2021 The 21st International Conference on Control, Automation and Systems (ICCAS 2021)
Ramada Plaza Hotel, Jeju, Korea, Oct. 12∼15, 2021

End-to-End control of USV swarm using graph centric Multi-agent Reinforcement Learning

Kanghoon Lee [1], Kyuree Ahn[1] and Jinkyoo Park [1*]

[1]Department of Industrial and Systems Engineering, KAIST, Korea
{leehoon ; ahnkjuree ; jinkyoo.park}@kaist.ac.kr * Corresponding author

Abstract: The Unmanned Surface Vehicles (USVs), which operate without a person at the surface, are used in various naval defense missions. Various missions can be conducted efficiently when a swarm of USVs are operated at the same time. However, it is challenging to establish a decentralised control strategy for all USVs. In addition, the strategy must consider various external factors, such as the ocean topography and the number of enemy forces. These difficulties necessitate a scalable and transferable decision-making module. This study proposes an algorithm to derive the decentralised and cooperative control strategy for the USV swarm using graph centric multi-agent reinforcement learning (MARL). The model first expresses the mission situation using a graph considering the various sensor ranges. Each USV agent encodes observed information into localized embedding and then derives coordinated action through communication with the surrounding agent. To derive a cooperative policy, we trained each agent's policy to maximize the team reward. Using the modified prey-predator environment of OpenAI gym, we have analyzed the effect of each component of the proposed model (state embedding, communication, and team reward). The ablation study shows that the proposed model could derive a scalable and transferable control policy of USVs, consistently achieving the highest win ratio.

Keywords: USV swarm, Multi-agent reinforcement learning, Graph Neural Network,

2021 21st International Conference on Control, Automation and Systems (ICCAS) | 978-89-93215-21-2/21/$31.00 ©2021ICROS | DOI: 10.23919/ICCAS52745.2021.9649839

1. INTRODUCTION

Unmanned Surface Vehicles (USVs) can be operated without human operations to support various naval defense missions, such as surveillance, perimeter monitoring, and search and rescue. Although various missions can be conducted more efficiently when a swarm of USVs are operated at the same time, it is challenging for USVs to make collective decisions to achieve the goal for several reasons. Firstly, each USV has only partial observations of the overall situation. For multiple USVs to make collective decisions, they should be able to grasp the whole situation. Therefore, it is necessary to extract important features through information sharing. Secondly, each USV must be assigned to an appropriate role. For example, to defend a large area, a group must be divided into several subgroups, each defending a different area. Making good decisions using centralized decision-making is difficult because the joint action space increases exponentially to the number of agents. It necessitates a process that makes decentralized but cooperative decisions. Lastly, it is difficult for the control policy to determine the action of the USV. For example, the number of allies and enemy USVs, and the surrounding environment, such as islands, may differ among missions. Therefore, the decision-making process should be scalable and transferable.

In this paper, we propose an algorithm to derive decentralised and cooperative control policies for the USV swarm to accomplish surveillance and combat mission. We first represent the mission situation as a graph to efficiently encode the global mission status into the localized information of an individual USV. The encoded graph node embedding value is fed into the policy of the individual agent to determine an action of the USV. By conducting both representational learning and policy learning in an end-to-end manner, we can derive a cooperative and scalable control policy for the USV swarm to conduct surveillance and combat mission.

2. BACKGROUNDS

2.1 Multi-agent Reinforcement Learning

We consider a decentralized partially observable Markov decision process (Dec-POMDP) framework [1]. Dec-POMDP, which consist of $< N, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{R}}, T >$, is defined as follows: N denotes the number of agents, which is a decision-making object; $\boldsymbol{s_t} \in \boldsymbol{\mathcal{S}}$ denotes the global state of the game; $o^i : \boldsymbol{\mathcal{S}} \mapsto \mathcal{O}^i$ denotes the partial information of the global state for agent i; $a^i \in \mathcal{A}^i$ denotes the action of agent i; $r^i : \boldsymbol{\mathcal{S}} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$ denotes the reward for agent i; T denotes the transition probability function. A multi-agent reinforcement learning (MARL), which is a multi-agent extension of reinforcement learning (RL), is a method that solves Dec-POMDP. In MARL, each agent learns a policy that makes an action with its observation to maximize its expected discounted reward. Agents can cooperate to achieve a common goal or compete to achieve their respective goals, depending on the problem situation. Also, they can cooperate internally and compete externally as a team-to-team setting.

