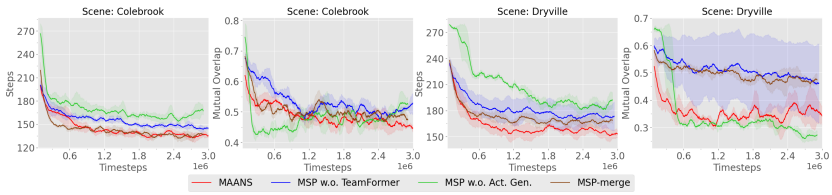

Fig. 5. Ablation studies on MSP components.

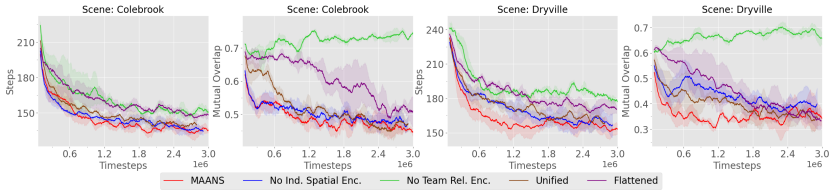

Fig. 6. Ablation studies on Spatial-TeamFormer.

We report training curves in Fig. 6. Compared with Spatial-TeamFormer, *No Team Rel. Enc.* has the highest *Mutual Overlap* and worst *Steps* on both scenes, which suggests that lacking partners' relationship attention significantly lowers the cooperation efficiency. We remark that *No Team Rel. Enc.* is indeed a single-agent variant of Spatial-TeamFormer: each agent plans global goal using its individual information while doing path planning still with the merged global map. The variant using *Flattened* features is also performing much worse than the full model with a clear margin, showing that the network architecture without utilizing spatial inductive bias could hurt final performance. When individual spatial encoder is removed (*No Ind. Spatial Enc.*), the sample efficiency drops greatly in scene *Dryville* and the method achieves a higher *Mutual Overlap* than MAANS. *Unified* also has worse sample efficiency that the full model. Note that *Unified* shows greater performance than *Flattened*, again confirming the importance of utilizing spatial inductive bias.

5.5 Main Results

Due to space constraints, we defer the full results to Appendix F.

(1) Comparison with Planning-based Baselines and RL baseline

a. Training with a Fixed Team Size: We first report the performance of MAANS and selected the baseline methods with a fixed team size of $N = 2$ agents on both representative training scenes and testing scenes in Table 1. We remark that only 9 policies of representative training scenes are used to do policy distillation(PD) since it takes a lot of work to train a separated policy for each scene. Except for the *Mutual Overlap* on the testing scenes where the performance is slightly worse than *Voronoi*, MAANS still outperforms all planning-based baselines in *Steps* and *Coverage* metrics on training and testing scenes. More concretely, MAANS reduces 20.56% exploration steps on training scenes and 7.99% exploration steps on testing scenes than the best planning-based competitor. We also compare with an RL baseline *MAANS w.o. PD*, which is trained by randomly sampling all training scenes instead of policy distillation. *MAANS w.o. PD* performs much worse than MAANS on both training and testing scenes, and slightly worse than the best single-agent planning-based method *RRT* and multi-agent planning-based method *Voronoi* on testing scenes, indicating the necessity of introducing policy distillation. Further illustration and analysis could be found in Appendix F.3.

Sce.	Metrics	Utility	RRT	APF	WMA-RRT	Voronoi	MAANS w.o PD	MAANS
Train	*Mut. Over.* ↓	0.68(0.01)	0.53(0.02)	0.61(0.01)	0.61(0.01)	0.44(0.01)	0.46(0.01)	**0.42(0.01)**
	Steps ↓	236.15(3.61)	199.59(3.27)	251.41(3.15)	268.20(2.24)	237.04(2.95)	180.25(2.35)	**158.55(2.25)**
	Coverage ↑	0.92(0.01)	0.96(0.00)	0.90(0.01)	0.87(0.01)	0.93(0.00)	0.96(0.00)	**0.97(0.00)**
Test	*Mut. Over.*↓	0.69(0.01)	0.57(0.01)	0.57(0.01)	0.64(0.01)	**0.51(0.01)**	0.57(0.01)	0.54(0.02)
	Steps ↓	161.28(2.32)	157.29(2.59)	181.18(4.17)	198.92(3.83)	156.68(3.21)	159.53(2.73)	**144.16(2.52)**
	Coverage ↑	0.95(0.00)	0.95(0.01)	0.93(0.01)	0.91(0.01)	**0.96(0.01)**	**0.96(0.00)**	**0.96(0.00)**

