Pattern-RL: Multi-Robot Cooperative Pattern Formation via Deep Reinforcement Learning

Jia Wang, Jiannong Cao, Milos Stojmenovic, Miao Zhao, Jinlin Chen, Shan Jiang
The Hong Kong Polytechnic University, Kowloon, Hong Kong, China
The Hong Kong Polytechnic University ShenZhen Research Institute, Shenzhen
Email: {csjiawang, csjcao, csmolis, csmiaoz, csjinlin, cssjiang}@comp.polyu.edu.hk

***Abstract*— Autonomous, arbitrary pattern formation is one of the most critical applications in multi-robot systems, where robots are required to form into circles, lines, and meshes or any other desired configuration. This task is important in military applications, search and rescue operations, and visual inspection of infrastructure and equipment tasks to name a few. Most existing works are very rigid, and only able to form certain shapes, where slight target changes can cause failure in the predefined pattern-specific rules and trigger algorithm redesign. We propose a novel, deep reinforcement learning based method that generates general-purpose pattern formation strategies, in the form of deep neural networks (DNN), for any target pattern. Our method uses the trial-and-error feedback of each round of training to gradually generate the pattern formation strategy. Thus, robots are able to query the trained DNN model to select their optimal directions and speeds in a fully distributed manner. Considering that reinforcement learning models do not perform well with large state spaces and highly variant training samples, we employ auto-encoders to learn the condensed representation for each state and compute model-free policy gradients for arbitrary pattern formation. We experimentally show that groups of robots are able to form various general target patterns while minimizing the number of completion time steps.**

I. Introduction

Cooperation is the foundation for multi-robot systems (MRSs) to better perform tasks than a single robot, with respect to efficiency and robustness. As one of the most challenging directions in MRS cooperation, pattern formation has attracted growing research attention. In many real-world applications, a group of autonomous robots are required to behave in a cooperative fashion such as autonomous vehicle alignment on a highway [1], area coverage using ground/aerial/underwater robots [2], security and surveillance [3], and cooperative search [4]. With our designed multi-robot pattern formation protocol, the team level objectives can be achieved through sensing, action planning and movement control [5].

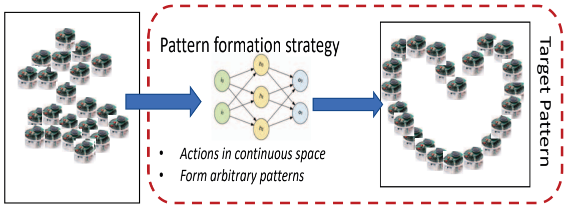


Fig. 1: Example of pattern formation

The multi-robot pattern formation problem has been extensively studied from different perspectives [6], [7]. Some solutions are based on centralized control [8], [9], which assume that a central server has global knowledge when planning the actions of the robots; and others are distributed approaches which can be roughly categorized into three classes [10], [11]: consensus-based methods, leader-follower methods, and bio-inspired methods. Consensus-based algorithms [12], [13] require all robots to reach an agreement initially through communication and then move to corresponding target locations. Leader-follower methods [14], [15] force each group member to follow the movement of the selected leader to perform the task. Bio-inspired methods [16], [17] enable multi-agents to achieve a specific task by employing attractive and repulsive physical forces. The above mentioned methods are almost all rule-based. They require to discretize agent's continuous actions to compute the pattern formation strategies. However, a coarse discretization suffers from low solution quality and a fine discretization makes it intractable to compute the optimal strategy using mathematical programming techniques, especially in high-dimensional, continuous action spaces. In addition, rule based methods do not cater to all interaction scenarios, and the designed rules are pattern specific, not very robust, and have great difficulty guaranteeing optimal performance.