2.2 Graph Neural Network

A graph neural network (GNN) is a neural network designed to process graph-structured input data. GNN uses the graph structure and node features as input, and outputs the updated node features while considering the pairwise relationship between nodes within the graph. As

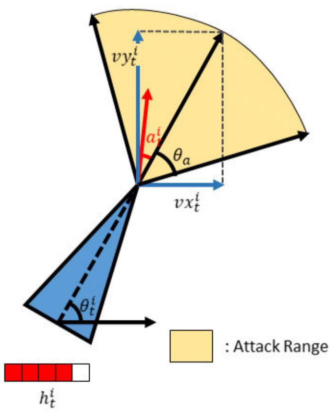

Fig. 1. Visual description of the state of an individual USV.

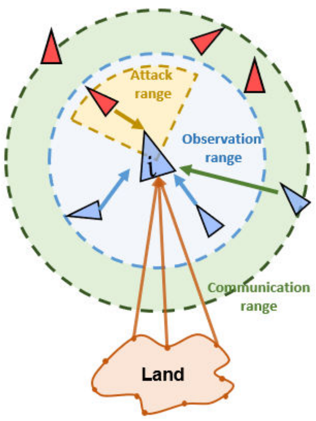

Fig. 2. Visual description of edge connectivity on the perspective of agent i.

GNN recursively computes and propagates messages between two nodes, information from each node is propagated to the neighboring node. GNN has the advantage that it is invariant to the input graph size or structure because it is calculated based on the node-to-node relationship. We use message-based GNNs [2] among many variants [3], [4].

3. FORMULATION

In this section, we formulate the USV swarm control problem in an Dec-POMDP framework. The Dec-POMDP consists of $< N, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{R}}, T >$, which are defined as follows:

- Agents, N: We define an individual USV as a decision making agent. The individual agent will be indexed with $i \in 1, \dots, N$.
- State $\boldsymbol{\mathcal{S}}$: The global state $\boldsymbol{s_t} \in S$ is a set of individual agents' state, $\boldsymbol{s_t} = (s_t^1, \dots, s_t^N)$, where s_t^i is the state of agent i. As shown in Fig. 1, the state s_t^i of agent i is defined as $s_t^i = (x_t^i, y_t^i, vx_t^i, vy_t^i, \theta_t^i, h_t^i, b_t^i)$.
  - (x_t^i, y_t^i) are coordinates of agent i at time t,
  - (vx_t^i, vy_t^i) are the x and y components of the velocity at time t,
  - θ_t^i is the yaw angle of agent i at time t
  - h_t^i is the health level of agent i at time t,
  - b_t^i is the number of bullets remaining.
- Observation $\boldsymbol{\mathcal{O}}$: We assume that each agent observes a partial information of the global state, $o_t^i = o^i(\boldsymbol{s_t})$. Each agent can observe the state of the ally and enemy agent within a circular observation range (Fig. 2).
- Action $\boldsymbol{\mathcal{A}}$: The joint action $\boldsymbol{a_t} = (a_t^1, \dots, a_t^N) \in \boldsymbol{\mathcal{A}}$ is the product of the actions of the individual agents. The action of agent i at time t, a_t^i is defined as the amount of angle to rotate at timestep t, as shown in red arrow in Fig. 1.
- Reward $\boldsymbol{\mathcal{R}}$: The rewards of the individual agent is a function of state, action, and next state, $r^i(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{s_{t+1}})$. Each agent receives a positive reward if the agent attacks (+10) or adversarial agent is dead (+50), and receives -1 otherwise.

The objective of the USV swarm problem is to find a set of individual policies, $\mu(o_t^i)$, that maximizes an expected cumulative sum of individual reward, $\mathbb{E}_{s_i,a_i}[\sum_t r^i(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{s_{t+1}})]$. As all the agents are trained to maximize the team reward jointly (i.e., the sum of the individual rewards), the derived decentralized policy can be used to conduct the cooperative combat task.

4. METHODOLOGY

4.1 Graph Construction

We represent a global state using a graph to explicitly define the relationship between the agents. The graph nodes are composed of three type of nodes, V_a, V_e and V_l, where V_a is a set of agent nodes, V_e is a set of adversarial agent nodes, and V_l is a set of land nodes. The initial node feature is defined as the individual state of the node, $v_i^{(0)} \leftarrow o_i$.

The edges of the graph represents the relationship between two nodes. We use different type of edges to represent the different types of relationships between two nodes. The type of edge will determine the type of neural network to update the edge feature. Thus, edges of the same type learn how to process message from the source node to the destination node in the same way.