Table 1. Performance of MAANS and selected *planning-based* baselines and RL baseline with a fixed size of $N = 2$ agents on both training and testing scenes.

b. Zero-Shot Transfer to Different Team Sizes: In this part, 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 the best single-agent baseline *RRT* and the best multi-agent baseline *Voronoi* on both training and testing scenes is shown in Table 2. Note that experiments on testing scenes are extremely challenging since MAANS is never trained with $N = 3, 4$ team sizes on testing scenes. MAANS achieves much better performance than the best planning-based methods with every novel team size on training scenes (17.56% fewer *Steps* with $N = 3$ and 18.36% fewer *Steps* with $N = 4$) and comparable performance on testing scenes (< 3 more *Steps* with $N = 3, 4$).

# Agent	Metrics	Training Scenes			Testing Scenes		
		RRT	Voronoi	MAANS	RRT	Voronoi	MAANS
3	*Mut. Over.* ↓	0.44(0.01)	**0.37(0.01)**	0.42(0.01)	0.45(0.01)	**0.43(0.01)**	0.53(0.01)
	Steps ↓	155.13(3.26)	180.27(2.51)	**127.88(1.91)**	128.33(1.66)	**119.98(2.31)**	122.48(2.22)
	Coverage ↑	0.95(0.01)	0.95(0.00)	**0.97(0.00)**	0.95(0.01)	**0.96(0.00)**	**0.96(0.00)**
4	*Mut. Over.* ↓	0.36(0.01)	**0.34(0.01)**	0.42(0.01)	0.41(0.01)	**0.39(0.01)**	0.50(0.01)
	Steps ↓	140.57(1.78)	147.01(2.38)	**114.75(1.69)**	111.30(1.58)	**101.90(2.36)**	109.07(2.02)
	Coverage ↑	0.92(0.01)	0.93(0.00)	**0.96(0.00)**	0.93(0.01)	**0.95(0.00)**	0.94(0.00)
2 ⇒ 3	*Mut. Over.*↓	0.36(0.01)	0.35(0.01)	**0.30(0.01)**	0.43(0.01)	**0.32(0.02)**	0.46(0.01)
	Steps↓	185.94(1.83)	200.91(2.32)	**148.82(2.01)**	136.42(2.41)	148.12(6.69)	**134.11(2.88)**
	Coverage↑	0.94(0.00)	0.92(0.00)	**0.96(0.00)**	**0.96(0.01)**	0.92(0.05)	**0.96(0.00)**
3 ⇒ 2	*Mut. Over.*↓	0.35(0.01)	**0.33(0.01)**	0.41(0.01)	**0.39(0.01)**	0.42(0.01)	0.43(0.01)
	Steps↓	187.93(1.98)	206.94(2.50)	**145.14(2.83)**	139.52(3.74)	**133.77(2.83)**	145.43(3.44)
	Coverage↑	0.91(0.00)	0.89(0.01)	**0.95(0.00)**	0.94(0.01)	**0.95(0.01)**	0.94(0.01)

Table 2. Generalization performance of MAANS and selected *planning-based methods* to novel fixed and varying team sizes on training and testing scenes. Note that MAANS has the best performance on training scenes and comparable results on testing scenes.

c. Varying Team Size within an Episode We further consider the setting where the team size varies within an episode in Table 2. 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 half of episode length. In cases where the team size increases, MAANS produces substantially better performances w.r.t. every metric. In particular, MAANS achieves 33 fewer *Steps* in training scenes and lower *Steps*

than other methods in testing scenes, which suggests that MAANS has the capability to adaptively adjust its strategy. Regarding the cases where the team size decreases, MAANS consumes over 40 fewer *Steps* in $3 \Rightarrow 2$ than RRT in training scenes.

