Review on Formation Control and Motion Planning for Multi Agents

Liu Xiangyu

May 12, 2020

1 Introduction

The purpose of this article is to briefly introduce the related work on formation control and motion planning for multi agents, for a quick start on our research on formation control via reinforcement learning. In this review, complicated mathmatical details are omitted since detailed mathmatical deduction is not in the scope of this review.

The organisation of this review is as follows. Section 2,3 and 4 are basically a brief version of the survey of Liu in [1]. Section 5 reviewed some recent works on formation control algorithms using reinforcement learning.

2 Relationship between Formation Control and Cooperative Motion Planning

Relationship is illustrated in Figure 1, details later.

3 Formation Control

3.1 System Architecture of Multi Agents Formation

A generic hierarchical architecture formation system is proposed by Liu and Bucknall in [2]. The system is displayed in Figure 2, which consists of three layers, i.e. Task Management Layer, Path Planning Layer and Task Execution Layer. According to Liu and Bucknall, the first layer Task Management Layer allocates mission to indivisual vehicles based upon the critiria of maximum overall performance and minimum mission time. In other words, this layer analyzes the mission, which can be defined as a set of waypoints including starting and ending points, and design a proper formation shape, and allocate the formation information to each vehicle.

The Path Planning Layer is comprised of three sub modules, which can concluded as,

1. Cooperative path planning module. This is the core of the system, determining the optimised path for each vehicle.

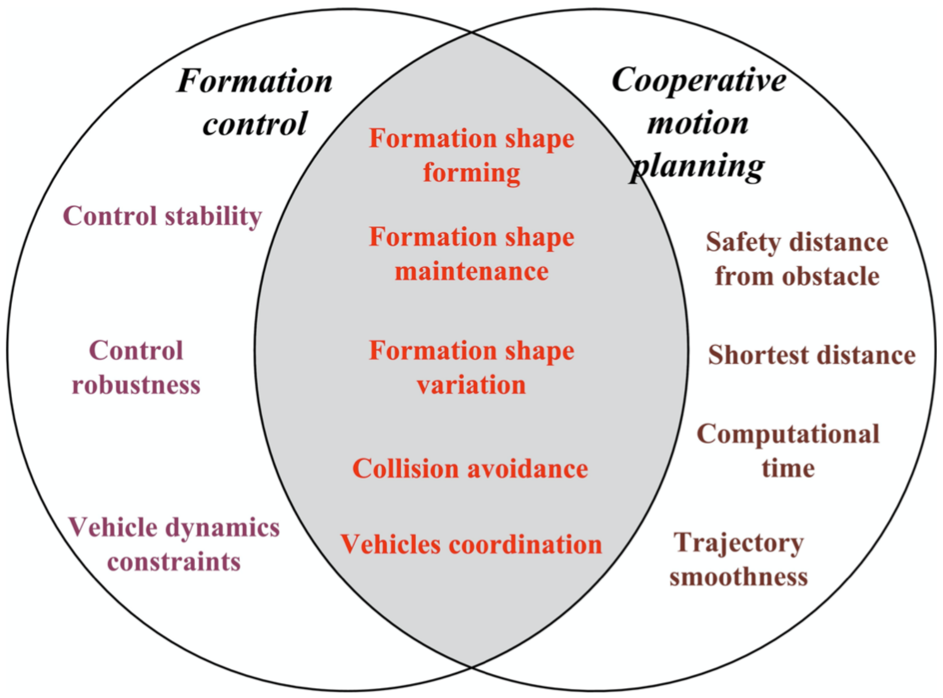


Figure 1: Comparison of formation control and cooperative motion planning [1].

2. Real-time trajectory modification module. The calculated optimised path is furthur modificated in this module considering the uncertainties in practical applications.
3. Data acquisition module.

The last module i.e. Task Execution Layer is the execution layer that directly connects to the controllers of vehicles. And execution results are fed back to upper layer to generate a closed loop control.

3.2 System Patterns of Formation

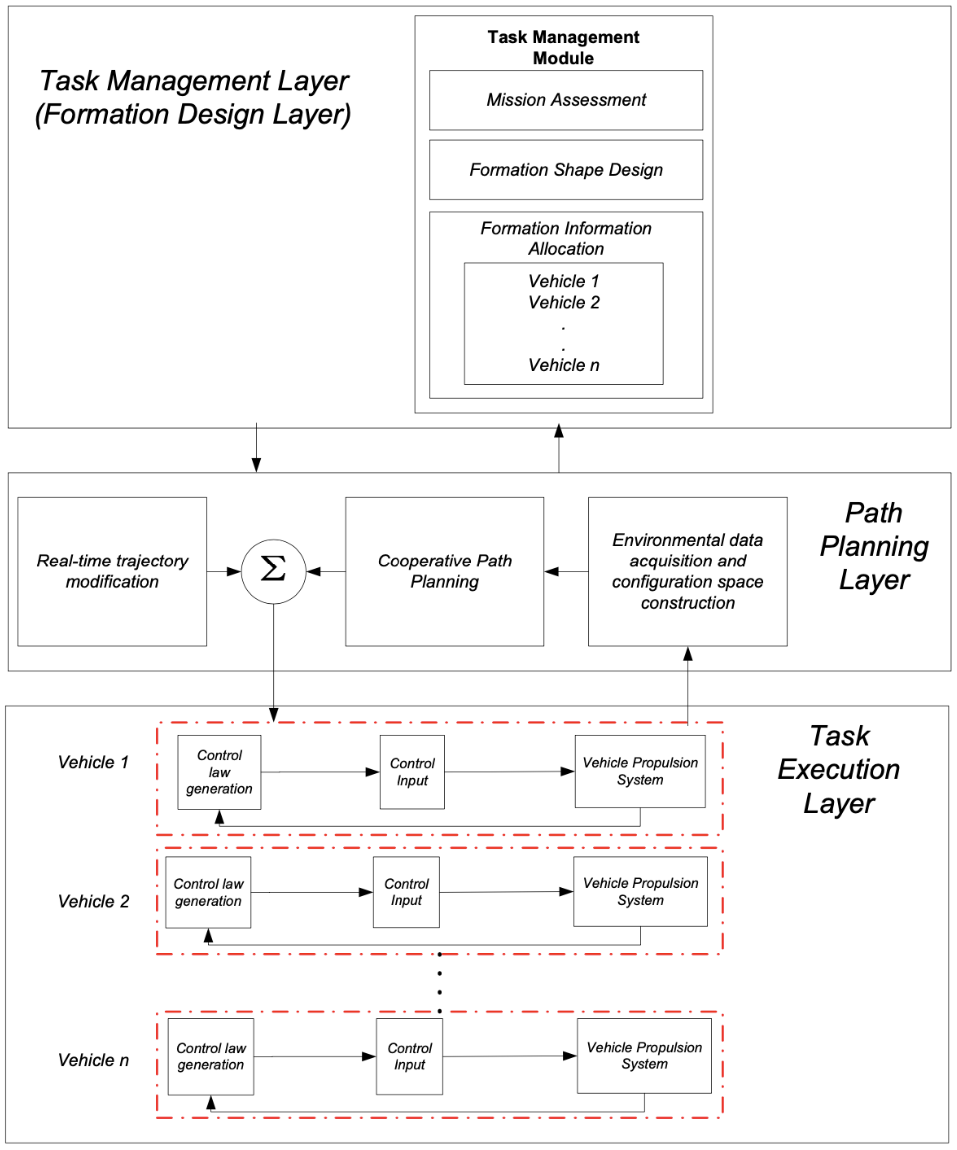
Based on Campbell et al. [3], as demonstrated in Figure 3, four shapes are commonly used:

1. Column shape, gives a wide mission area.
2. Line shape, forms a small mission area but useful in highly constrained environments.
3. V shape, suitable in normal operations, offers a good view of the surroundings, and easy and direct communication can be eastablished between vehicles.
4. Diamond shape, a variation of V shape, also frequently used in normal operations.