We made a machine learning based attempt to solve the multi-robot arbitrary pattern formation problem in order to address the limitations in existing work. As illustrated in Fig. 1, the robots are initially scattered in the defined space, but aim to form the target patterns from their starting positions. The target patterns can be located anywhere in the space, as only the relative positions of the robots are important for task completion. Previous work [18] employed Q-learning methods but only had success for discrete actions, where the robot can only move in four directions. To enable continuous movement, we model the pattern formation strategy as deep neural networks and consider the policy gradient method [19], [20] to update the strategy. However, there are two major challenges when directly employing policy gradients to solve this problem. First, the substantial number of interaction possibilities between robots and the environment forms high-dimensional state-action spaces. Second, the intended movement of the other robots is unknown when a robot plans its own next movement, so it usually takes quite

978-1-7281-4550-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICMLA.2019.00040

a long time to converge with a conventional implementation of policy gradients.

To address the aforementioned challenges, we propose a novel multi-robot pattern formation scheme via deep reinforcement learning named *pattern-RL*, in which the general purpose coordination strategy is learned by the group-level interactions among robots. In particular, our proposed method utilizes auto-encoders [21] to learn latent and compressed representations of observation states. As a result, the policy search space can be greatly narrowed down to facilitate a quick response from robots to environmental changes. Moreover, the learning process is build upon the proximal policy optimization (PPO) [22], in which the gradient updating is performed on the batch of samples instead of updating a single gradient per data sample. This extension effectively reduces the variance in the estimation of the gradient direction, which is especially useful to achieve an effective and stable pattern formation policy learning.

The main contributions of this work are: (1) To the best of our knowledge, we are the first to employ deep reinforcement learning to achieve multi-robot general pattern formation. By first training over many different basic formations, Pattern-RL can continuously learn a new difficult formation without a long training process. Our method outperforms existing rules based approaches, and is robust to collision avoidance. (2) Our method is able to solve sequences of multi-robot pattern formations, which is difficult for rule-based methods to accomplish. To assist these major contributions, the minor contributions are: (1) Considering the high dimensionality of the observation space, our proposed method applies the cascading usage of auto-encoders to condense representation learning in order to accelerate the learning process. (2) We enable each agent to interact with the historical actions and bound the new policy relatively closely to its predecessor, which ensures a steady strategy improvement. Experimental results demonstrate that the performance of pattern-RL significantly outperforms the collision avoidance based solutions by reducing the number of time steps required to form n-regular polygons by 37.5% to 43.8% when testing between 4 to 20 agents.

II. Pattern Formation Problem

The multi-robot pattern formation problem considers n homogeneous robots moving on a 2D Euclidean plane to form a desired pattern C in the shortest possible time. Here C consists of the desired coordinates of the n robots. All robots cycle through three stages of behavior: observing, computing, and moving synchronously. At the observation stage at time t, a robot i captures all other robot positions' under its local coordinate system by its observations. Then the robot computes its action a_i^t and then adjusts its trajectory and speed. The current configuration of the robot group is P, and our objective is to allow each robot to make a sequence of decisions so that the whole group of robots can form the predefined pattern C in the shortest possible time without collisions. The problem is formally defined as:

$$\arg\min_{\pi} \; E[\frac{1}{N}\Sigma_{i=1}^{N} t_i|\pi] \tag{1}$$

$$\textit{s.t. } p_i^t = p_i^{t-1} + \triangle t a_i^t, \forall t \tag{2}$$

$$||p_i^t - p_j^t|| > 2R, (i \neq j) \tag{3}$$

$$||a_i^t|| < v_{max} \tag{4}$$

$$P \equiv C. \tag{5}$$

Eq. (1) represents our objective, which is to find the policy π which minimizes the time for N robots to form the predefined shape C, where t_i is the time step for the i^{th} agent to form the shape. Eq. (2) describes the robot's updated position. Eq. (3) enforces a distance gap of $2R$ between any two robots to ensure collision avoidance, Eq. (4) constrains the velocity below the maximum value v_{max}. Eq. (5) constrains that multi-robots finally form the target pattern. It is worth pointing out that traditional distributed methods address such an objective using rule-based logic. Therefore, they require a complete re-design of the algorithm if the designated pattern changes. In contrast, *pattern-RL* can adaptively learn the strategy model based on multi-agent[1] real-time interaction feedback in order to adapt to the pattern changes. We study the pattern formation problem from an algorithmic point of view, and therefore consider robotic hardware, sensors, and actuators as out of the scope of this paper.