(2) Learned Strategy

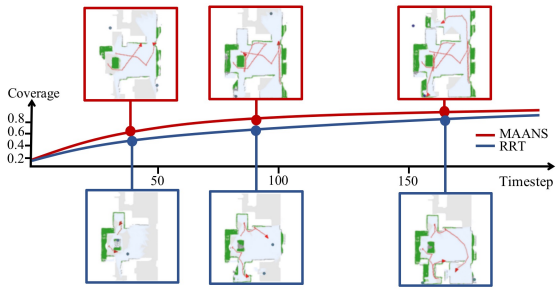

Fig. 7. Learned strategy on scene *Colebrook* of MAANS vs. RRT, where the red line with arrow represents the trajectory, the explored area shows in blue and the obstacle shows in green. MAANS achieves much higher and faster coverage ratio than RRT.

Fig. 7 demonstrates two 2-agent trials of MAANS and RRT, the most competitive planning-based method, with the same birth place. The merged global map are shown in keep timesteps. As shown in Fig 7, MAANS's coverage ratio goes up faster than RRT, indicating higher exploration efficiency. At timestep around 90, MAANS produces global goals successfully allocate the agents towards two distant unexplored area while RRT guides the agents towards the same part of the map. And at timestep around 170 when MAANS reaches 90% coverage ratio, RRT still stuck in previous explored area though there is obviously another large open space. Notice that at this key timestep RRT selects two frontiers that are marked unexplored but with no actual benefit, which an agent utilizing prior knowledge about room structures would certainly avoid.

6 Conclusion

We propose the first multi-agent cooperative exploration framework, *Multi-Agent Active Neural SLAM* (MAANS) that outperforms planning-based competitors in a photo-realist physical environment. The key component of MAANS is the RL-based planning module, *Multi-agent Spatial Planner (MSP)*, which leverages a transformer-based architecture, Spatial-TeamFormer, to capture team-size-invariant representation with strong spatial structures. We also implement a collection of multi-agent-specific enhancements and policy distillation for better generalization. Experiments on Habitat show that MAANS achieves better training and testing performances than all the baselines. We hope MAANS can inspire more powerful multi-agent methods in the future.

References

1. Anguelov, D., Dulong, C., Filip, D., Frueh, C., Lafon, S., Lyon, R., Ogale, A., Vincent, L., Weaver, J.: Google street view: Capturing the world at street level. Computer **43**(6), 32–38 (2010)
2. Bhatti, S., Desmaison, A., Miksik, O., Nardelli, N., Siddharth, N., Torr, P.H.: Playing doom with slam-augmented deep reinforcement learning. arXiv preprint arXiv:1612.00380 (2016)
3. Bresson, G., Alsayed, Z., Yu, L., Glaser, S.: Simultaneous localization and mapping: A survey of current trends in autonomous driving. IEEE Transactions on Intelligent Vehicles **2**(3), 194–220 (2017)
4. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Transactions on robotics **21**(3), 376–386 (2005)
5. Čáp, M., Novák, P., Vokřínek, J., Pěchouček, M.: Multi-agent rrt*: Sampling-based cooperative pathfinding. arXiv preprint arXiv:1302.2828 (2013)
6. 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)
7. Chaplot, D.S., Gandhi, D.P., Gupta, A., Salakhutdinov, R.R.: Object goal navigation using goal-oriented semantic exploration. Advances in Neural Information Processing Systems **33** (2020)
8. Chaplot, D.S., Salakhutdinov, R., Gupta, A., Gupta, S.: Neural topological slam for visual navigation. In: CVPR (2020)
9. Chen, T., Gupta, S., Gupta, A.: Learning exploration policies for navigation. In: International Conference on Learning Representations. ICLR (2019)
10. Chu, X., Ye, H.: Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning. CoRR **abs/1710.00336** (2017)
11. Cohen, W.W.: Adaptive mapping and navigation by teams of simple robots. Robotics and autonomous systems **18**(4), 411–434 (1996)
12. Desaraju, V.R., How, J.P.: Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees. In: 2011 IEEE International Conference on Robotics and Automation. pp. 4956–4961. IEEE (2011)
13. 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)
14. Duan, Y., Andrychowicz, M., Stadie, B.C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., Zaremba, W.: One-shot imitation learning. In: NIPS (2017)
15. Foerster, J.N., Assael, Y.M., De Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. arXiv preprint arXiv:1605.06676 (2016)
16. Henriques, J.F., Vedaldi, A.: Mapnet: An allocentric spatial memory for mapping environments. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8476–8484 (2018)
17. Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., van Hasselt, H.: Multi-task deep reinforcement learning with popart. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 3796–3803 (2019)
18. 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)
19. Iqbal, S., Sha, F.: Actor-attention-critic for multi-agent reinforcement learning. In: International Conference on Machine Learning. pp. 2961–2970. PMLR (2019)