As mentioned in [1], there are no particular restrictions on the choice of formation shape. The shape is determined by the task management layer, and should be deformable during tasks, e.g. deform and reform into another shape when trying to enter a narrow area.

In addition, when formation shape change is occurring, the formation control strategy should consider the following additional constraints:

1. Inter-vehicle collision avoidance,
2. Coordinating of multiple vehicles, i.e. avoid the situation waiting or fully stop due to vehicles lagging behind.



Task Management Layer (Formation Design Layer)
Task Management Module
Mission Assessment
Formation Shape Design
Formation Information Allocation
Vehicle 1
Vehicle 2
.
.
Vehicle n
Real-time trajectory modification
Σ
Cooperative Path Planning
Environmental data acquisition and configuration space construction
Path Planning Layer
Task Execution Layer
Vehicle 1
Control law generation
Control Input
Vehicle Propulsion System
Vehicle 2
Control law generation
Control Input
Vehicle Propulsion System
Vehicle n
Control law generation
Control Input
Vehicle Propulsion System

Figure 2: A formationc control architecture.

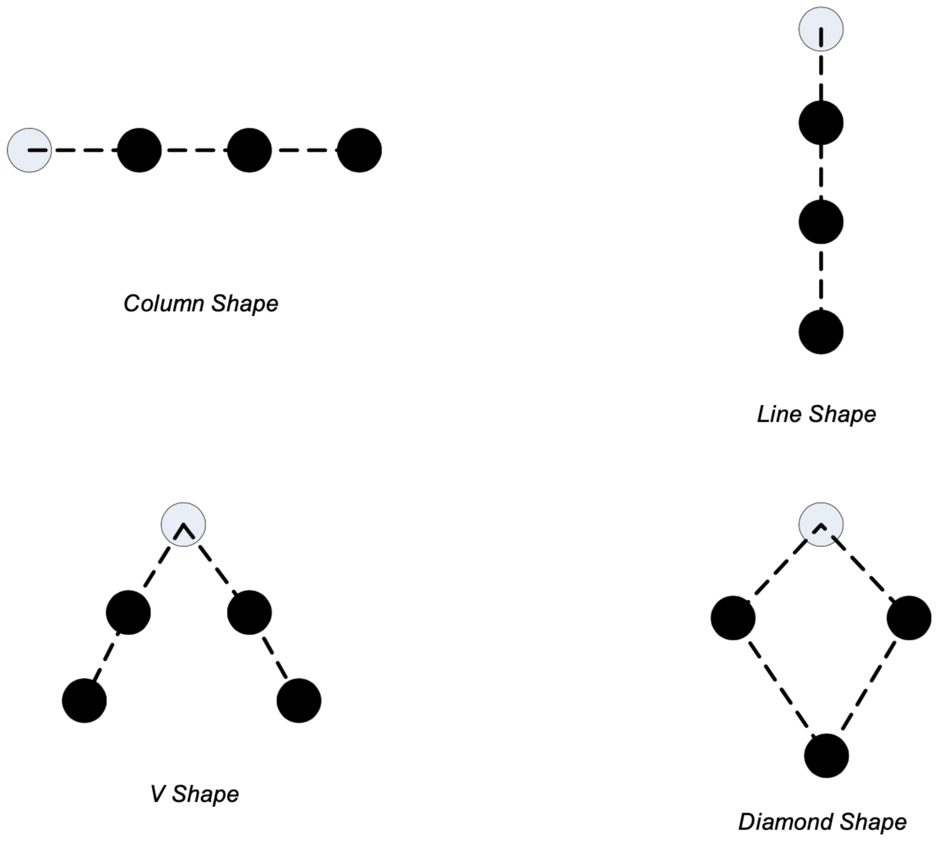


Figure 3: Patterns of formation.

3. Avoiding deadlock situation, vehicles should not block others' paths.

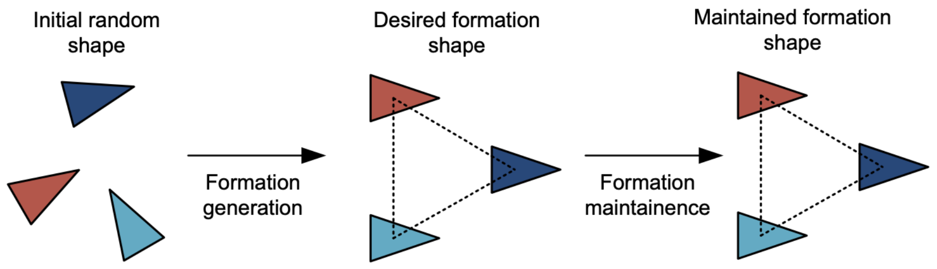
3.3 Review of formation control strategies

Also as summarised in [1], there exist three main types of formation maintaince according to different travel scenarios, as displayed in Figure 4:

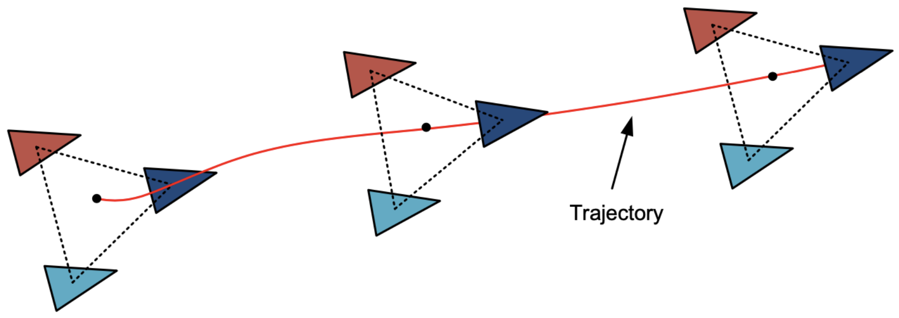
1. Formation generation and maintenance (Type1), the formation is formed when vehicles are located at random positions with arbitrary headings. Once attained, the shape needs to be maintained to continue the mission.
2. Formation maintenance during trajectory tracking (Type 2), formation needs to be rigorously maintained while following a predefined trajectory.
3. Formation shape variation and re-generation (Type 3), the formationshape is maintained as defined in Type2, but the shape also requires adjustment and re-generation while avoiding obsracles.

Three methods have been proposed to achieve formation achieve formation maintenance,

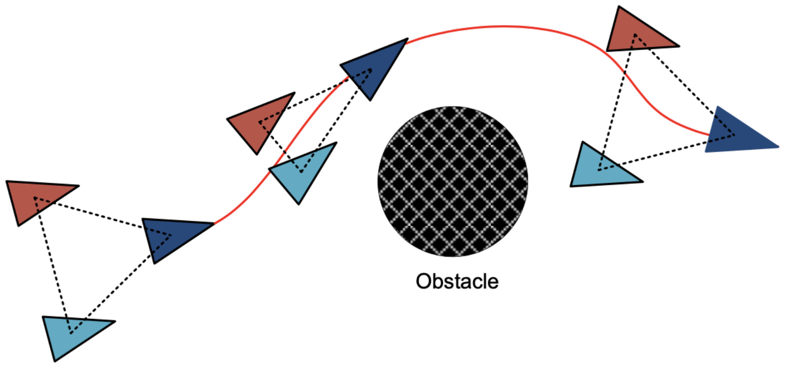
1. leader-follower. One vehicle is regarded as the group leader with full access to the overall navigation information and works as the reference vehicle in the formation. The leader could be virtual in some cases.
2. virtual structure. In this method, formation shape is treated as a virtual structure



(a)



(b)



(c)

Figure 4: Three different types of formation shape maintenance. (a) Formation generation and maintenance. (b) Formation maintenance while tracking trajectory. (c) Formation shape variation and re-generation.

