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

Zichen Chen[1], Budhitama Subagdja[2], and Ah-Hwee Tan[3]

[1,3]School of Computer Science and Engineering
[2]ST Engineering-NTU Corporate Laboratory
Nanyang Technological University, Singapore
[1]zichen002@e.ntu.edu.sg, {[2]budhitama, [3]asahtan}@ntu.edu.sg

***Abstract*—Exploring an unknown environment by multiple autonomous robots is a major challenge in robotics domains. As multiple robots are assigned to explore different locations, they may interfere each other making the overall tasks less efficient. In this paper, we present a new model called CNN-based Multi-agent Proximal Policy Optimization (CMAPPO) to multi-agent exploration wherein the agents learn the effective strategy to allocate and explore the environment using a new deep reinforcement learning architecture. The model combines convolutional neural network to process multi-channel visual inputs, curriculum-based learning, and PPO algorithm for motivation based reinforcement learning. Evaluations show that 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

A main challenge in robotics is to explore an environment when no or only partial a priori knowledge about the environment is available. The robot or agent commonly needs to map the environment and select a frontier point located in the boundary between known and unknown areas to visit while performing its domain tasks (e.g surveillance[1], search and rescue tasks[2], cleaning[3], collecting objects[4]). Some 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 also have to avoid conflicts or interferences among the agents that can reduce the efficiency.

In this paper, we present a deep reinforcement learning approach for multi-agent exploration rather than with a fixed strategy to allocate the agents to their targets. Raw visual observations of the environment containing multi-channel maps of agents, known regions, unexplored areas, and obstacles are used as the state representation and as the input to a CNN (Convolutional Neural Network) model for features extraction. The features are then passed to a PPO (Proximal Policy Optimization) network to determine the target action for every agent to take. The model is also enhanced with curriculum model and intrinsic rewards such that a new better strategy can be discovered allowing efficient exploration for different environments and conditions.

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

Our evaluation shows that CMAPPO can discover better strategies for multi-agent exploration compared to the state-of-the-art Frontier-based algorithm [5, 6].

II. Related Work

Yamauchi [5] introduces the frontier-based method in the multi-agent systems. The goal of this method is to explore the environment as much as possible and to obtain more information about the world by allowing each agent to maintain its local map and to decide which frontier points to visit. Without a proper coordination, however, this approach may not be efficient as the agents may be in conflict or interfere each other. Bautin et al. [7] propose an improvement to the frontier method wherein each agent evaluates its relative rank among the others in terms of travel distance to each frontier and they are allocated based on the ranks. 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, deep reinforcement learning approach have been used to tackle multi-agent problems. Tampuu et al. [10] address multi-agent cooperation and competition using deep Q-Learning framework in a popular video game Pong. However, only few works have been done recently in applying deep reinforcement learning for robot exploration problems. Zhu et al. [11] use CNN and LSTM for feature extraction and A3C to approximate the model but limited only for a single agent with known map and A* algorithm to visit frontier points. To the best of our knowledge, the proposed model in this paper is the first kind that applies deep reinforcement learning for multi-agent exploration in unknown environment with only visual observation as the inputs.

III. CMAPPO Model

The architecture of CMAPPO can be depicted in Fig. 1. A CNN model is adopted to represent the current observed situation (state) and features in the environment that relevant to the exploration tasks. 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 which can be decomposed into self-agent channel (o_a^t), other agents channel ($o_{a'}^t$), obstacles channel (o_w^t), unexplored regions channel (o_g^t) and explored regions channel (o_e^t). 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.

In the reinforcement learning part of the architecture, we employ Proximal Policy Optimization (PPO) [12] for deep reinforcement learning. In particular, n agents share and contribute to the same PPO network. It is assumed that every agent i observes its state $s_{i,t}$ at time t simultaneously and then computes the action to select and execute. Each agent then learns by updating the same CMAPPO network.

To update the policy, constraints and advantage estimation are used to optimize the KL divergence,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.