III. Preliminaries and Framework

We start with an explanation of the RL concept, then we introduce the framework of the distributed multi-robot pattern formation system. Finally, we elaborate on the learning algorithm of our proposed *pattern-RL* solution.

A. Reinforcement Learning Setup

Reinforcement learning is a class of machine learning for solving sequential decision-making problems with unknown state-transition dynamics [23]. The pattern formation problem requires multi-robots to make a sequence of decisions based on the topology of their neighbors. This problem can be formulated as a partially observed Markov decision process (POMDP) and solved using the reinforcement learning framework. Typically, a POMDP contains 6-tuple (S, A, T, R, O, Ω), where S, A, T, R are the states, actions, transitions, and rewards of the system. The partial observation space $O \sim \Omega(s)$ is relevant to the observation probability distribution Ω and the system state $s \in S$. In the following, we formulate the pattern formation problem in the context of reinforcement learning:

- **Observation space O:** An observation $o_i^t \in \mathbf{O}$ of robot i is defined as the combination of the position data (coordinates) of all other robots in three consecutive frames.
- **Action space A:** An action $a_i^t \in \mathbf{A}$ controls the direction and velocity of the robot i at time t.

[1]The terms agent and robot are considered interchangeable in this paper.

- **Reward R:** After robot i takes action a_i^t under observation o_i^t, the robot receives immediate reward r_i^t as follows:
$$r_i^t = r_g + r_c + \beta r_p. \tag{6}$$
Based on the aforementioned objective function, we design the reward r_i^t by jointly considering the fatigue penalty r_g (how many steps robot i has already taken so far), collision penalty r_c, and the formation progress term r_p. Specifically, $r_g = R_g$, which penalizes the robot at each step to encourage the robots to form the pattern in the minimal number of time steps. The second term $r_c = R_c$ denotes the collision punishment. Here R_g and R_c are both negative constants. β is a magnification factor for the last term r_p which evaluates pattern formation progress by computing the similarity of the current geometry and the target pattern as follows:
$$r_p = \min \|P_\tau - C\|_2, \tau \in \Gamma, \tag{7}$$
where Γ represents all possible sorted sets of observed positions P. As one possible set in Γ, P_τ generates an order by which the position set has the shortest absolute difference with the target pattern.

In our problem, after taking action a_i^t at observation o_i^t, the robot receives reward r_i^t. Correspondingly, the objective of *pattern-RL* is to learn the optimal strategy π_θ^* with respect to θ^* as follows:
$$\theta^* = \arg\max_\theta E_{\tau\sim\pi_\theta}(\sum_{i=1}^{N}\sum_{t=1}^{T}\gamma^t r_i^t). \tag{8}$$
The coordination strategy is the mapping function $\mathbf{O} \rightarrow \mathbf{A}$, which enables the robots to query their optimal directions and speeds at each step. The coordination strategy π_θ is realized by the deep neural network with parameter θ. τ represents the collected interaction samples (i.e.,(o_i^t, a_i^t)) based on the current version of strategy π_θ at each step. Thus, we can optimize the coordination strategy by maximizing the cumulative reward along the moving trajectory.