Methods	Advantages	Disadvantages	Formation maintenance types	Platforms
Leader-follower[45;47;48]	1. Easy to be designed and implemented. 2. Efficient communication within the system	1. Highly dependent on the leader vehicle. 2. Lack of the feedback from the follower to the leader.	Type 1, Type 2 and Type 3	Widely adopted across various platforms
Virtual Structure[54–56]	1. Good performance in shape keeping. 2. Good representation of the relationship and the coordination between each vehicle in the formation.	1. Not flexible for shape deformation. 2. Not easy for collision avoidance	Type 1 and Type 2	Most applications seen on mobile robots. Less application on unmanned vehicles
Behaviour-based[57;59;61]	1. Capable of dealing with multi-task mission	1. Not easy to mathematically express the system behaviour. 2. Difficult to prove and guarantee the system stability.	Type 1, Type 2 and Type 3	Mobile robots and UGVs are two popular platforms

Figure 5: Comparison of formation control strategies.

or a rigid body. Then the formation is maintained by minimising the position error between the virtual structure and actual formation position.

3. behaviour-based methods. Control commands are based on various kinds of formation missions e.g. move-to-goal, avoid-static-obstacle, avoid-robot and maintain-formation. And a weight vector are assigned to each kind of command to generate the final control scheme.

Among three methods, leader-follower is the most widely adopted strategy, due to its simple design and implementation. While virtual structure method provides better performance in formation maintenance, but with limited capability to deal with collision avoidance since it has no flexibility to adjust the shape. The behaviour-based method is most adoptable approach, but lacks system stability analysis, making it unsuitable for large scale utilisation. The differences between methods are shown in Figure 5.

For future research, a hybrid control strategy appears to be the trend, also fault-tolerant formation control, i.e. capable of dealing with situations when the formation fails.

4 Cooperative Formation Path Planning

write this section later.

5 Formation Control using Reinforcement Learning

5.1 Leader-Follower Fromation using GQ(λ) Reinforcement Learning

M. Knopp, C. Aykin et al. proposed their method to train a leader-follower formation controller using GQ(λ) reinforcement learning in [4].

5.1.1 Framework

In their work, they train each agent to follow the closest agent in front of it. The learning process takes proximity sensors as states and motor speed as action.

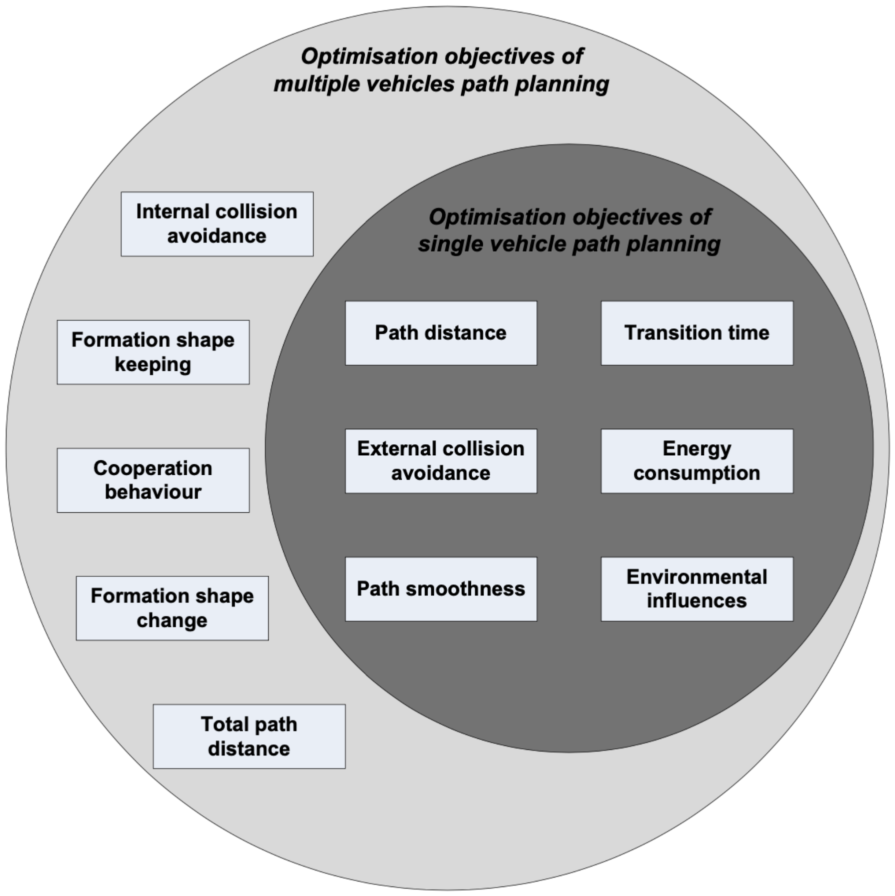
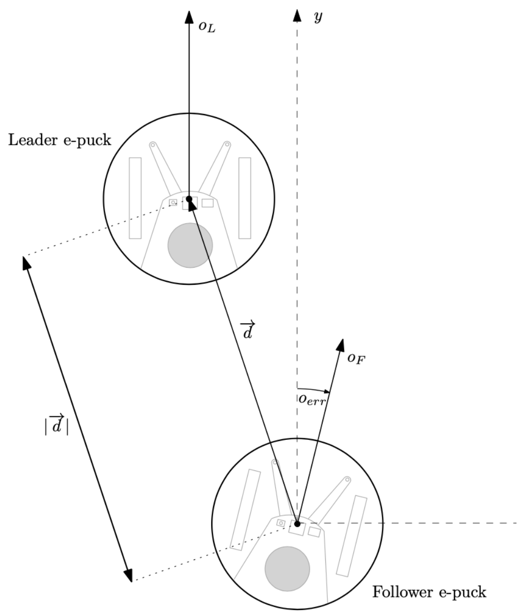
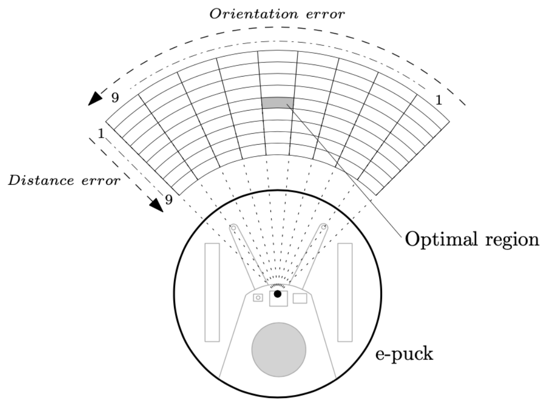


Figure 6: Comparison of optimisation objectives of single and multi agent path planning.



(a) Definition of state space.



(b) 9 × 9 tile coding of state.

Figure 7: States and tile coding of leader-follower using RL method.

They used the GQλ algorithm from Maei and Sutton [6] and combined it with an ϵ-greedy behaviour policy to balance between exploration and exploitation.