We define following sets to indicate the node index connected to agent i. Each set of nodes are defined as the nodes within the sensor scope of agent i.

- $V_o(i)$: The index of nodes in the observation scope of agent i.
- $V_c(i)$: The index of agent nodes within the communication scope of agent i.
- $V_e(i)$: The index of adversarial agent nodes within the attack range of agent i.
- $V_l(i)$: The index of land nodes. We assume that every agent possesses the prior knowledge about land location and, therefore, the land indices are not time-varying.

With nodes within the different sensor range defined, the edge connectivity is defined accordingly. The edges connecting nodes in the same sensor range are defined as edges of the same type. Fig. 2 illustrates an exam-

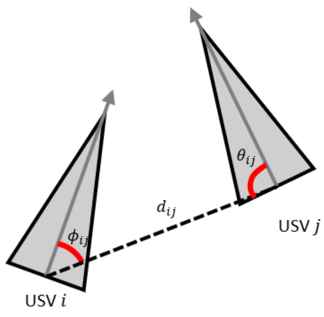

Fig. 3. Example of the relative orientation and bearing angle of agent i with respect to agent j.

ple of the edge connectivity in the perspective of agent i. Note that two nodes may have multiple relationships. For example, if the distance between two agent nodes is less than both observation and communication range, the two agents are connected with both the observation and communication edges.

The initial edge feature is defined to represent the relative information from the source node to the destination node. The edge feature includes the distance and the relative angle between two nodes, which was first suggested in [5]. Fig. 3 illustrates the example of the edge feature between two neighboring agents:

- Distance between two nodes,

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- Bearing to neighboring node,

$$\phi_{ij} = \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) - \theta_i$$

- Relative orientation,

$$\theta_{ij} = \arctan\left(\frac{y_i - y_j}{x_i - x_j}\right) - \theta_j$$

4.2 State representation using graph neural network

In order to efficiently process the observation of the individual agent and make communication between the agents, we use the message passing GNN on a global state graph. A single layer of GNN is composed of edge update, node aggregation, and node update. An edge update first computes the edge embedding using the source node feature, destination node feature, and edge feature. After computing the updated edge feature, each node aggregates the edge embeddings from the incoming edges. The aggregated node embeddings are fed into a neural network to produce the updated node features.

Our state representation module is composed of two layers: a state embedding layer and a message embedding layer. The state embedding layer updates the node feature of each agent by processing messages from neighboring nodes in the observation and attack range:

$$v_i^{(l+1)} = f_v^s([v_i^{(l)} \| \sum_{j \in V_o(i)} m_{ji} \| \sum_{j \in V_e(i)} m_{ji} \| \sum_{j \in V_l(i)} m_{ji}]; \theta_v^s) \tag{1}$$

where $\|$ is a concatenation operator, l is index of the hop , and $f_v^s(\cdot;\theta_v^s)$ is a node update function. m_{ji} is the message from node v_j to node v_i as follows:

$$m_{ji} = f_e^s([v_i \| v_j \| e_{ji}]; \theta_e^s) \tag{2}$$

where $f(\cdot;\theta_m^V)$ is a message update function.

The communication layer is designed to share the individualized state information to coordinate with agents in the communication range. The computation of the communication layer is same as the computation of the state embedding layer, except for the type of edges used to compute message. The communication layer aggregates messages coming from the communication edges as follows:

$$v_i^{(l+1)} = f_v^c\left([v_i^{(l)} \| \sum_{j \in V_c(i)} m_{ji}]; \theta_v^c\right) \tag{3}$$

By communicating with other agents using the processed state embedding, agents can obtain the condensed representation of the global state from the perspective of each agent.

4.3 Decentralized decision making

Once the node embedding value has been computed, the shared policy network maps the node embedding value to a control action for each agent, $\mu(v_t^i;\theta_\mu)$. In order to do a better exploration, we add Gaussian noise to the action at training time, $a_t^i = \mu(v_t^i;\theta_\mu) + \mathcal{N}$.

5. TRAINING MARL POLICY

The objective of the USV swarm problem is to maximize the sum of the agent's rewards (team reward), and thus increase the system-level performance. When training the decentralized policy for such a team game, a typical MARL training approach is the centralized training and decentralized execution (CTDE) [6]. In such CTDE framework, the agent learns the centralized critic value using the global observation and derives the decentralized policies using only the local observations.

