10-2019

End-to-end deep reinforcement learning for multi-agent collaborative exploration

Zichen CHEN

Budhitama SUBAGDJA
Singapore Management University, budhitamas@smu.edu.sg

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Citation

1

End-to-end Deep Reinforcement Learning for Multi-agent Collaborative Exploration

Zichen Chen[1], Budhitama Subagdja[2], and Ah-Hwee Tan[1]

[1]School of Computer Science and Engineering
Nanyang Technological University, Singapore
[2]ST Engineering-NTU Corporate Laboratory
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore
zichen002@e.ntu.edu.sg, {budhitama, asahtan}@ntu.edu.sg

***Abstract*—Exploring an unknown environment by multiple autonomous robots is a major challenge in domains such as autonomous surveillance, search and rescue, and domestic cleaning. Conventional methods usually employ a fixed strategy to allocate the robots or agents to explore selected locations like frontier points to construct a map of the environment. Although these methods can be effective in a single agent case, assigning multiple robots to explore different locations is more challenging as different agents may interfere each other making the overall tasks less efficient. In this paper, we present an approach to multi-agent exploration wherein the agents learn the effective strategy to allocate and explore the environment using a new deep reinforcement learning architecture. We propose CNN-based Multi-agent Proximal Policy Optimization (CMAPPO) algorithm for allocating multiple agents to explore different environments while, over time, improving their strategies to allocate the tasks more efficiently. This algorithm combines convolutional neural network to process multi-channel visual inputs from the observed environment, curriculum-based learning for improving learning efficiency, and PPO algorithm for motivation based reinforcement learning. Based on our evaluation, the proposed method can learn more efficient strategy for multiple agents to explore the environment than the conventional frontier-based method.**

***Index Terms*—Multi-agent exploration, Deep learning, Reinforcement Learning**

I. Introduction

One fundamental challenge in robotics domain is exploring an environment when no a priori knowledge or only partial information about the environment is available. The robot or agent needs to map the environment while performing its domain tasks like surveillance[1], search and rescue tasks[2], cleaning[3], or collecting objects[4].

The common method of robots exploration is by taking a frontier point which is located in the boundary between a known area and an unknown region as the target location to visit. This point is selected from other frontiers as revealed whenever the robots observe the environment. Some ad-hoc strategies or heuristics can be applied to optimize the selection or allocation of the targets to the agents[5][6]. However, when multiple robots are involved, the task becomes more challenging as they have to explore the unknown environment as efficient and fast as possible while avoiding conflicts or interferences among the agents that can reduce the efficiency.

In this paper, we present a learning approach for multi-agent exploration task rather than a fixed strategy to allocate the agents to their targets. We develop a deep reinforcement learning model that takes raw visual observations of the environment and represents the state of affair in terms of multi-channel maps of agents, known regions, unexplored areas, and obstacles. The state representation is used as the input to a CNN (Convolutional Neural Network) model to process and extract important features of the environment relevant to the exploration task in hand. The extracted features are then used as the input to a PPO (Proximal Policy Optimization) network to determine the target action for every agent to take.

To further improve the learning performance, we apply a curriculum model and intrinsic rewards to direct the exploration. In this way, a new better strategy can be discovered allowing efficient exploration applicable to different environments and conditions.

We evaluate our approach by comparing it with the state-of-the-art Frontier-based multi-agent exploration algorithm [5] [6]. The experimental results show that our learning approach can discover better strategies for the multi-agent indoor exploration problem. The model exhibit reasonable generalization ability across teams of agents of different sizes and in various environments.

For the remaining parts of this article, we firstly give a brief review of the relevant literature in Sec.II, and then present the formulation and training process of CMAPPO in Sec.III. An enhancement with curriculum method is introduced in Sec.IV. Lastly, the experiments and a conclusion are subsequently presented in Sec.V and Sec.VI.

II. Related Work

Yamauchi [5] introduces the frontier-based method in the multi-agent systems. Frontiers are defined as the areas on the boundary between the open environment and unexplored

The research was partially supported by the ST Engineering NTU Corporate Lab through the NRF corporate lab@university scheme.

environment. During the exploration time, each agent can maintain its local map and make a decision individually on which frontier points to visit. The goal of this method is to explore the environment as much as possible and to obtain more information about the world. Although the agents have shared a global map of the explored areas in the environment, they may still fall into conflicting or overcrowded situations since they still act individually without coordination.