B. System Framework

The framework and workflow of the proposed method are illustrated in Fig. 2. The distributed multi-robot pattern formation system consists of two major working phases: training and executing. In the training phase, all robots repeat the trial-and-error process starting from their initial positions, then query the current strategy model at each step towards forming the pattern. The data collected from each round are uploaded to the server which updates the strategy model. In the execution phase, the learned models are transplanted to each robot, and act as a "brain" to control its behavior. Thus, a group of robots can execute the pattern formation problem in a distributed manner. These two phases can be executed more than once if necessary. Consider the case where the performance of the execution phase is unacceptable due to possible configuration changes (i.e., one robot is missing). We can then move back to the training phase and further modify the strategy model before updating the robots with new 'brains'. Our *Pattern-RL* method consists of two components: the auto-encoder and the strategy model. Their details are given as follows:

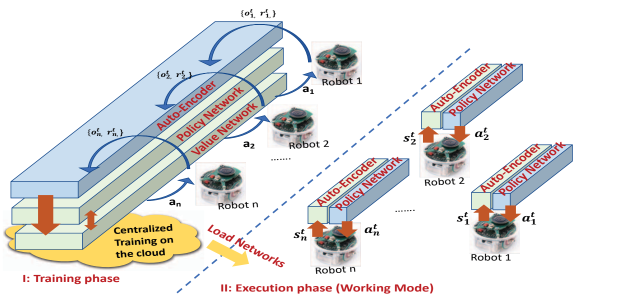


Fig. 2: The multi-robot pattern formation system framework.

Auto-Encoder. An auto-encoder is an unsupervised learning algorithm that applies back-propagation, setting the target values equal to the inputs. In practice, it is inefficient to train the model with raw observations as inputs since the observation space is large and we cannot know the spatial relations between different observations. In this paper, we obtain dense and low-dimensional vector representations for observations o_i^t via the auto-encoder network. The key idea of using the auto-encoder networks is to reduce the dimension of the observations and shrink the search space. Specifically, at each time step, the auto-encoder takes an observation vector o_i^t as input, and outputs an encoded state e_i^t as the input of the strategy networks.

Strategy model. As illustrated in Fig. 3, our strategy model includes two fully connected neural networks (i.e., a policy network and a value network). The outputs of the policy network are the mean and deviation of two normal distributions which represent the direction and speed the robots can chose from respectively. The robots select a random variable from each distribution which represents their action. The standard deviations of the distributions in the initial phases of training are relatively large, which enable robots to select from a wide range of action possibilities. As training progresses, the standard deviations become smaller, specifying which possible actions are optimal. Meanwhile, the value network V_ϕ is implemented to evaluate the quality of each action. By taking an observation and action pair as input, it estimates the cumulative future reward for taking the proposed course of action. The future rewards refer to an estimate of how close the set of robots will come to forming the designated pattern. The value network guides towards the training goal of the policy network. Fluctuations in one network would interfere with the learning of the other network, so we optimize the networks separately which leads to greater stability and better results in practice. Details on how to train these networks are explained in the next section.

C. Strategy Learning Algorithm

The core of our approach is the learning algorithm that can handle the high dimensionality of the state-action space problem. The policy gradient (PG) method can naturally handle the large state-action space problem by a function approximator. However, since this method performs only one

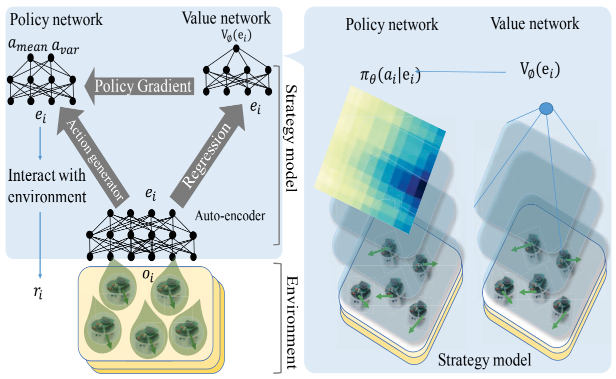


Fig. 3: *Pattern-RL* algorithm architecture.

gradient update per data sample, the naive implementation would make the strategy slow to converge. We extend the recently proposed policy gradient algorithm PPO [22] to our framework in order to learn a pattern formation strategy which performs robustly and effectively when the total number of robots is large.