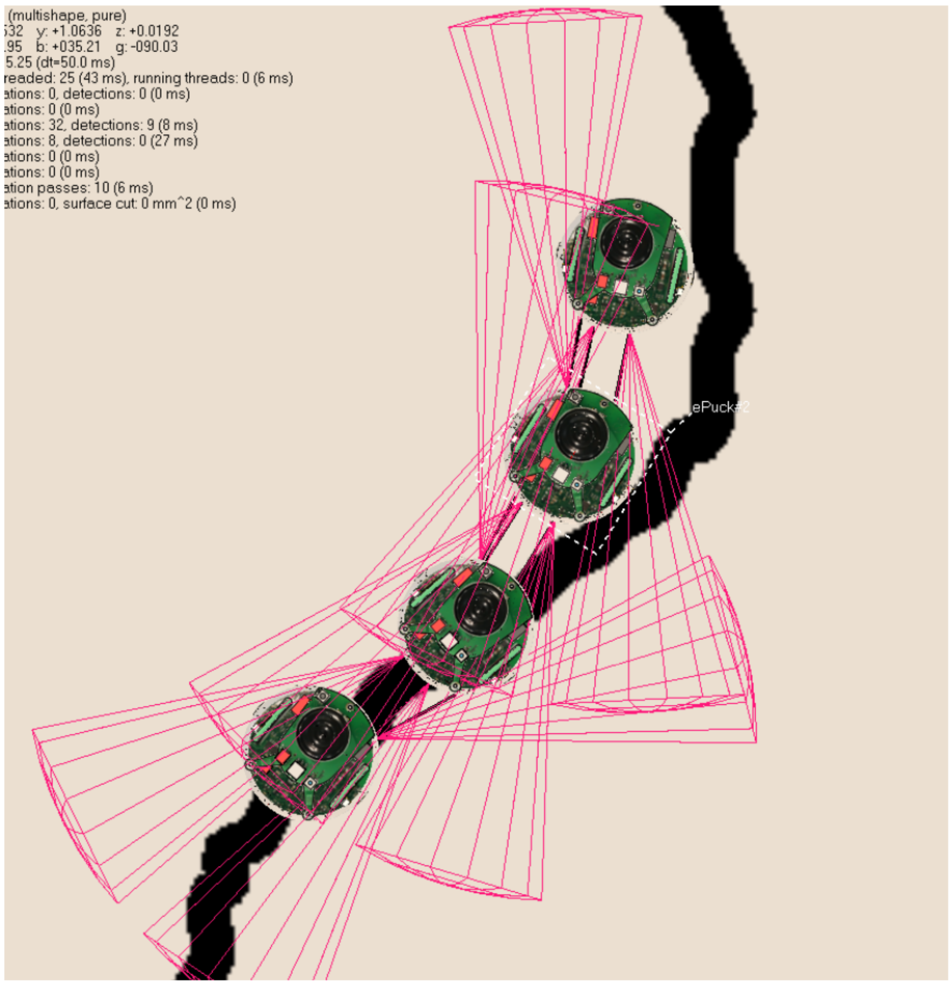
They build both a simulation using V-REP simulator and a real environment with e-puck robots.

5.1.2 State Representation

Their robots have two motors and the speed of each can be set independently to move in 2D, therefore the action space is two-dimensional. State space also has two dimensions, as the first dimensional the deviation from the optimal distance, and angular error as the second dimension, as shown in Figure 7(a). And to reduce ocmputational complexity for deployment on e-puck robots, action and state spaces are discretized with static binary single feature tile coding, in detail, uniform linear tile coding and linear tile coding respectively. The tile coding of state space is shown in Figure 7(b).

5.1.3 Reinforcements

The reinforcement process including its definition of reward function is not clearly stated is their paper. They might use the two errors in the state to form the reward function, or there might be a specified rewarding principle in the pipeline of GQλ algorithm, which remains to discover.



```
 (multishape, pure)
532   y: +1.0636   z: +0.0192
.95   b: +035.21   g: -090.03
 5.25 (dt=50.0 ms)
readed: 25 (43 ms), running threads: 0 (6 ms)
ations: 0, detections: 0 (0 ms)
ations: 0 (0 ms)
ations: 32, detections: 9 (8 ms)
ations: 8, detections: 0 (27 ms)
ations: 0 (0 ms)
ations: 0 (0 ms)
ation passes: 10 (6 ms)
ations: 0, surface cut: 0 mm^2 (0 ms)
```

Figure 8: Simple four agent line formation in V-REP,: the first leader can be found in the lower left corner. The red cones are showing active proximity sensors.

5.1.4 Experiment

In their experiment, the leader agent is programmed to follow a closed circular loop , which is controlled using no learning at all, and the other agents in the group is trained to follow the agent closest in front of them. They tested their algorithm both in the simulation and real environment. No quantitative evaluation is provided, but only graphic results, as shown in Figure 8 and 9.

5.1.5 Discussion

In this work, the sensor setting is more practical. Detection of agents in front of the current agents is a well developped technique, which can be done by a variaty of sensors, e.g. lidar or camera. Also, tile coding is a good method to discretize state and action space. Some shortcomings can also be summarised:

1. This work only explored how to design a leader-follower controller by reinforcement

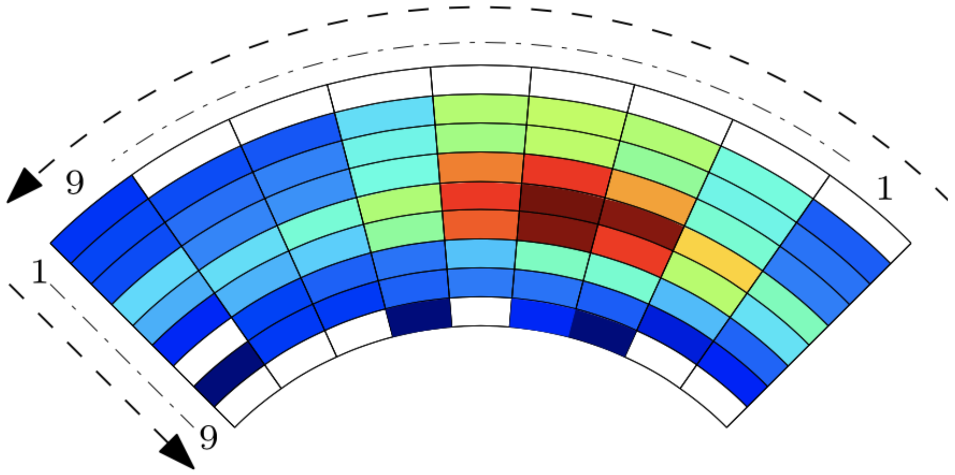


Figure 9: Distribution of visited states at around 1.3 million steps. In 74% of all cases, the leader was detected in the dark red area right to our optimal region. It stayed within the orange/red 3×3 area on the right side of the optimal region in 98.7% of the time. To make the rarely visited states discernible, a logarithmic scale is used: red and orange values are in the range of 30%–1%, green values are at around 0.1%, and blue values are another magnitude below that. The non-colored states were never visited in this experiment.

learning, but other formation shapes and maintenance strategy is not considered.

2. Also, environment obstacles either static or dynamic are ignored.

And they have their research updated in [5]. In this updated work, they use DQN instead of GQ(λ), and take images as input.

5.2 DRL-enhanced Behaviour-based Method

In [7], R. Johns proposed a method to enhance behaviour-based formationc control with deep reinforcement learning. In this work, the bahaviour-based method proposed by Balch and Arkin [8] is used as a comparison baseline and also the base from which the reinforment learning algorithms started their training.

5.2.1 Framework

In this method, the reinforment learning network starts with the behaviours of the traditional formation control algorithm, output a action vector $F_{i,rl}$, which is represented as a vector of forces applied to agent i to move. Then the final control vector $F_{i,control}$ is caculated as a combinition of the behaviour-based control algorithm's force $F_{i,bb}$ and $F_{i,rl}$. A single DRL model is used by all agents, as described in the section Parameter Sharing in [7], and displayed in Figure 10.