Based on the frontier-based method, Bautin et al. [7] propose an improved method wherein each agent evaluates its relative rank among the other agents in terms of travel distance to each frontier. Based on the rank, each agent is assigned to the corresponding frontier as the location to visit and explore. Recently, the leading solutions to this problem are offered by segmentation-based exploration [6], perception-based exploration [8], and topological-based exploration [9].

On the other hand, tackling multi-agent problems using deep reinforcement learning have been increasingly popular nowadays. Tampuu et al. [10] introduce multi-agent cooperation and competition problem using deep Q-Learning framework in a famous video game Pong. They can learn robust strategies to play against and cooperate with another adaptive agent to gain scores. However, the solution is only for two agents (or two adversarial agents) and may not be transferable to other situations. Compared to traditional optimization solutions, deep reinforcement learning provides a solution through adaptation and learning.

Few works have been done in applying deep reinforcement learning robot exploration problems. Recently, Zhu et al. [11] use CNN and LSTM for feature extraction and A3C to approximate the model. However, their method is only for a single agent with known map and still requiring A* algorithm for planning the path to reach the nearest frontier point similar to the frontier-based method. To the best of our knowledge, the work presented in this paper is the first attempt to apply deep reinforcement learning for multi-agent exploration in unknown environment with only visual observation as the inputs.

III. CMAPPO Model

In this section, we introduce the proposed CMAPPO reinforcement learning method to solve the coordinated multi-agent exploration. We present it in two parts, one is the CNN-based observation model for understanding the exploration environment situation, the other is learning-based approach for performing multi-agent exploration.

A. Representation Learning

To represent the current observed situation and identify the important features, we use a CNN model. The CNN reflects the exploration situation and features in the environment that are relevant to the agents' overall exploration tasks.

In order to enable the agents to learn the coordinated exploration strategy, the features discovered from the CNN model are used as the input state to a reinforcement learner. In this case, the exploration strategy can be learned from scratch without any prior knowledge through two stages of learning: feature extraction by CNN and actions learning by a reinforcement learner.

The architecture of CMAPPO can be depicted in Fig. 1. The CNN part of the architecture contains six layers - three convolutional and three fully-connected networks to extract features from the raw visual input of the environment.

The first convolutional layer filters the 84x84x3 input pattern where the observation is decomposed to agents channel, obstacles channel, unexplored regions channel and explored regions channel. Each of the input channels are connected with 16 filters with kernel size of 8x8 and a stride of 4. The second convolutional layer takes the input from the output of the first convolutional layer and produces the output with 32 filters, each with kernel size of 4x4 and a stride of 2. The third convolutional layer is same as the second one and takes the output of the second layer as the input. Each hidden layer is followed by the ReLUs non-linearity with 256 neurons.

B. Proximal Policy Optimization Based Learning System

Proximal Policy Optimization (PPO) [12] is a new family of policy gradient methods for deep reinforcement learning, which alternate between sampling data through interaction with the environment. It formulates the constraint as a penalty in the objective function and optimizes it with stochastic gradient ascent.

We extend the PPO to handle the multi-agent issues by developing centralized learning and decentralized execution.

In particular, the agents share and contribute to the same PPO network. We consider that the system involves n agents and each agent i can obtain its state $s_{i,t}$ at time t. With the given state s for agent i, we compute the value and update the policy with experiences collected by all agents simultaneously. Each agent utilizes the same policy to generate an exploration strategy.

The training process alternates between actions by executing the policy in parallel and updating the policy with the all selected actions. We use the constraints and advantage estimation to optimize the KL divergence, which can be described as follows,

$$L^{KLPEN}(\theta) = \mathbb{E}[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t(s_t,a_t) - \beta KL[\pi_{\theta old}(\cdot|s_t), \pi_\theta(\cdot|s_t)]], \tag{1}$$

where π_θ is the policy, a is the selected action, A_t is the generalized advantage function, and β is a parameter used for next policy update. In each iteration, the selected actions are used to build the surrogate loss $L^{KLPEN}(\theta)$ that is optimized with the Adam optimizer [13] under the KL divergence constraint for K epochs.