After running the policy for T time steps, a loss L^V for value V_ϕ is built with squared-error and Adam optimizer, as follows,

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

We use a truncated version of generalized advantage estimation,

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

where i is the agent, $t \in [0, T]$ is the time index γ is the discount parameter for future rewards, and λ is a bias-variance trade off. Policy and value function are updated independently and their parameters are not shared during the training.

The detailed information about the agent's state space, action space and reward function can be described as follows.

1) State Space: As mentioned above, the state is represented as multi-channel observation $o_i^t = (o_a^t, o_{a'}^t, o_w^t, o_g^t, o_e^t)$. 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, $({}^g\mathrm{r})_i^t = r_{unexplored}$ whenever i arrives at an unexplored region or $({}^g\mathrm{r})_i^t = 0$ otherwise.

$({}^p\mathrm{r})_i^t$ is a small penalty for i at each timestep. $({}^f\mathrm{r})_i^t$ is a final reward 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 during training.

The completed procedure of CMAPPO is presented in Algorithm 1.

IV. CMAPPO WITH CURRICULUM

Learning effective strategies for exploration based on multiple channels of raw observation images and maps may take very long time. We apply curriculum learning method to CMAPPO to direct the process of learning in a teacher-student fashion comprising different curriculum levels to simplify the learning process. We define a curriculum as training distribution Q_λ based on [14] which can be formulated as follows,

$$Q_\lambda(z) \propto W_\lambda(z)P(z) \qquad \forall z. \tag{5}$$

where z denotes a random variable representing an example, $P(z)$ is the target training distribution, $W_\lambda(z)$ is the weight applied to 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,

$$H(Q_\lambda) < H(Q_{\lambda+\epsilon}) \qquad \forall \epsilon > 0. \tag{6}$$

Besides the curriculum, the agents may also receive rewards based on conditions intrinsic individually to the agents to speed up the learning. Following the idea from Pathak et al. [15], an agent is also rewarded according to the level of surprise when visiting a location. The intrinsic reward can be based on forward model which predicts feature $\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_F)$ of state s_{t+1} based on the previous state feature $\widehat{\phi}(s_t)$ and the previous action taken a_t. θ_F is the likelihood estimation parameter of the feature $\hat{\phi}(s_{t+1})$. In CMAPPO, the intrinsic reward ${}^ir$ is computed as weighted loss function ${}^ir = \eta L_F(\hat{\phi}(s_{t+1}), \phi(s_t))$, where η is a scaling factor and L_F is the loss function of the forward model. The complete reward $\widehat{r}_i^t$ can be formulated as $\widehat{r}_i^t = r_i^t + \zeta({}^i\mathrm{r})_i^t$, where $\zeta \in [0, 1]$ denotes the significance of intrinsic reward in the overall reward. Thus, together with the inverse model $\hat{a}_t = (s_t, s_{t+1}; \theta_I)$ to determine the optimal action given the two consecutive states, the learning in CMAPPO involves optimizing the likelihood estimation parameters of the network

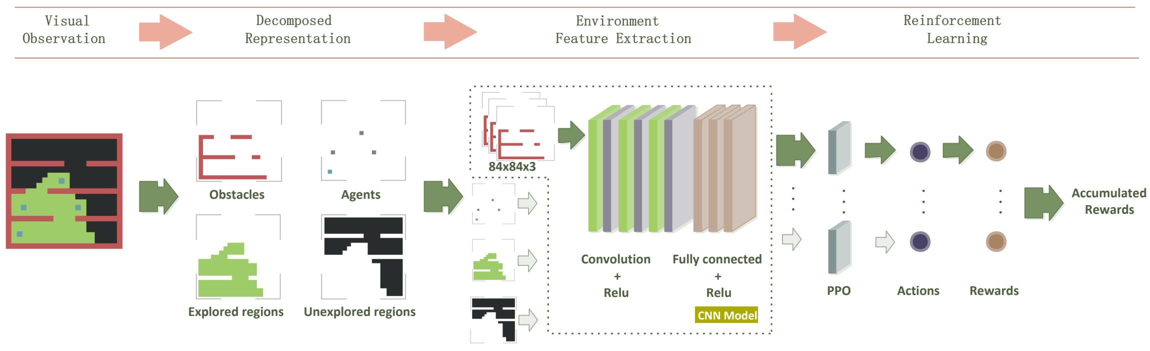


Fig. 1: The architecture and process flow of CMAPPO.