The algorithm is implemented using OpenAI's Baselines package [9], which is a set of state-of-the-art Python implementations of reinforcement learning algorithms running on

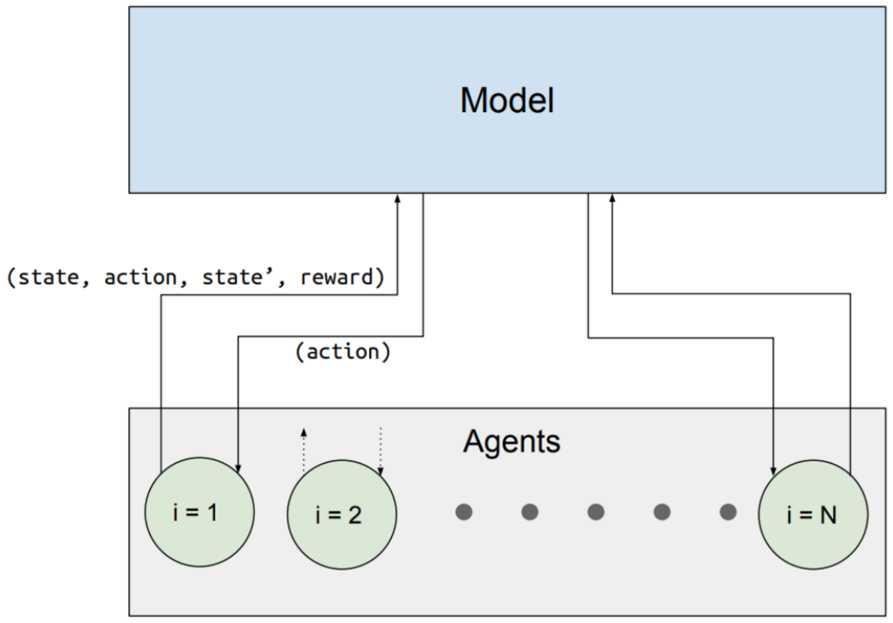


Figure 10: The parameter shared DRL model used by all agents.

top of Tensorflow.

5.2.2 State Representation

In the experiment of this method, each agent is equipped with 8 range sensors, and each lidar sensor is able to recognise if an obstacle or a neighbor agent is being observed. And the distance data is converted to sensor power based on a value mapping function as shown in Figure 11.

5.2.3 Reinforcements

The reward function is composed by four reward parts according to Equation 1.

$$R = R_{ARG} + R_{GRG} + R_{OC} + R_{FD} \tag{1}$$

1. R_{ARG}, i.e. Agent Reached Goal, is given to an agent reaching the goal.
2. R_{GRG}, i.e. Group Reached Goal, is awarded to the entire group if one group member reaches the goal.
3. R_{OC}, i.e. Obstacle Collision, added to teach agents to avoid obstacles.
4. R_{FD}, i.e. Formation Displacement, defined as the negative average sum of all agents' formation displacement.

5.2.4 Experiment

Expriment is carried on in a 2D simulation map, which is built with the help of OpenAI gym [10], and obstacles are represented black circles, and agents blue, shown in Figure 12.

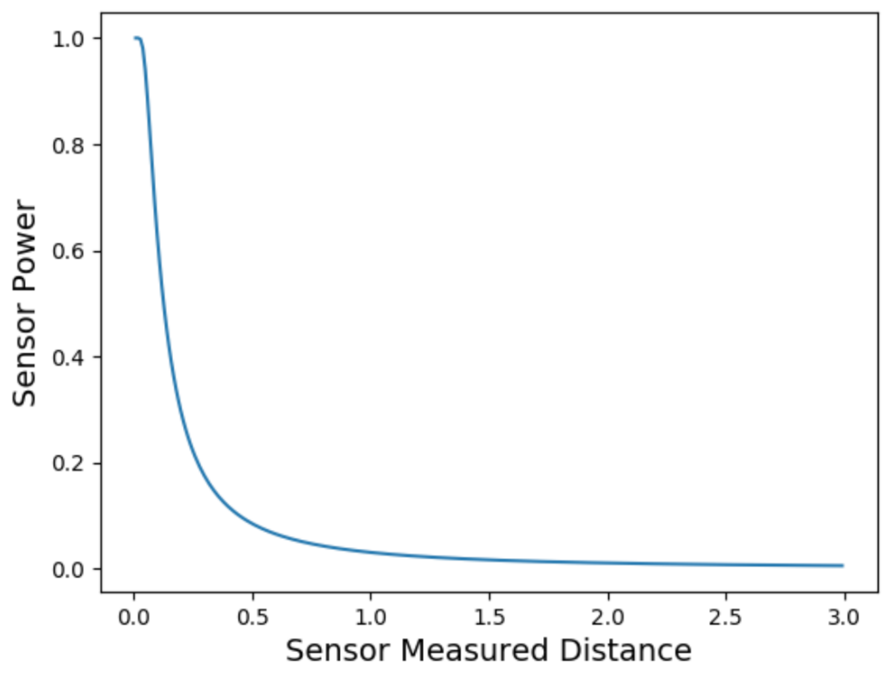


Figure 11: The relationship between the distance measured by a sensor and the sensor's power in an agents state space.

This work also establishes five evaluation metrics:

1. Path Ratio. The ratio of the average distance traveled by the agents divided by the Euclidean distance between the start and end point.
2. Obstacle Collision Frequency. Proportion of time the agents were in contact with an obstacle. A value of zero would imply no agent ever touched an obstacle during the episode, whereas a value of one would indicate that every agent was constantly touching an obstacle during the episode.
3. Average Formation displacement. A sense of how well the algorithm were at keeping the formation could be established, which is calculated as:

$$E_{FD} = \frac{1}{TN} \sum_{t=0}^{T} \sum_{n=0}^{N} |x_{n,desired}(t) - x(t)| \tag{2}$$

4. Success. The most important measurement was the percentage of successful attempts to reach the goal. Even if the goal position was not reached in a perfect formation or in the fastest time, the fact that the goal position was reached was considered crucial.
5. Iterations to Goal. To study how well the agent made use of its ability to move fast, the number of iterations needed to successfully reach the goal was monitored.

The method is tested in 3 experiment stages:

1. Fixed Obstacle Positions and Fixed Start and End Positions,
2. Fixed Obstacle Positions with Random Start and End Positions,
3. Random Obstacle Positions and Random Start and End Positions.

A sample learning curve is shown in Figure 14, and a sample result table in Figure 13.