In PPO, the value function V is used to estimate the advantage $\hat{A}_i^t$ that does not look beyond time-step T. After running the policy for T timesteps, the selected actions are used for an update. The loss L^V for value V_ϕ is built with squared-error loss and Adam optimizer, which can be defined as follows,

$$L^V(\phi) = -\sum_{i=1}^{N}\sum_{t=1}^{T_i}(\sum_{t'>t}\gamma^{t'-t}r_i^{t'} - V_\phi(s_i^t))^2 \tag{2}$$

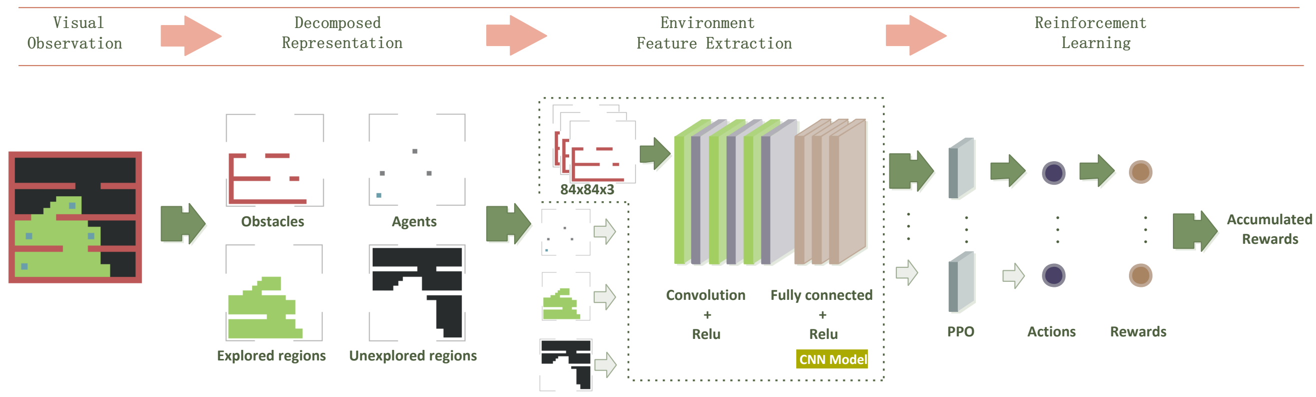


Fig. 1: The architecture of the CMAPPO. In decomposed representation, the information of obstacles is incrementally increased during the unknown environment exploration.

We use a truncated version of generalized advantage estimation, which can be formulated as follows

$$\hat{A}_i^t = \sum_{l=0}^{T_i} (\gamma\lambda)^{l-t+1} \delta_i^{l-1}, \\ \delta_i^t = r_i^t + \gamma V(s_i^{t+1}) - V(s_i^t), \tag{3}$$

where i is the agent, t specifies the time index in $[0, T]$, γ is the discount parameter for future rewards, and λ is a parameter which provides a bias-variance trade off. Policy and value function are updated independently and their parameters are not shared during the training. The multiple PPO can be scaled to the multi-agent system in a decentralized fashion where each agent performs the action individually. The completed procedure is presented in the Algorithm 1. In the next section, we will introduce some enhancement strategies to improve learning efficiency in exploring an unknown environment.

C. Problem Setup

Reinforcement learning for the multi-agent coordinated exploration problem can be formulated as a Markov Decision Process (MDP). Formally, an MDP can be described as a 6-tuple (S, A, r, p, γ, o), where S is the state space, A is the action space, r is the reward function, p is the state transition distribution, γ is the observation space and o is the observation probability distribution. In the multiple agents setting, the training network is centralized for policy learning and decentralized for action execution. Each agent has its own local observations. We describe the detailed information about our state space, action space and reward function as follows.

1) State Space: As mentioned before, the observation o_i^t consists of current self agent situation map o_a^t, other agent situation map $o_{a'}^t$, obstacles (walls) map o_w^t, unexplored regions map (goals) map o_g^t and explored regions map o_e^t. The self-agent situation map o_a^t shows current self location while other agent situation map $o_{a'}^t$ indicates the locations of other agents. The obstacles map o_w^t represents the obstacles where an agent can not pass through. The unexplored regions map o_g^t represents a region that has not been explored or visited yet, and the explored regions map o_e^t represents empty regions that have been explored by the agents. The observations are processed by CNN to approximate the policy network. Specifically, those inputs include the measurements of the current frames from a 360-degree scanner which has a maximum range of 1. It provides the 84x84 matrix of the distance values as the input for each channel (i.e. $o_i^t \in \mathbb{R}^{3x84x84}$).