The learning algorithm of our method is formally described in Algorithm 1. The training process alternates between parallel sampling of trajectories of multiple robots, and optimizing the strategy model with the sampled data. To learn a pattern, in each step, we let each robot run the current strategy and store the samples $\{(o_i, a_i, r_i)_n | i = 1, 2, ..., N; n = 1, 2, ..., E\}$ into the memory. With the samples drawn from memory, we first train an auto-encoder which yields the embedded representation e_i^t for the observation (lines 5-7). Since the action taken by a robot only affects its follow-up trajectory, the accumulated reward function Φ_π only counts the prospective rewards after time t (line 11). We construct a surrogate loss J_θ by jointly considering the advantage $\widehat{A}_\pi$ and the KL penalty constraint (line 13), where $\widehat{A}_\pi$ represents the estimated difference between the accumulated reward for the current policy and the average state value. This loss is optimized with the Adam optimizer for M iterations in order to update the policy network (lines 8-13). Consequentially, the value network $V_\phi(e_i^t)$ which outputs the baseline to estimate the advantage value is updated (lines 14-18). The coefficient λ for the KL penalty is adaptively updated and set according to lines 19-22. We set the weights of the KL penalty within the range set on line 23, and add it to the loss function (line 13). Albeit simple, this modification ensures the deviation of the new policy from the preceding policy is relatively small, which assures the learning process becomes more stable.

The executing algorithm of our method is summarized in Algorithm 2. After the training process, the policy network π_θ is mounted on each robot. At the execution stage, each robot i receives the sensor observation o_i^t and then generates the action a_i^t with the policy network π_θ. Each robot then iteratively goes through the cycles of observation, computation, and action, until the final pattern is achieved.

Algorithm 1: Distributed multi-robot coordination learning algorithm.

```
for E-episodes do
    Initialize from the random positions;
    ▷ Play game and update memory.
    Run policy $\pi_\theta$ T-steps, collecting samples $\{o_i^t, a_i^t, r_i^t\}$ for each agent $i$;
    Store samples in mem;
    for P-steps do
        ▷ Train the auto-encoder.
        $I(\mu) = \Sigma_{i=1}^N \Sigma_{t=1}^T (I_\mu(o_i^t) - o_i^t)^2$ ;
        Update $\mu$ by the gradient method w.r.t. $I(\mu)$;
    for M-steps do
        ▷ Update the policy network.
        Draw samples $\{o_i^t, a_i^t, r_i^t\}_{i=1:mG}$ from mem;
        $e_i^t = I_\mu(o_i^t)$ ;
        $\Phi_\pi = \Sigma_{t'>t} \gamma^{t'-t} r_i^{t'}$;
        Compute advantage $\widehat{A}_\pi = \Phi_\pi - V_\phi(e_i^t)$ ;
        $J(\theta) = \Sigma_{i=1}^N \Sigma_{t=1}^T \frac{\pi_\theta(a_i^t|e_i^t)}{\pi_{old}(a_i^t|e_i^t)} \widehat{A}_\pi - \lambda KL(\pi_\theta|\pi_{old})$
        Update $\theta$ by the gradient method w.r.t. $J(\theta)$;
    for B-steps do
        ▷ Update the value network.
        Draw samples $\{o_i^t, a_i^t, r_i^t\}_{i=1:mG}$ from mem;
        $e_i^t = I_\mu(o_i^t)$ ;
        $\nu(\phi) = \Sigma_{i=1}^N \Sigma_{t=1}^T (\Phi_\pi - V_\phi(e_i^t))^2$ ;
        Update $\phi$ by the gradient method w.r.t. $\nu(\phi)$;
    ▷ Adapt the KL penalty coefficient.
    if $KL(\pi_\theta|\pi_{old}) > \beta_{high} KL_{target}$ then
        $\lambda \leftarrow \alpha\lambda$ ;
    if $KL(\pi_\theta|\pi_{old}) < \beta_{Low} KL_{target}$ then
        $\lambda \leftarrow \lambda/\alpha$ ;
    ▷ Set the weight of KL penalty
      coefficient within a range.
    $\lambda \leftarrow clip(\lambda, \vartheta_{low}, \vartheta_{high})$
```

IV. Experimental evaluation

In this section, we conduct experiments to evaluate the performance of *pattern-RL*. Specifically, we begin by introducing the experimental settings and explaining the details of the implementation. Then we discuss the efficiency and stability of the learning algorithm by analyzing its learning curve. Next, we employ our method to form various patterns and evaluate their accuracy. Finally, further experiments are conducted to compare *pattern-RL* with the existing methods in terms of time efficiency.