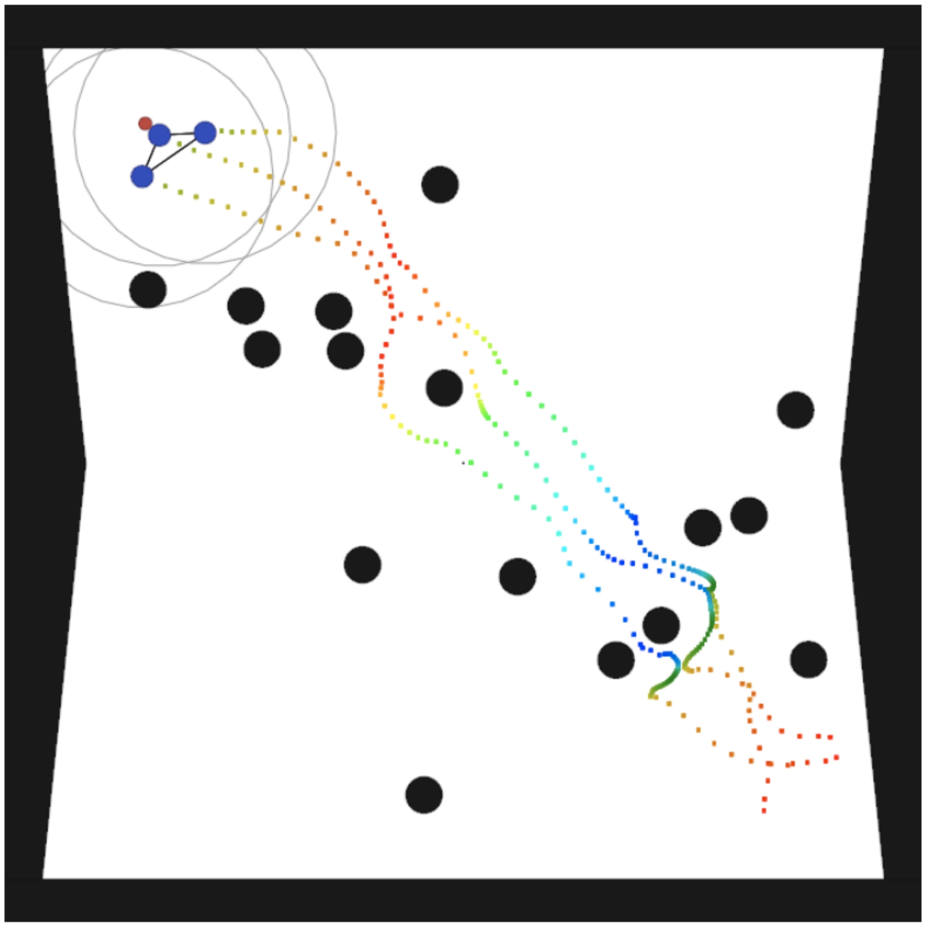
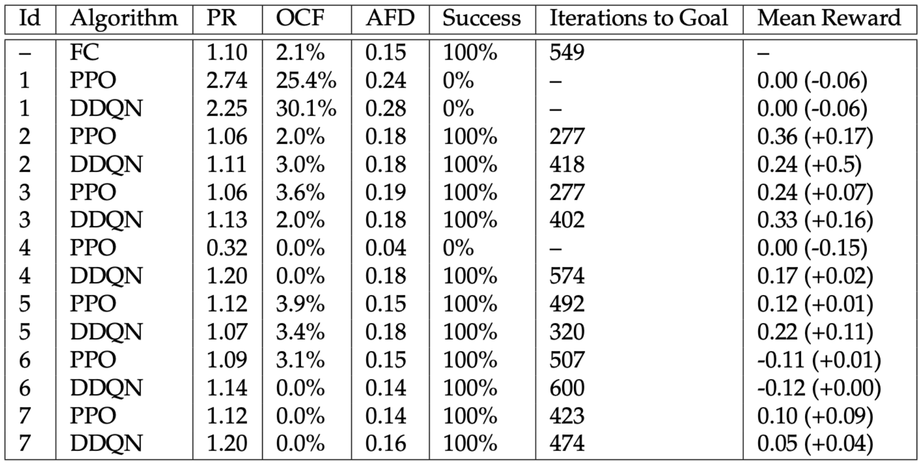


Figure 12: The experiment simulation environment of DRL-enhanced bahaviour-based method.



Id	Algorithm	PR	OCF	AFD	Success	Iterations to Goal	Mean Reward
–	FC	1.10	2.1%	0.15	100%	549	–
1	PPO	2.74	25.4%	0.24	0%	–	0.00 (-0.06)
1	DDQN	2.25	30.1%	0.28	0%	–	0.00 (-0.06)
2	PPO	1.06	2.0%	0.18	100%	277	0.36 (+0.17)
2	DDQN	1.11	3.0%	0.18	100%	418	0.24 (+0.5)
3	PPO	1.06	3.6%	0.19	100%	277	0.24 (+0.07)
3	DDQN	1.13	2.0%	0.18	100%	402	0.33 (+0.16)
4	PPO	0.32	0.0%	0.04	0%	–	0.00 (-0.15)
4	DDQN	1.20	0.0%	0.18	100%	574	0.17 (+0.02)
5	PPO	1.12	3.9%	0.15	100%	492	0.12 (+0.01)
5	DDQN	1.07	3.4%	0.18	100%	320	0.22 (+0.11)
6	PPO	1.09	3.1%	0.15	100%	507	-0.11 (+0.01)
6	DDQN	1.14	0.0%	0.14	100%	600	-0.12 (+0.00)
7	PPO	1.12	0.0%	0.14	100%	423	0.10 (+0.09)
7	DDQN	1.20	0.0%	0.16	100%	474	0.05 (+0.04)

Figure 13: A sample result of the experiments of DRL-enhanced Behaviour-based Method.

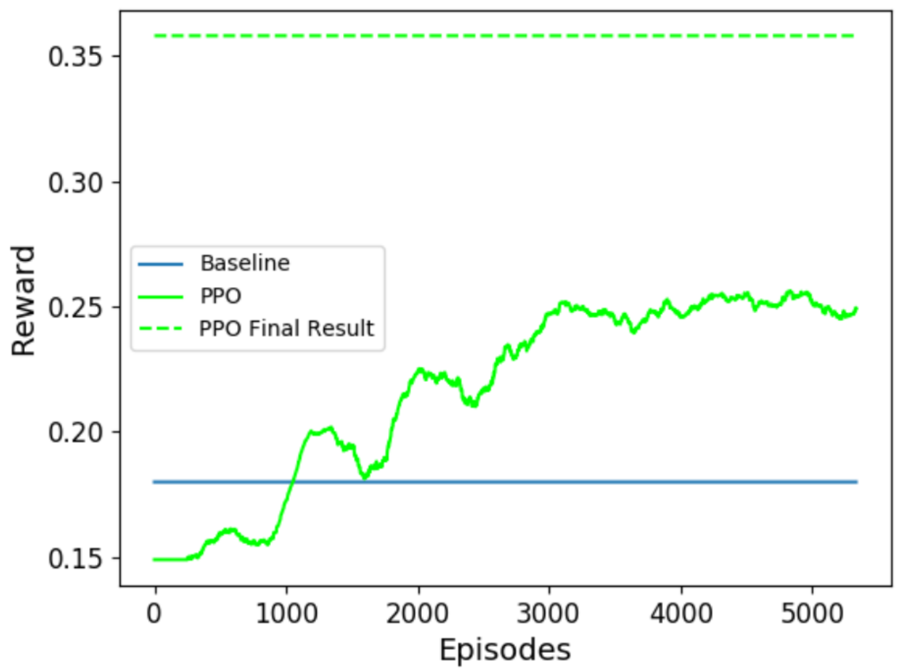


Figure 14: A sample learning cureve of the experiments of DRL-enhanced Behaviour-based Method.

5.2.5 Discussion

This work uses behaviour-based formation control as a base of the training of reinforcement learning network, makeing this method more stable and fast to converge. A few shortcomings of the method can be concluded:

1. The assignment problem is ignored, which means this algorithm has no capability to assign a agent to a new position in the formation during missions, and each agent can only maintain a fixed position, even swithing positions could be more efficient.
2. The state representation, the set of 8 sensors which can recognise obstacles or neighbor agents, is a set simplified as a simulation and appears to be difficult to realise in a real-world application.
3. This method employ deep reinforcement learning starting with behaviour-based method, makeing this method easier to train and more practical to apply, but also it introduce extra work to tune the behaviour-based method.

5.3 Coordinating Formation Control using Reinforcement Learning

In comparison of the two methods above, A. Rawat and K. Karlapalem [11] proposed another method using reinforcement learning to achive formation control. The contribution of their method is stated as exploring the formation control problem through a coordination based approach using reinforcement learning, i.e. instead of following a designated leader, agents are detecting neighbor agents and trained to coordinate with them to form a specific formation.

5.3.1 Framework

Following the work of the authors in [12, 13], the paradigm of centralized training with decentralized execution is used in this work. They use the actor-critic architecture with a central critic as shown in Figure 15.