2) Action Space: The action space consists of directional actions in discrete space. Each agent has four actions: move up, move down, move right and move left. For each step the agent only can move one cell and the agent only can scan its surrounding within range of a single cell. The output size is $4i$, where i is the number of agents.

3) Reward Function: The rewards for each agent can be formulated as follows,

$$r_i^t = ({}^g\mathrm{r})_i^t + ({}^p\mathrm{r})_i^t + ({}^f\mathrm{r})_i^t. \tag{4}$$

where r_i^t is the reward received by agent i at timestep t which is the sum of three components, ${}^g\mathrm{r}$, ${}^p\mathrm{r}$ and terminal reward ${}^f\mathrm{r}$. In particular, each agent is rewarded by $({}^g\mathrm{r})_i^t$ for arriving at an unexplored region

$$({}^g\mathrm{r})_i^t = \begin{cases} r_{unexplored} & if \left\| P_i^t - P_g^t \right\| \leq 1.0 \\ 0 & otherwise \end{cases}, \tag{5}$$

where the P_i^t and P_g^t are the centre location of the agent and the unexplored region respectively.

The agent will be given a small punishment $({}^p\mathrm{r})_i^t$ at each timestep. $({}^f\mathrm{r})_i^t$ is a final reward and will be given when all the areas are completely explored. We set $r_{unexplored}$ = 0.2, ${}^p\mathrm{r}$ = 0.001 and ${}^f\mathrm{r}$ = 1.0 in the training.

IV. Enhancement of CMAPPO

A. Curriculum Learning Based Sampling

Although CMAPPO may be able to learn some efficient strategy based on the raw input image of the map, it can take too long to finish given the complexity of the environment and limited observation of each agent. In this case, we apply curriculum learning to direct the learning process to acquire knowledge more accurately. The learning is conducted in a teacher-student fashion that the teacher monitors the training process and different curriculum levels, while the student learns the given tasks.

Algorithm 1: CMAPPO Algorithm

```
Input  : Environment situation observation images
Output: Motion action for each agents
Initialize the policy network $\pi_\theta$ and value function $V(s_t)$.
while True do
    for agent i = 1, 2, ...N do
        Run policy $\pi_{\theta old}$ in environment for $T$ time-steps.
        Process the observation through the CNN model, obtaining $o_i^t$.
        Obtain observation, reward and action $\{o_i^t, r_i^t, a_i^t\}$.
        Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$ using Equation 3.
        If $T_i > T_{max}$, Break (where $T_{max}$ is the maximun step)
    end
    $\pi_{\theta old} \leftarrow \pi_\theta$
    for j = 1, ..., $E_\pi$ do
        $L^{KLPEN}(\theta) = \hat{\mathbb{E}}_i^t[\frac{\pi_\theta(a_i^t|o_i^t)}{\pi_{\theta_{old}}(a_i^t|o_i^t)}\hat{A}_i^t - \beta KL[\pi_{\theta_{old}}|\pi_\theta]]$
        if KL divergence is not satisfied then
            break
        end
        Update $\theta$ by Adam optimizer.
    end
    for k = 1, ..., $E_V$ do
        $L^V(\phi) = -\sum_{i=1}^N \sum_{t=1}^{T_i}(\sum_{t'>t}\gamma^{t'-t}r_i^{t'} - V_\phi(s_i^t))^2$
        Update $\phi$ by Adam optimizer.
    end
    if KL$[\pi_{\theta_{old}}|\pi_\theta]_{>\beta_{high}KL_{target}}$ then
        $\beta \leftarrow \alpha\beta$
    end
    else
        KL$[\pi_{\theta_{old}}|\pi_\theta]_{<\beta_{high}KL_{target}}$ end
        $\beta \leftarrow \beta/\alpha$
    end
```

Based on [14], we define the curriculum as training distribution Q_λ and formulate the curriculum learning as follows

$$Q_\lambda(z) \propto W_\lambda(z)P(z) \qquad \forall z. \tag{6}$$

denote z as the random variable representing an example, $P(z)$ is the target training distribution, $W_\lambda(z)$ is the weight applied to example z at stage level λ in the curriculum sequence, and $0 \leq \lambda \leq 1$. The entropy of Q_λ distribution is increasing with $W_\lambda(z)$ and should meet the following requirements

$$H(Q_\lambda) < H(Q_{\lambda+\epsilon}) \qquad \forall\epsilon > 0, \tag{7}$$

In this case, $W_\lambda(z)$ is monotonically increasing whenever new example (ϵ) is added.