A. Experimental Setup

All simulations are carried out on a computer with an i7-5820K CPU and an 8GB of RAM. Our algorithms are implemented in Python 2.7 with tensorflow 1.2 and the experiments are conducted using the Webots[2] platform, which simulates a team of robots. The simulated robots have the following features: (1) Robots cannot communicate but they know other robots' positions. (2) The robots have compatible compasses that have a common north orientation.

[2]https://www.cyberbotics.com/

Algorithm 2: Distributed multi-robot coordination execution algorithm.

```
Input  : Pattern formation strategy $\pi_\theta$.
Output: Trajectory $p_i^{t_0:t_g}$.
for $i \in \{1, ..., N\}$ do
    ▷ For each given agent $i$.
    while $\mathbb{P} \neq \mathbb{C}$ do
        Update time $t \leftarrow t + 1$;
        Receive sensor measurements $o_i$ ;
        Extract Feature $e_t$ ←auto-encoder($o_i$);
        $a_i^t \leftarrow \pi_\theta(e_i)$;
        $p_i^{t+1} \leftarrow p_i^t + a_i^t$;
```

TABLE I: The hyper-parameters of our experiment on training a pattern formation model.

parameters	value	parameters	value
β in Eq.6	5	E in Alg.1	800
ϑ_{high} in Alg.1	10	P in Alg.1	256
R_g in Eq.6	-0.5	M in Alg.1	256
R_c in Eq.6	-20	B in Alg.1	256
γ in Alg.1	0.9	T in Alg.1	6000
β_{high} in Alg.1	2	β_{low} in Alg.1	0.67
KL_{target} in Alg.1	0.01	α in Alg.1	0.5
ϑ_{low} in Alg.1	10^{-4}		

B. Model Configuration and Computational Complexity

To conduct the experiment, we design a policy network as a deep neural network with 3 hidden layers that consists of $256, 128$,and 128 rectified linear units (ReLUs) units, respectively. The third hidden layer is fed into the output layers with hyperbolic tangent (tanh) activation function for a_μ and softplus activation function for a_{std}. Thus, this output enables us to select the action a_i^t from a Gaussian distribution $N(a_\mu, a_{std})$, where a_μ and a_{std} refers to the mean and the standard deviation of the desired velocity. Meanwhile, the value network is implemented as a fully connected neural network with 3 hidden layers (i.e., 128, 128, and 256 ReLu units on each layer) and a single neuron output layer.

Learning a new pattern with 10 robots from scratch, the training process takes an average of 0.5 hours (about 300 episodes) to converge to a satisfactory solution. Similar to the curriculum learning paradigm, when a strategy model is trained, learning a similar pattern will take less time to converge. The hyper-parameters of our experiment are summarized in Table I. Specifically, we set the learning rate of the policy network to be 10^{-4} and value network to be 2×10^{-4}. When executing the strategy, the response time of generating new actions for each step takes 1.2 ms on average.