This central critic [12] maps the true state of the system and the joint actions taken by the agents at that state to the expected future return. The true state of the system is defined as the pose information of all the agents with respect to the centroid of the current formation i.e. $[x_i^c, y_i^c, \theta_i^c]$, $\forall i \in V$ augmented with the position vector of the goal state with respect to the centroid. A centralized experience replay is built to store the tuple of past experiences $< s, r, s', u_1, ..., u_{|A|}, o_1, ..., o_{|A|}, o'_1, ..., o'_{|A|} >$, s' and o' are the next state of the environment and the observation of each agent in s'. To speed up learning, a experience modifier similar to [14] is used to change the goal state to the actual state achived by the agent. In this case, the goal is observed with respect to different observation frames and the input to the networks is a sequence containing those goals at different time-steps.

As for actors, each agent has its history of interactions with the environment which it uses to train its policy. The decentralized execution structure is illustrated in Figure 16.

5.3.2 State Representation

They leverage the notion of minimal rigidityand persistent graphs [15] to define the system. Each agent is denoted as a vortex $v_i \in V$ in a graph $\mathbb{G} = (V, E)$. There is a directed edge $e_{i,j} = (v_i, v_j) \in E$ between two agents v_i, v_j if and only if agent v_i observes agent v_j. If each of the agents for any shape of formation observes at least two other agents then a minimally rigid graph is formed.

5.3.3 Reinforcement

The agents are rewarded if they are able to maintain the specific distance with the agents that they can observe. A reward of greater magnitude is given if the agents are able to reach the goal while maintaining said formation. Three conditions are introduced as following, to define the reward function:

C1. Formation condition: If all the agents maintain their specified edge lengths within an error of ϵ_{form}.

C2. Collision condition: This condition is met if the distance between any two agents is less than some threshold (i.e. collision), ϵ_{col}.

C3. Success condition: If formation condition is satisfied and the centroid of the formation approaches the goal within a threshold distace ϵ_{goal}.

Then the reward function is defined as:

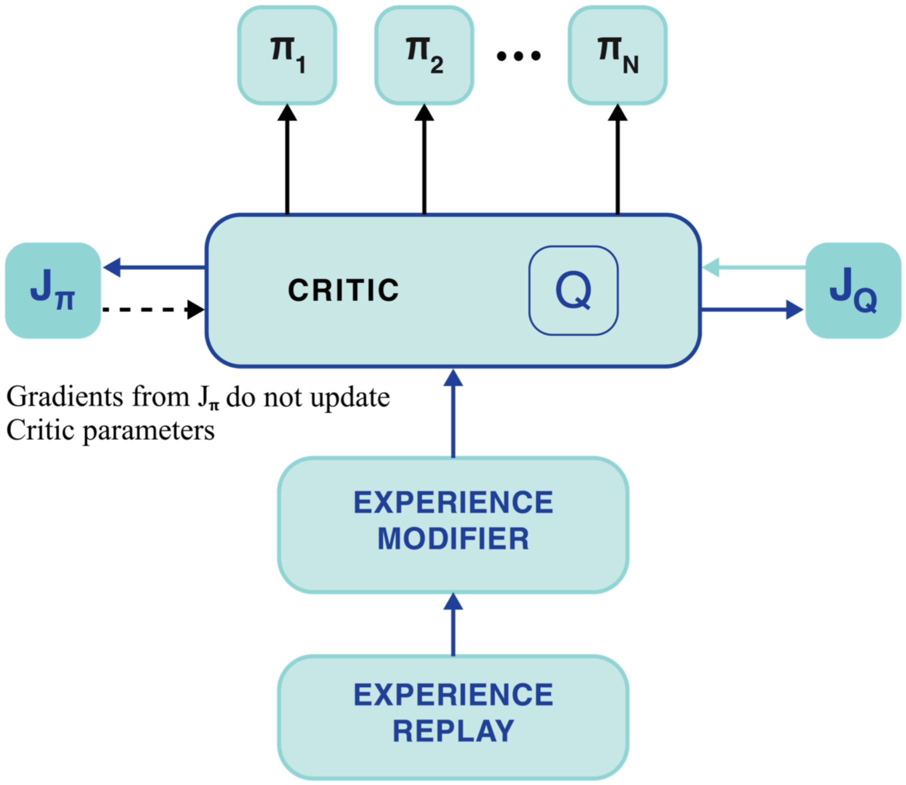


Figure 15: The training involves sampling a batch of experiences, then modifying some of the experiences for positive reinforcement.

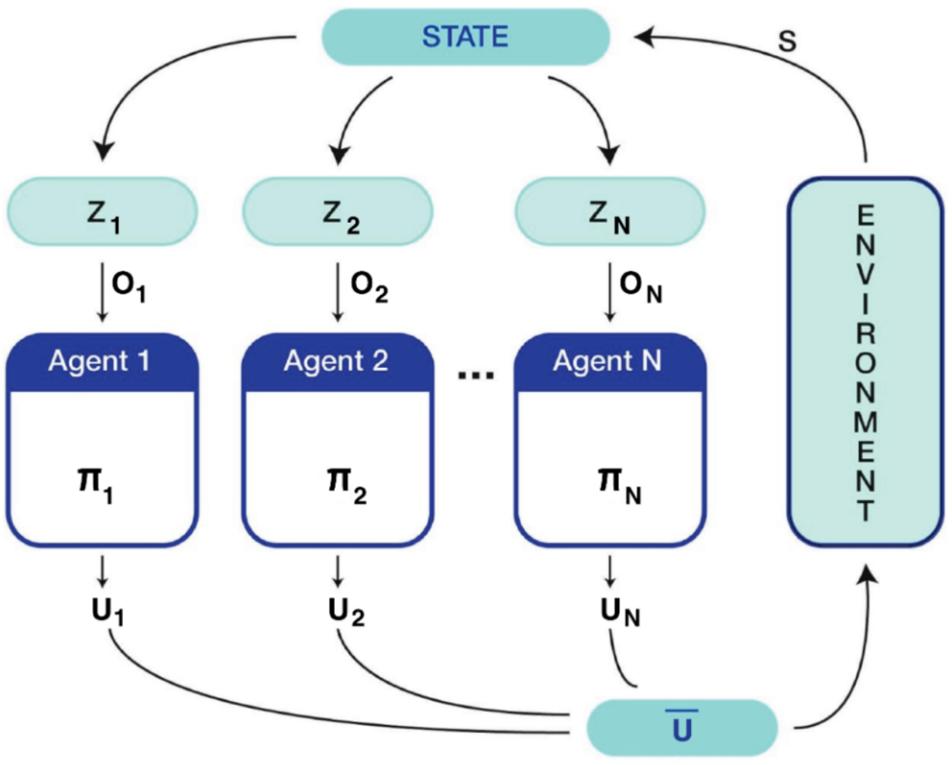


Figure 16: Interaction of agents with the environment. The state of the environment is observed through the observation function Z_i of the agent i. The agent acts according to its policy π_i. The actions of all the agents $\bar{U}$ transitions the environment to some new state.