B. Intrinsic-driven Exploration

To improve the exploration, it is possible to provide the rewards not only from the environment or the domain itself but also from a condition intrinsic to the agent. The idea of intrinsic reward is introduced by Pathak et al. [15]. They propose that intrinsic reward can be a signal to enable the agents to explore the environment more rather than just getting extrinsic rewards. The approach is to reward the agents according to how much surprising the environment is, which drives the agents to learn more strategies to explore the unvisited area and find more about the surprising states. Specifically, agents tend to discover the new area (unexplored regions) more often.

With the intrinsic reward, the networks are trained by a forward and an inverse model. In the inverse model, the policy π is trained to optimize the sum of the extrinsic reward and the intrinsic reward and encodes the states s_t and s_{t+1} into features $\phi(s_t)$ and $\phi(s_{t+1})$ that are trained to predict action a_t. In the forward model, $\phi(s_t)$ and a_t are the inputs to predict the feature representation $\widehat{\phi}(s_{t+1})$ of s_{t+1}. The prediction error in the feature space is used as the intrinsic reward, which means a bigger difference of observations enables the agent to obtain a higher intrinsic reward.

We use $\hat{a}_t = (s_t, s_{t+1}; \theta_I)$ to represent the latent space and L_I (intrinsic loss) to calculate the difference between reality and prediction. The later state can be predicted by action and current state feature, denoted as $\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_F)$. We use L_F (PPO loss) to calculate the difference.

For optimizing the CMAPPO, the loss of final model is $\min_{\theta_\pi, \theta_I, \theta_F}\left[-\lambda\mathbb{E}_{\pi(s_t;\theta_\pi)}[\Sigma_t r_t] + (1-\beta)L_I + \beta L_F\right]$, where θ_π is optimized to maximize the expected sum of rewards, θ_I is the likelihood estimation that is maximum by minimizing L_I amounts under a multinomial distribution, L_I is the loss function that measures the discrepancy between the predicted and actual actions, θ_F are optimized by minimizing the loss function L_F, $0 \leq \beta \leq 1$ is used to weigh the inverse model loss against the forward model loss, $\lambda > 0$ is used to weigh the importance of the policy gradient loss against the importance of learning the intrinsic reward signal. The intrinsic reward ^{i}r is computed as $\eta L_F(\widehat{\phi}(s_{t+1}), \phi(s_t))$, where η is a scaling factor. The completed rewards can be formulated as follows

$$r_i^t = ({}^g\mathrm{r})_i^t + ({}^p\mathrm{r})_i^t + ({}^f\mathrm{r})_i^t + \zeta({}^i\mathrm{r})_i^t, \tag{8}$$

we set η = 1/128 and ζ = 0.01 in the experiments.

V. Experiments and Results

A. Experiment Environment

The algorithm of CMAPPO is implemented with TensorFlow [16] and the group of agents with scanning camera and the environment are simulated in the Unity simulation framework. We train the multi-agent coordinated exploration on a computer with a Ryzen Threadripper 1900X CPU and an Nvidia GTX 1080 Ti GPU.

The environment is presented as a 20 x 20 cells space. Each unit in the environment is assigned to one object (Figure 1): obstacle (red), explored region (green), unexplored region (black) and agent (self-perspective agent is blue, other agents are grey).

In the training experiment, we assume each agent can only move one cell within one cell scanning range at a time in the environment. We reinitialize the exploration scene when the agents are reaching the maximal steps or completing all the tasks. To demonstrate the robustness of our model, we randomly place the agents and doors, generating random office environments for the agents at each re-initialization. We use Average Moving Distance (AMD) as the measurement, which is the average moving cells of 100 times experiments. As for the evaluation, Table I shows the results from the experiments as averaged over 100 runs for each different environment.

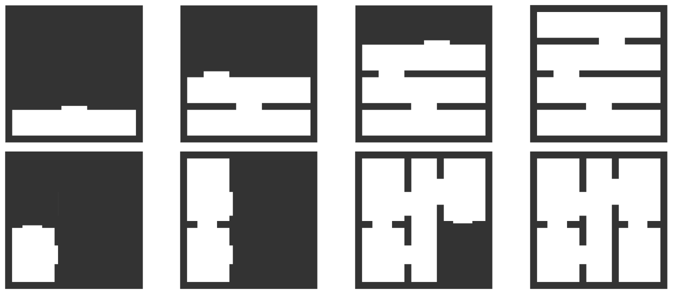

(a) Level 1 (b) Level 2 (c) Level 3 (d) Level 4

Fig. 2: The sample maps of different stage levels. The top is the maze-like environment, the bottom is the office-like environment.