20. Iqbal, S., Sha, F.: Coordinated exploration via intrinsic rewards for multi-agent reinforcement learning. arXiv preprint arXiv:1905.12127 (2019)
21. Isler, S., Sabzevari, R., Delmerico, J., Scaramuzza, D.: An information gain formulation for active volumetric 3d reconstruction. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 3477–3484. IEEE (2016)
22. Jain, U., Weihs, L., Kolve, E., Rastegari, M., Lazebnik, S., Farhadi, A., Schwing, A.G., Kembhavi, A.: Two body problem: Collaborative visual task completion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6689–6699 (2019)
23. Jiang, J., Dun, C., Huang, T., Lu, Z.: Graph convolutional reinforcement learning. arXiv preprint arXiv:1810.09202 (2018)
24. Jiang, J., Lu, Z.: Learning attentional communication for multi-agent cooperation. Advances in Neural Information Processing Systems **31**, 7254–7264 (2018)
25. 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)
26. Kleiner, A., Prediger, J., Nebel, B.: Rfid technology-based exploration and slam for search and rescue. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 4054–4059. IEEE (2006)
27. Li, A.Q.: Exploration and mapping with groups of robots: Recent trends. Current Robotics Reports pp. 1–11 (2020)
28. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
29. Liu, I.J., Jain, U., Yeh, R.A., Schwing, A.: Cooperative exploration for multi-agent deep reinforcement learning. In: International Conference on Machine Learning. pp. 6826–6836. PMLR (2021)
30. Liu, X., Guo, D., Liu, H., Sun, F.: Multi-agent embodied visual semantic navigation with scene prior knowledge. arXiv preprint arXiv:2109.09531 (2021)
31. Long, Q., Zhou, Z., Gupta, A., Fang, F., Wu, Y., Wang, X.: Evolutionary population curriculum for scaling multi-agent reinforcement learning. In: International Conference on Learning Representations (2020)
32. Malysheva, A., Sung, T.T., Sohn, C.B., Kudenko, D., Shpilman, A.: Deep multi-agent reinforcement learning with relevance graphs. arXiv preprint arXiv:1811.12557 (2018)
33. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
34. Mousavian, A., Toshev, A., Fišer, M., Košecká, J., Wahid, A., Davidson, J.: Visual representations for semantic target driven navigation. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8846–8852. IEEE (2019)
35. 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)
36. Nguyen, T.T., Nguyen, N.D., Nahavandi, S.: Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE transactions on cybernetics **50**(9), 3826–3839 (2020)
37. Parisotto, E., Salakhutdinov, R.: Neural map: Structured memory for deep reinforcement learning. In: International Conference on Learning Representations. ICLR (2018)
38. Patel, S., Hariharan, S., Dhulipala, P., Lin, M.C., Manocha, D., Xu, H., Otte, M.: Multi-agent ergodic coverage in urban environments