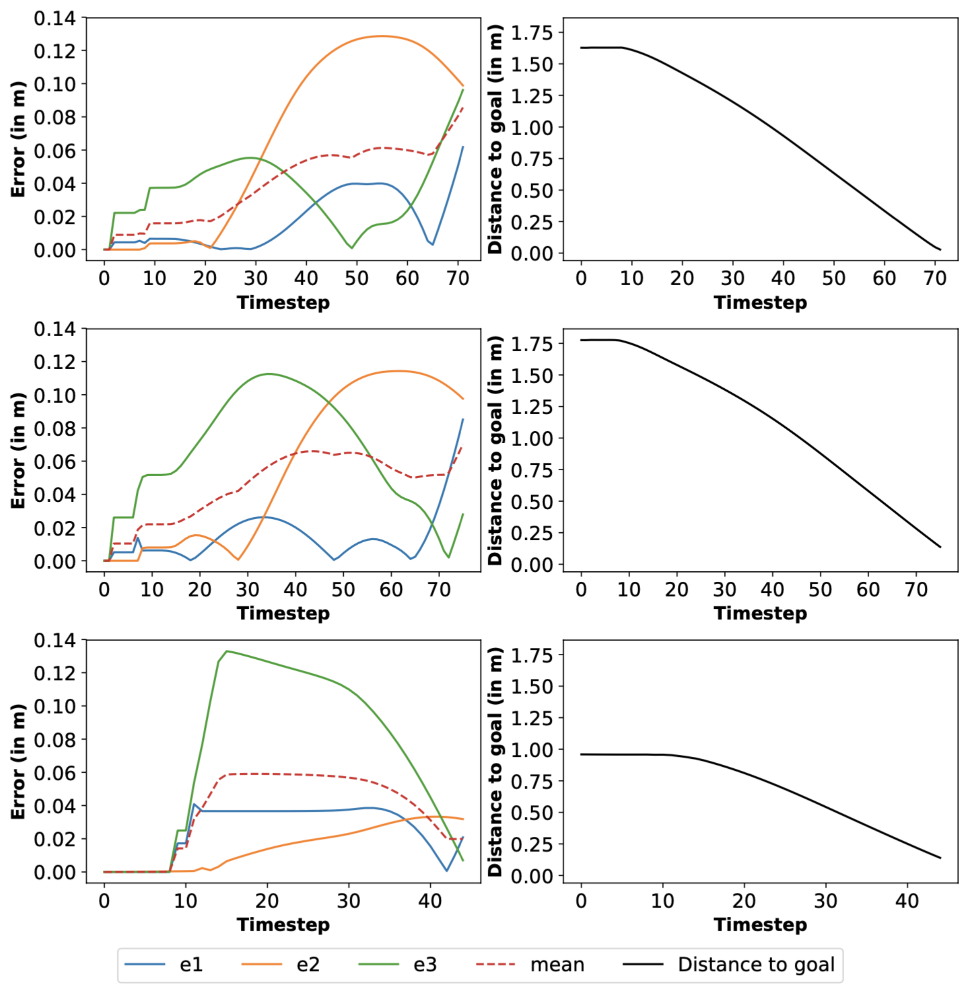


Figure 17: The plots on the left show the error in each edge of the formation. The plots on the right show the distance between the centroid of the formation and the goal. The edges between agents are denoted by e1, e2 and e3. The dotted red line denotes the mean error of all the edges.

$$R(s,a,s') = \begin{cases} r_{edge}, & if\ only\ C1\ is\ true \\ r_{collision}, & if\ C2\ is\ true \\ r_{goal}, & if\ C3\ is\ true \\ r_{penalty}, & otherwise \end{cases} \tag{3}$$

5.3.4 Experiment

The experiment is executed in a custom environment made with the help of OpenAI gym [10]. The paper illustrated the error in each edge of the formation, and the distance of the centroid from the goal with respect to time for the corresponding trial, as shown in 17, also the learning curve is shown in Figure 18.

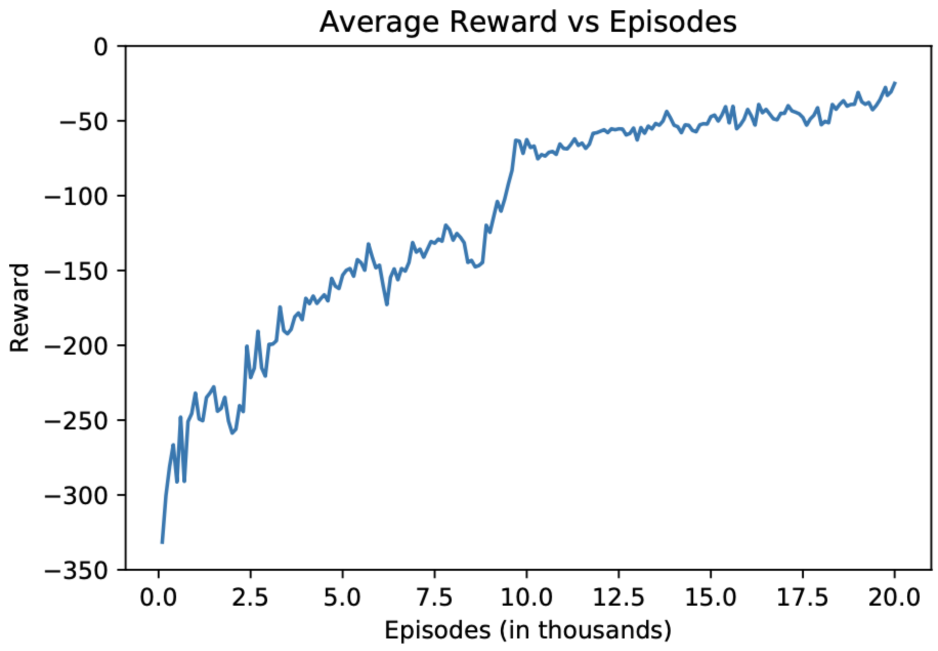


Figure 18: Average reward per episode vs. number of training episodes.

5.3.5 Discussion

Compared with the papers reviewed above, this method starts to use the structure of multi-agent reinforcement learning. The combinition of a centralized critic and decentralized actors is a good application of multi-agent reinforcement learning technique. Because it uses reinforcement learning for both target reaching and formation control, this method requires less workload to design the controller. But also shortcomings can be concluded:

1. It did not consider the problem of obstacle avoidance,
2. This method train the deep RL network with no traditional formation control algorithm as a base, so it is suppose to require dramatically more episodes to train until at least a applicable result is attained. But the reward function definitions and full trainnig processes differ significantly between methods reviewed in this article, this possible shortcoming remains for furthur research.

References

[1] Y. Liu and R. Bucknall, “A survey of formation control and motion planning of multiple unmanned vehicles,” *Robotica*, vol. 36, no. 7, pp. 1019–1047, 2018.

[2] Y. Liu and R. Bucknall, “Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment,” *Ocean Engineering*, vol. 97, pp. 126 – 144, 2015.

[3] S. Campbell, W. Naeem, and G. W. Irwin, “A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres,” *Annual Reviews in Control*, vol. 36, no. 2, pp. 267–283, 2012.

[4] M. Knopp, C. Aykın, J. Feldmaier, and H. Shen, “Formation control using gq (λ) reinforcement learning,” in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 1043–1048, IEEE, 2017.

[5] C. Aykin, M. Knopp, and K. Diepold, “Deep reinforcement learning for formation control,” in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–5, IEEE, 2018.

[6] H. R. Maei and R. S. Sutton, “Gq (lambda): A general gradient algorithm for temporal-difference prediction learning with eligibility traces,” in *3d Conference on Artificial General Intelligence (AGI-2010)*, Atlantis Press, 2010.

[7] R. J. Johns, “Intelligent formation control using deep reinforcement learning,” 2018.

[8] T. Balch and R. C. Arkin, “Behavior-based formation control for multirobot teams,” *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.

[9] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, “Openai baselines,” 2017.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

[11] A. Rawat and K. Karlapalem, “Multi-robot formation control using reinforcement learning,” *arXiv preprint arXiv:2001.04527*, 2020.

[12] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[13] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Advances in neural information processing systems*, pp. 6379–6390, 2017.

[14] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, “Hindsight experience replay,” in *Advances in neural information processing systems*, pp. 5048–5058, 2017.

[15] J. M. Hendrickx, B. D. Anderson, and V. D. Blondel, “Rigidity and persistence of directed graphs,” in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 2176–2181, IEEE, 2005.