(a) (b)

Fig. 3: (a) and (b) are the example of randomly generated maze-like scene and office-like scene respectively.

B. Curriculum Experiment Environment

The environments with curriculum learning have different lesson levels. The first lesson places the agents in one room to explore, so that the agents only learn to reach unexplored regions within a small area. The main objective here is to assist the agents to learn useful exploration knowledge and develop collaboration strategies first. We created a simple curriculum with 4 stages, which can be illustrated as four different uncovered regions as in Fig. 2.

The maximal steps are 1000 for each individual agent. The strength of the entropy regularization β is 0.001, discount factor γ is 0.7, and update value estimate λ is weighted by 0.95, with the constraint parameter ϵ 0.2.

C. Evaluation of CMAPPO

In order to evaluate the performance of our proposed CMAPPO model (with curriculum learning and intrinsic reward), new maps are randomly generated as our testing scenes (examples are illustrated in Fig. 3, the doors are randomly positioned). We apply the CMAPPO method in a varying difficulty level of the environment.

In all trials with CMAPPO and curriculum learning, the agents can completely explore the entire areas in the environment. Without the curriculum stages and the intrinsic reward, it fails to complete the exploration tasks as shown in Fig. 4 as no CI configuration. The figure shows the average steps taken to explore the entire environment in every consecutive stage of training with the curriculum-based method. It is shown that, features learned from a simpler environment in one stage enables faster more stable learning on another more complex one in the next subsequent stage of learning. By applying the curriculum-based learning, the agents can also generalize their learned knowledge to be applicable in a variety of conditions and environments.

Agents/Method	CMAPPO		Frontier-based Method	
	Maze	Office	Maze	Office
One agent	501	501	758	933
Two Agents	387	386	676	605
Three Agents	348	311	440	410
Four Agents	226	254	334	371

TABLE I: Performance metrics of AMD evaluated for CMAPPO method, and Frontier-based method on the target unknown environment with different number of agents (the results are rounded off to the nearest integer).

CMAPPO is able to learn the strategies for coordinated exploration in the complex unknown environment. The results are shown in Table I, which compares CMAPPO with the conventional Frontier-based method applied with an ad hoc strategy to efficiently allocate the exploration tasks to different agents. The comparison is done for different unknown environments. The Frontier-based method is a coordinated multi-agent exploration algorithm [5] [6]. Each agent selects its target based on the distance to the possible target frontier points and the utility of those points. During the exploration, the detailed selection criteria of how the agent i chooses a target t can be described as follows

$$(i,t) = \arg\max_{(i',t')}(U_{t'} - \beta \cdot V_{t'}^{i'}) \tag{9}$$

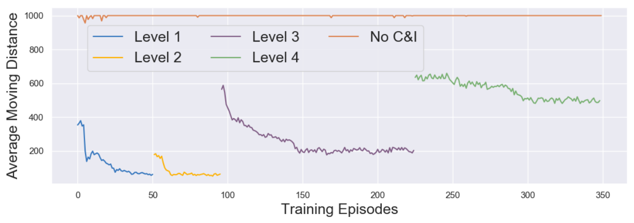
where $U_{t'}$ is the computed utility of target t, utility is defined as the minimum distance of an agent within scanning range to visit, and $V_{t'}^{i'}$ is the distance to target t. β is the relative importance of utility and distance, which generally is set to 1.

It can be seen that CMAPPO takes less steps than Frontier-based method. In Table I, we can observe that CMAPPO can make the agents explore the environment efficiently. When the number of agent is small, the CMAPPO agents can significantly outperform the frontier-based ones in terms of AMD. With more agents involved, the difference of AMD is reduced, but CMAPPO agents can still explore the environment more efficiently.