C. Study of the learning curve

To evaluate the learning stability of *pattern-RL*, Fig. 5 illustrates the learning curves of 10 robots jointly training a sequence of patterns (i.e., a circle, a cross, a line, and a heart). The experiment is repeated 4 times. The averaged rewards of each episodes are illustrated in Fig. 5. We compare *Pattern-RL* with the standard PPO (i.e., with clipped surrogate objective and no auto-encoder). One can see that *pattern-RL* achieves the fast convergence speed and receives higher reward values in each iteration. This experiment indicates the effectiveness of the auto-encoder and KL constraint. Intuitively, this is because it forces the updated policy to be close to the previous one, so an agent can more easily adapt to other's intentions (actions) from its previous policies and gradually improve it. Also, this experiment gives an intuition on how to choose the weight for KL constraint. We need to enfeeble the KL constraint so that robots can explore many more possible ways to form the pattern. However, weakening KL constraint comes at the cost of increasing the training time. Through many experiments we found that with $\alpha = 0.5$, the adaptive change of the KL constraint provided good performance while keeping the training cost relatively small by around 380 episodes.

D. Study of the pattern formation efficiency

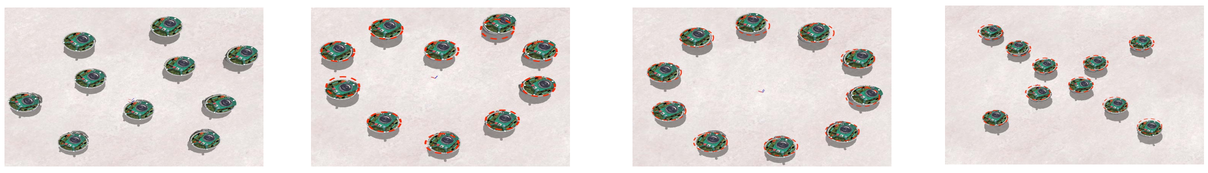
We compare pattern formation efficiency with a rule-based baseline model NH-ORCA[24]. NH-ORCA first creates rules to calculate the optimal target position for each robot, and then let the robots move there using the ORCA algorithm. We evaluate groups of n robots forming n-regular polygons trained with *pattern-RL* and with NH-ORCA. The average travel distances starting from initial random positions to the target patterns are shown in the Fig. **??** (b). In general, the path generated by the *pattern-RL* is smoother and it outperforms the baseline method in each scenario. For example, it achieves a reduction of 37.5% to 43.8% in travelling distance as the number of agents increased from 4 to 20, respectively. This experiment demonstrates that although our method takes time to learn a strategy, it is more efficient in the execution process.

E. The trajectories of forming irregular patterns

We measure the *time step* metric to evaluate the performance of our proposed approach. The *time step* counts the total number of steps for each robot to form the predefined pattern (i.e., a circle, a heart, a line, and a triangle) based on the trained strategy model. Fig. 6 illustrates the trajectories of each robot to form the target pattern. The initial positions of the robots are in the lightest color. As the robot moves, the colors of circles get more saturated. Finally, robots move to the positions highlighted by the small red circles. From the experimental results, one see that the robots are able to achieve time efficiency by following the optimal trajectories without collisions.

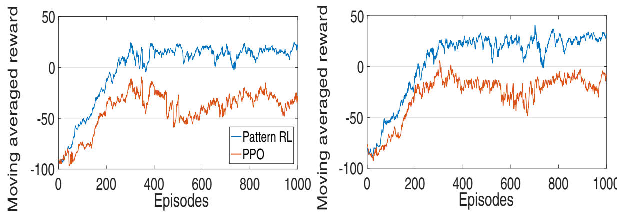
V. Conclusion

In this paper, we applied deep reinforcement learning to address the distributed multi-robot coordination problem and demonstrated its applicability on the successive pattern formation problem. Our proposed approach enables each robot to learn the general formation strategy automatically. Our method not only accelerates the training process by employing the auto-encoder to compress the high-order state space, but also stabilizes it by bounding the weights in the successive policy networks to be relatively close. Experimental results show that our method outperforms the



(a) (b) (c) (d)

Fig. 4: Multi-robot pattern formation: Ten robots (a) start from random positions transiting to form (b) a heart pattern, (c) a circle pattern, and (d) a cross pattern.



(a) (b)

Fig. 5: The average rewards of forming (a) a cross pattern and (b) a circle pattern during the training process.