Following the idea of the CTDE approach, our training objective is to learn a joint state-action value, $Q(\boldsymbol{s_t}, \boldsymbol{a_t}) = \mathbb{E}_{a_t^i \sim \mu^i}[\sum_t \sum_i r^i(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{s_{t+1}})]$. Instead of training the joint critic directly, we assume that the joint state-action value can be decomposed as the sum of individual state-action value, $Q(\boldsymbol{s_t}, \boldsymbol{a_t}) \approx \sum_i Q(h_t^i, a_t^i)$, and train $Q(h_t^i, a_t^i)$ for each agent.

To train the policy of agent, we use deep deterministic policy gradient (DDPG, [7]), which is an actor-critic algorithm designed for continuous action space. The critic network, $Q(h_t^i, a_t^i;\theta_Q)$ and actor network, $\mu(h_t^i;\theta_\mu)$ are both composed of GNN+MLP. We denote the parameter of the critic and actor by θ_μ and θ_Q respectively.

A critic network, θ_Q is updated to minimize the error of the estimated Q-value as follows:

$$\mathcal{L}(\theta_Q) = \mathbb{E}_{s,a,r\sim\mathcal{D}}\left[\sum_i \left(Q(h_t^i, a_t^i; \theta_Q) - y_t^i\right)^2\right], \quad (4)$$

$$y_t^i = r_t^i + \gamma \cdot Q(h_{t+1}^i, \mu(h_{t+1}^i; \theta_\mu); \theta_Q), \quad (5)$$

$$\theta_Q \leftarrow \theta_Q - \alpha \cdot \nabla_{\theta_Q}\mathcal{L}(\theta_Q) \quad (6)$$

An actor network, θ_μ is updated as follows:

$$\nabla_{\theta_\mu}\mathcal{J}(\theta_\mu) = \mathbb{E}_{s\sim\mathcal{D}}\left[\sum_i \nabla_{a_t^i}Q(h_t^i, a_t^i; \theta_Q)\nabla_{\theta_\mu}\mu(h_t^i; \theta_\mu)\right] \quad (7)$$

$$\theta_\mu \leftarrow \theta_\mu + \beta \cdot \nabla_{\theta_\mu}\mathcal{L}(\theta_\mu) \quad (8)$$

The policy gradient combined with GNN enables end-to-end training of the state representation learning and policy learning. The agents use shared parameter, as we assume heterogeneity between agents.

6. EXPERIMENTS

6.1 Experiment setups

We validate the performance of the proposed method using the modified prey-predator environment of OpenAI gym [8] (Fig. 4). In both training and testing, we set the number of agents as 3, and the number of adversary agents as 5. The initial position of the agents and adversarial agents are randomly generated in the map. The speed of the adversary agent USV is 0.2 times the speed of the agent USV. The central angle of the circular sector attack range is $\pi/4$. The adversary agents move toward to the landmark starting from the initial position heuristically.

We trained 20,000 episodes for training, each of which consists of 200 timesteps. The hyperparameter details for the model and experiments are listed in Table 1.

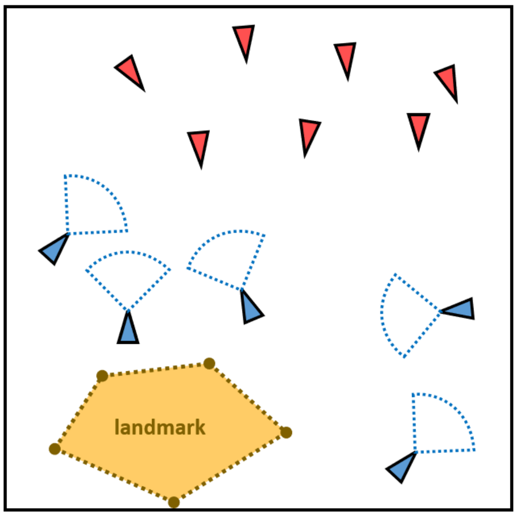

Fig. 4. Illustration of the modified prey-predator environment

Hyperparameter	Value
Optimizer	Adam optimizer
Critic learning rate, α	0.001
Actor learning rate, β	0.0001
Repay memory size	100,000
Batch size	512
Discount factor	0.8
Soft target update rate, τ	0.001
Noise	Gaussian noise, $\sigma = 0.7$