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.
        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.
        Break when $T_i > T_{max}$ (where $T_{max}$ is the maximum 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
```

$\min_{\theta_\pi, \theta_I, \theta_F} \left[-\lambda \mathbb{E}_{\pi(s_t;\theta_\pi)}[\Sigma_t r_t] + (1-\beta)L_I + \beta L_F\right]$, where θ_π, θ_I, and θ_F are expected sum of rewards, likelihood estimation of the inverse model, and likelihood estimation of the forward model respectively. L_I and L_F are the loss functions for inverse and forward model respectively. $\beta \in [0, 1]$ is used to weigh the inverse model loss against the forward model loss and $\lambda > 0$ is used to weigh the importance of the policy gradient loss against the importance of learning the intrinsic reward signal. In our experiments, we set η = 1/128 and ζ = 0.01.

V. Experiments and Results

CMAPPO is implemented with TensorFlow and the Unity game/simulation framework to simulate the environment and the tasks of the agents as multi-robot exploration with laser proximity sensors. The training process is conducted on a PC workstation equipped with a Ryzen Threadripper 1900X CPU and an Nvidia GTX 1080 Ti GPU.

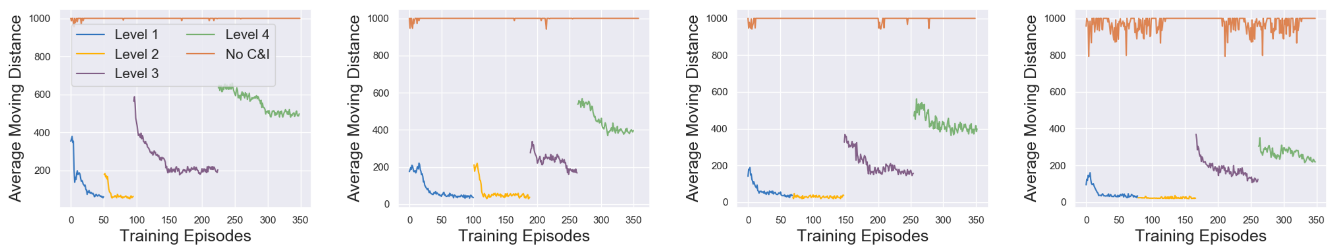
The environment is represented as a 20 x 20 cells space, each is assigned with an attribute (Figure 1): obstacle (red), explored region (green), unexplored region (black) and agent (self-perspective agent is blue, other agents are grey). Each agent is assumed to be able to only move one cell at a time with a single cell scanning range. The task environment is reinitialized every time the agents complete the task or exceed the time limit. We also randomly place the agents, doors, and randomly generate office environments for each restart. We use Average Moving Distance (AMD) as the measurement, which is the average moving cells of 100 times experiments. Table I shows the results from the experiments averaged over 100 runs for each different environment.

Curriculum learning is applied by putting the agents to four different environments with different complexity or difficulty levels. The agents learn to explore a simple environment (single room) first and becomes more challenging over time. Each individual agent can move limited to 1000 steps only. 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.

The results show that in all trials with CMAPPO, the agents can completely explore the entire areas. Without the curriculum stages and the intrinsic reward, it fails to complete the exploration tasks as shown in Fig. 2 (no C&I configuration).

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 [5, 6]. The comparison is done for different unknown environments. The Frontier-based method chooses a target based on the distance to possible frontier points and their utilities.

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 with 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



(a) One Agent (b) Two Agents (c) Three Agents (d) Four Agents

Fig. 2: 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.

Agents/Method	CMAPPO		Frontier-based Method	
	Maze	Office	Maze	Office
One agent	501	498	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).

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 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.