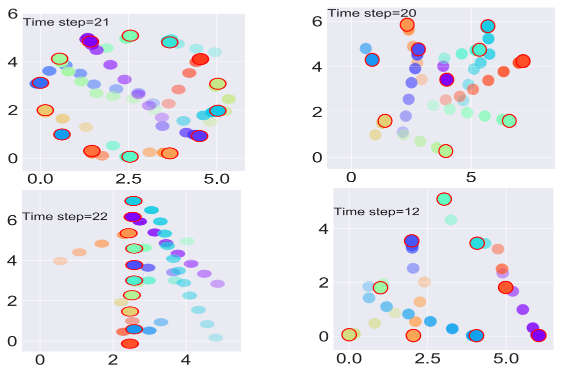


Fig. 6: The trajectories of robots. Starting from light color circles, the multi-robots form a circle pattern, a heart pattern, a line pattern, and a triangle pattern.

existing pattern formation methods in term of time efficiency as well as generality. Experiments on a real robot test-bed will be conducted in the future. We also plan to extend our framework and use it on other multi-robots coordination problems in 3D space.

Acknowledgement

This work was supported by Shenzhen Basic Research Funding Scheme under JCYJ20170818104222072.

References

[1] J. Wei, J. M. Dolan, and B. Litkouhi, "Autonomous vehicle social behavior for highway entrance ramp management," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 201–207.

[2] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[3] A. Farinelli, L. Iocchi, and D. Nardi, "Distributed on-line dynamic task assignment for multi-robot patrolling," *Autonomous Robots*, pp. 1–25, 2016.

[4] A. Macwan, J. Vilela, G. Nejat, and B. Benhabib, "Multi-robot deployment for wilderness search and rescue," *Int. J. Robot. Autom.*, vol. 31, no. 1, 2016.

[5] X. Zang, S. Iqbal, Y. Zhu, X. Liu, and J. Zhao, "Applications of chaotic dynamics in robotics," *International Journal of Advanced Robotic Systems*, vol. 13, no. 2, p. 60, 2016.

[6] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 73–80.

[7] A. Singh, H. Sharma, N. Singh, and P. Katyal, "A survey on cooperative mobile robotics," *International Journal of Applied Engineering Research*, vol. 13, no. 7, pp. 5160–5166, 2018.

[8] C. Belta and V. Kumar, "Trajectory design for formations of robots by kinetic energy shaping," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3. IEEE, 2002, pp. 2593–2598.

[9] V. Indelman, "Towards cooperative multi-robot belief space planning in unknown environments," in *Robotics Research*. Springer, 2018, pp. 441–457.

[10] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[11] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[12] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[13] E. Hasan, K. Al-Wahedi, B. Jumah, D. W. Dawoud, and J. Dias, "Circle formation in multi-robot systems with limited visibility," in *Iberian Robotics conference*. Springer, 2017, pp. 323–336.

[14] Y. Hong, G. Chen, and L. Bushnell, "Distributed observers design for leader-following control of multi-agent networks," *Automatica*, vol. 44, no. 3, pp. 846–850, 2008.

[15] S. Carpin and L. E. Parker, "Cooperative leader following in a distributed multi-robot system," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3. IEEE, 2002, pp. 2994–3001.

[16] J. Bai, G. Wen, A. Rahmani, and Y. Yu, "Distributed formation control of fractional-order multi-agent systems with absolute damping and communication delay," *International Journal of Systems Science*, vol. 46, no. 13, pp. 2380–2392, 2015.

[17] J. Wang, Y. Gu, and X. Li, "Multi-robot task allocation based on ant colony algorithm," *Journal of Computers*, vol. 7, no. 9, pp. 2160–2167, 2012.

[18] B. S. Prasad, A. G. Manjunath, and H. Ramasangu, "Multi-agent polygon formation using reinforcement learning." 2017.

[19] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1889–1897.

[20] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[21] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

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

[23] M. Egorov, "Deep reinforcement learning with pomdps," 2015.

[24] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Multi-robot system for artistic pattern formation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4512–4517.