Table 1. List of model hyperparmeters

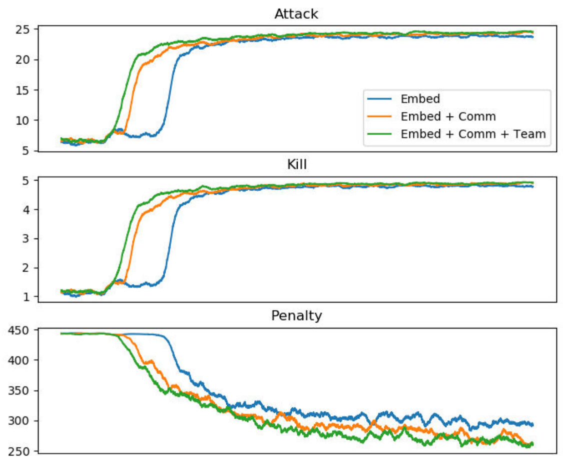

Fig. 5. Performance curve

We compared the performance of following models to see how each component of model affects the cooperation of the agents. Three models are compared to analyze the performance of the model:

- Embed: Only the state embedding layer is used to make observation embedding of policy.
- Embed + Comm: Communication layer is used in addition to the embedding layer.
- Embed + Comm + Team: In addition to the original reward function, we add a team reward to enhance the cooperation between agents. The team reward is defined as the summation of all the agent's rewards on the same team, multiplied by a regularizer factor less than 1.

Fig. 5 shows the performance of three different models. It shows that the performance improves as a component is added to the model, especially, when the communication layer was used. This indicates that the communication layer is essential to achieve the goal. In the learning process, the team reward led to a slight performance improvement.

We conducted two evaluations with the trained models. Each experiment shows how well the trained model performs when the number of agents and adversarial agents changes. Tables 2-3 shows the win ratio of the trained model in each scenario. The reported win ratio was obtained through 50 tests each. Table 2 shows that the win rate of all models decreases as the number of adversarial agents increases. The model trained using the team reward always achieve the highest win rates than models without the team reward. Therefore it supports the fact that the model using team reward achieves the goal better by dividing the roles of each agent well, even when there are many adversarial agents. Table 3 shows that the win rate of all models increases as the number of agents increases. The model trained using team reward shows lots of performance improvement compared to the models as the number of agents increased. One noteworthy point is that as the number of agents increased, the communication layer did not affect the performance improvement. Both experiments indicate that the team rewards are essential to achieve the goal of the swarm.

	5	7	10	15	20
Embed	98%	94%	84%	66%	40%
+Comm	96%	98%	88%	58%	48%
+Team	**100%**	**98%**	**94%**	**80%**	**62%**

Table 2. Win ratio when the number of adversarial agents are changing (the number of agents is fixed as 3).

	3	4	5	6	7
Embed	40%	58%	62%	80%	82%
+Comm	48%	56%	62%	78%	70%
+Team	**64%**	**80%**	**92%**	**92%**	**100%**

Table 3. Win ratio when the number of agents are changing (the number of adversarial agents is fixed as 20).

7. CONCLUSION

We established a cooperative decision making process for the USV swarm system using a graph-centric multi agent reinforcement learning. We represented the global state as a graph with multiple node/edge types. The GNN conducted both state embedding and communication to achieve an individualized node embedding of each agent. The experimental results show that the suggested model could derive a cooperative and decentralized control policy for the USVs for various number of agents.

ACKNOWLEDGEMENT

This research was supported by Future Challenge Program through the Agency for Defense Development funded by the Defense Acquisition Program Administration.

REFERENCES

[1] Bernstein, Daniel S., et al. ”The complexity of decentralized control of Markov decision processes.” Mathematics of operations research 27.4 (2002): 819-840.

[2] Battaglia, Peter W., et al. ”Relational inductive biases, deep learning, and graph networks.” arXiv preprint arXiv:1806.01261 (2018).

[3] Veličković, Petar, et al. ”Graph attention networks.” arXiv preprint arXiv:1710.10903 (2017).

[4] Kipf, Thomas N., and Max Welling. ”Semi-supervised classification with graph convolutional networks.” arXiv preprint arXiv:1609.02907 (2016).

[5] Hüttenrauch, Maximilian, Sosic Adrian, and Gerhard Neumann. ”Deep reinforcement learning for swarm systems.” Journal of Machine Learning Research 20.54 (2019): 1-31.

[6] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Advances in Neural Information Processing Systems, pages 6379, 2017.

[7] Lillicrap, Timothy P., et al. ”Continuous control with deep reinforcement learning.” arXiv preprint arXiv:1509.02971 (2015).

[8] Brockman, Greg, et al. ”Openai gym.” arXiv preprint arXiv:1606.01540 (2016).