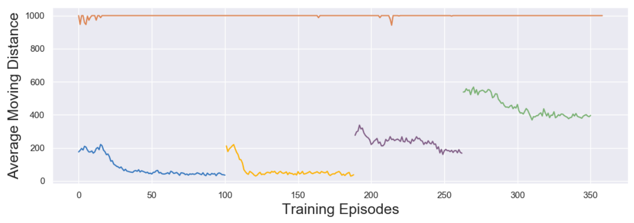
Table I shows that the number of steps taken using the strategy learned by CMAPPO is consistently smaller for all configurations of environment and the number of agents than the one using the Frontier-based strategy which also considers other agents situations. This indicates that the deep reinforcement learning of CMAPPO can acquire an effective coordination strategy directly from raw input images from the map of the environment without explicitly taking the utility or cost information of other agents into consideration.

VI. Conclusion

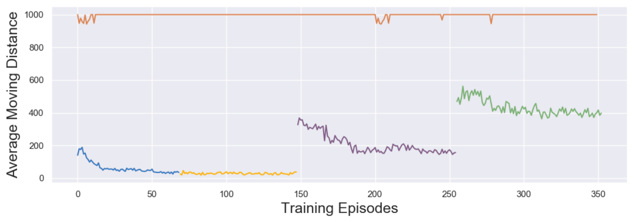
In this paper, we develop the multi-agent exploration method wherein the agents learn their coordinating strategy to allocate their tasks from experiences. We combine the concept of curriculum learning and intrinsic reward to a deep reinforcement learning system called CMAPPO to solve multi-agent exploration in unknown environment problems which only takes raw visual observation as the inputs. We have demonstrated the learning capabilities of the proposed method against the state-of-the-art Frontier-based exploration



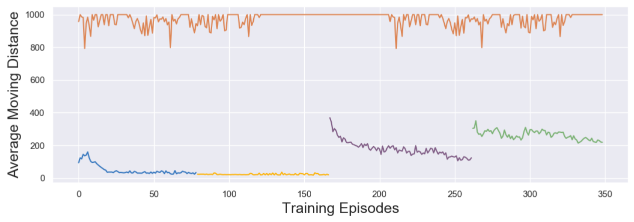
(a) One Agent



(b) Two Agents



(c) Three Agents



(d) Four Agents

Fig. 4: The AMD of average two training samples (maze-like maps) with CMAPPO method in 4 stage levels compared to the model without curriculum learning and intrinsic reward (No C&I). The total training episodes are 35,000, averaged over 100 runs.

method in various environment and conditions. It has been demonstrated that CMAPPO can learn better exploration coordination strategies that consistently outperform the Frontier-based method with a fixed mechanism for multi-agent task allocation based on cost and utility. In the future, the study can be extended to include bigger more complex environment evaluated in actual robotic systems to evaluate the practicality of the approach. The proposed model can also be extended further by including other kinds of intrinsic motivation such as exploration strategies based on novelty.

References

[1] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, 2006.

[2] Z. Beck, L. Teacy, A. Rogers, and N. R. Jennings, "Online planning for collaborative search and rescue by heterogeneous robot teams," in *AAMAS*. IFAAMAS, 2016, pp. 1024–1033.

[3] J. Fink, V. Bauwens, F. Kaplan, and P. Dillenbourg, "Living with a vacuum cleaning robot," *International Journal of Social Robotics*, vol. 5, no. 3, pp. 389–408, 2013.

[4] Z. Libin, Y. Qinghua, B. Guanjun, W. Yan, Q. Liyong, G. Feng, and X. Fang, "Overview of research on agricultural robot in china," *IJABE*, vol. 1, no. 1, pp. 12–21, 2008.

[5] B. Yamauchi, "Frontier-based exploration using multiple robots," in *AAMAS*. ACM, 1998, pp. 47–53.

[6] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *IROS*. IEEE, 2008, pp. 1160–1165.

[7] A. Bautin, O. Simonin, and F. Charpillet, "Minpos: A novel frontier allocation algorithm for multi-robot exploration," in *ICIRA*. Springer, 2012, pp. 496–508.

[8] M. Beetz, M. Tenorth, D. Jain, and J. Bandouch, "Towards automated models of activities of daily life," *Technology and disability*, vol. 22, no. 1, 2, pp. 27–40, 2010.

[9] V. Govindarajan, S. Bhattacharya, and V. Kumar, "Human-robot collaborative topological exploration for search and rescue applications," in *DARS*. Springer, 2016, pp. 17–32.

[10] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[11] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *ICRA*. IEEE, 2018, pp. 7548–7555.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*. ACM, 2009, pp. 41–48.

[15] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *ICML*, 2017.

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.