39. Peng, P., Yuan, Q., Wen, Y., Yang, Y., Tang, Z., Long, H., Wang, J.: Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. CoRR **abs/1703.10069** (2017), http://arxiv.org/abs/1703.10069
40. Ramakrishnan, S.K., Al-Halah, Z., Grauman, K.: Occupancy anticipation for efficient exploration and navigation. In: European Conference on Computer Vision. pp. 400–418. Springer (2020)
41. Ramakrishnan, S.K., Jayaraman, D., Grauman, K.: An exploration of embodied visual exploration. International Journal of Computer Vision **129**(5), 1616–1649 (2021)
42. Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., Whiteson, S.: Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: International Conference on Machine Learning. pp. 4295–4304. PMLR (2018)
43. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635. JMLR Workshop and Conference Proceedings (2011)
44. Ryu, H., Shin, H., Park, J.: Multi-agent actor-critic with hierarchical graph attention network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 7236–7243 (2020)
45. Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., Gelly, S.: Episodic curiosity through reachability. In: International Conference on Learning Representations. ICLR (2019)
46. Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al.: Habitat: A platform for embodied ai research. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9339–9347 (2019)
47. 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)
48. Sukhbaatar, S., Fergus, R., et al.: Learning multiagent communication with backpropagation. Advances in neural information processing systems **29**, 2244–2252 (2016)
49. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., et al.: Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. pp. 2085–2087 (2018)
50. Tagliabue, A., Schneider, S., Pavone, M., Agha-mohammadi, A.: Shapeshifter: A multi-agent, multi-modal robotic platform for exploration of titan. CoRR **abs/2002.00515** (2020)
51. Teh, Y.W., Bapst, V., Czarnecki, W.M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., Pascanu, R.: Distral: Robust multitask reinforcement learning. In: NIPS (2017)
52. Terry, J.K., Grammel, N., Hari, A., Santos, L., Black, B., Manocha, D.: Parameter sharing is surprisingly useful for multi-agent deep reinforcement learning. CoRR **abs/2005.13625** (2020)
53. 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
54. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio,

S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
55. Wakilpoor, C., Martin, P.J., Rebhuhn, C., Vu, A.: Heterogeneous multi-agent reinforcement learning for unknown environment mapping. arXiv preprint arXiv:2010.02663 (2020)
56. Wang, H., Wang, W., Zhu, X., Dai, J., Wang, L.: Collaborative visual navigation. arXiv preprint arXiv:2107.01151 (2021)
57. Wang*, T., Wang*, J., Wu, Y., Zhang, C.: Influence-based multi-agent exploration. In: International Conference on Learning Representations (2020)
58. Wang, W., Yang, T., Liu, Y., Hao, J., Hao, X., Hu, Y., Chen, Y., Fan, C., Gao, Y.: From few to more: Large-scale dynamic multiagent curriculum learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 7293–7300 (2020)
59. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7794–7803 (2018)
60. Wurm, K.M., Stachniss, C., Burgard, W.: Coordinated multi-robot exploration using a segmentation of the environment. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1160–1165. IEEE (2008)
61. Xia, F., Zamir, A.R., He, Z., Sax, A., Malik, J., Savarese, S.: Gibson env: Real-world perception for embodied agents. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9068–9079 (2018)
62. 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)
63. Yang, W., Wang, X., Farhadi, A., Gupta, A., Mottaghi, R.: Visual semantic navigation using scene priors. arXiv preprint arXiv:1810.06543 (2018)
64. 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)
65. 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)
66. Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., et al.: Relational deep reinforcement learning. arXiv preprint arXiv:1806.01830 (2018)
67. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 793–803 (2019)
68. Zhang, Y., Hare, J., Prugel-Bennett, A.: Deep set prediction networks. Advances in Neural Information Processing Systems **32**, 3212–3222 (2019)
69. Zhu, F., Hu, S., Zhang, Y., Hong, H., Zhu, Y., Chang, X., Liang, X.: Main: A multi-agent indoor navigation benchmark for cooperative learning (2021)