Graph-Based Reinforcement Learning Method with Control Barrier Functions for Safe Marine Traffic Control

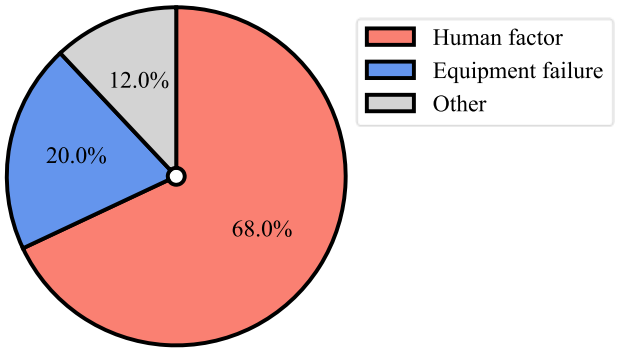
Department of Marine System Eng.,
Osaka Metropolitan Univ.

Hitoshi Yoshioka

Introduction

➢Background

- Most ship collision accidents are attributed to human error. Preventing such collisions is inherently challenging.
- The number of seafarer has been decreasing year by year.



Causes of collisions [1]

Autonomous navigation technology is expected to solve these issues.

Collision avoidance is one of the key technologies for ship navigation. The automation of collision avoidance is essential for realizing autonomous ships.

[1] Grad, Collisions – Why do they occur?

Introduction

- Recently, deep reinforcement learning (DRL)-based algorithms for ship collision avoidance has been studied.

The capability of them have been validated through numerical simulations[2], a scaled model ship experiments[3], and full-scale ship experiments[4].

The results show that their performance exceeds that of human experts.

✗ These studies focus on optimizing collision avoidance for individual ships. The performances of the such controls degrades when other ships take unanticipated actions.

In congested situations, optimizing all ships through centralized control is effective for ensuring safe marine traffic control.

[2] Sawada, R., et al. 2020. *Journal of Marine Science and Technology.*

[3] Shen, H., et al. 2019. *Applied Ocean Research.* [4] Yoshioka, H. and Hashimoto, H. 2023. *Proceedings of AMEC 2023.*

Challenges/Objective

➢Challenges for developing a marine traffic control AI

- Designing an appropriate data structure for marine traffic control
- Preventing local collision risks while optimizing marine traffic

➢Objective

To develop a centralized marine traffic control AI
capable of safely controlling multiple ships in congested situations

- Marine traffic is represented using a graph structure
  and the AI is trained via graph-based deep reinforcement learning.
- A safety constraint based on a control barrier function (CBF) is
  incorporated into the DRL framework to ensure safety.

Deep Reinforcement Learning

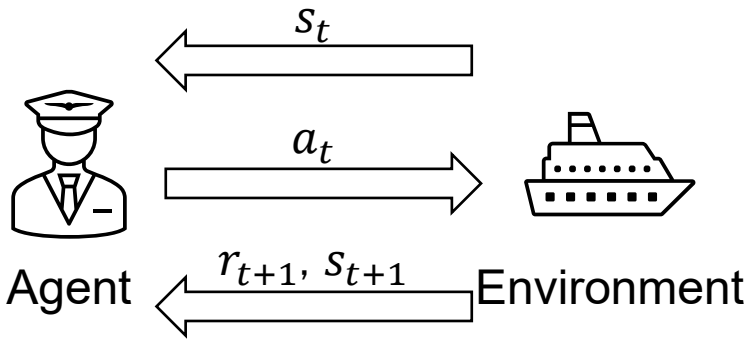
➢Flow of reinforcement learning

① Agent observes the current state of environment.

② Agent takes an action according to its policy.

③ Environment transits to a next state and returns a reward to the agent.

④ Policy is updated to maximize the expected reward using the obtained reward.



➢Objective of reinforcement learning

To get an optimal policy that maximize a discounted cumulative reward

We use deep deterministic policy gradients[5].

[5] Lillicrap, et al. (2016). *International Conference on Learning Representations.*

Control Barrier Function

- A control barrier function (CBF) imposes safety constraints on the control input to ensure the system remains within a safe set.

State equation	$\dot{x} = f(x) + g(x)u$
A function indicate whether x is safe ($h(x) \geq 0$) or dangerous	$h(x)$

Safety constraint : $\underbrace{L_f h(x) + L_g h(x) u}_{\dot{h}(x)} + \alpha\big(h(x)\big) \geq 0$

α: any monotonic increasing function

If the control input u satisfies the constraint, the state remain the safe set.

[]

[6] Aaron, et al. 2019. *18th European Control Conference.*

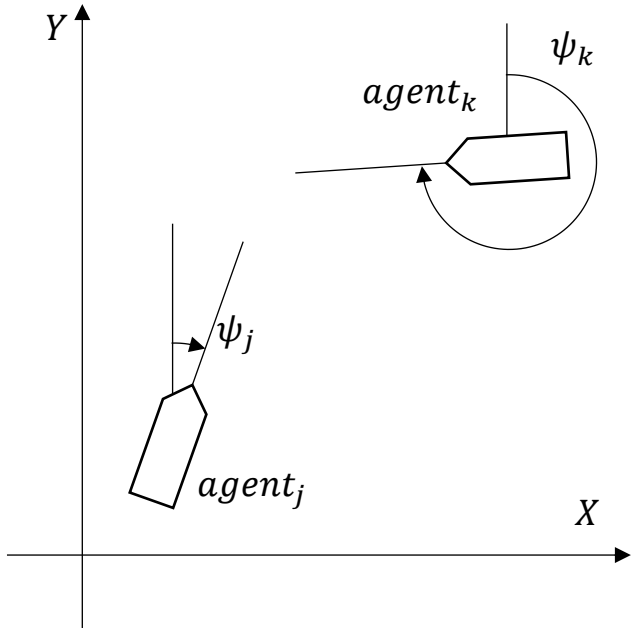
Problem setting

- The number of ships, their speeds, and their initial positions are randomly determined.

Number of agent in episode	2 ship ~ 20 ships
Ship's speed (constant)	8 knot ~ 12 knot

- Ship collision avoidance manoeuver is calculated using Nomoto's KT model[7].

State equation of $agent_j$	$x_j = \begin{bmatrix} X_j \\ Y_j \\ \psi_j \\ r_j \end{bmatrix}, \quad \dot{x}_j = \begin{bmatrix} U_j \cos(\psi_j) \\ U_j \sin(\psi_j) \\ r_j \\ -r_j/T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ K/T \end{bmatrix} u_j$
Control input of $agent_j$ (u_j)	Rudder angle: -10 ° ~ 10 °



[7] Nomoto. (1920). *1st Symposium on Ship Manoeuvrability, 275–304*

Graph Structure for Representing Marine Traffic

➢Characteristics of the marine traffic

- The number of ships is changing by time.
- Ships that are far away
  do not affect collision-avoidance decisions.
  Collision-avoidance actions can be
  determined by local interactions
  with neighboring ships.

➢Graph structures are suitable for
representing marine traffic.

Time-varying graph $\mathbb{G}_t = (\mathbb{V}, \mathbb{E}_t)$

Node $\mathbb{V}$	Ships (agent) $0, .., N-1$
Edge $\mathbb{E}_t$	Encounter relationship between ships

Detecting Encounter Relationship

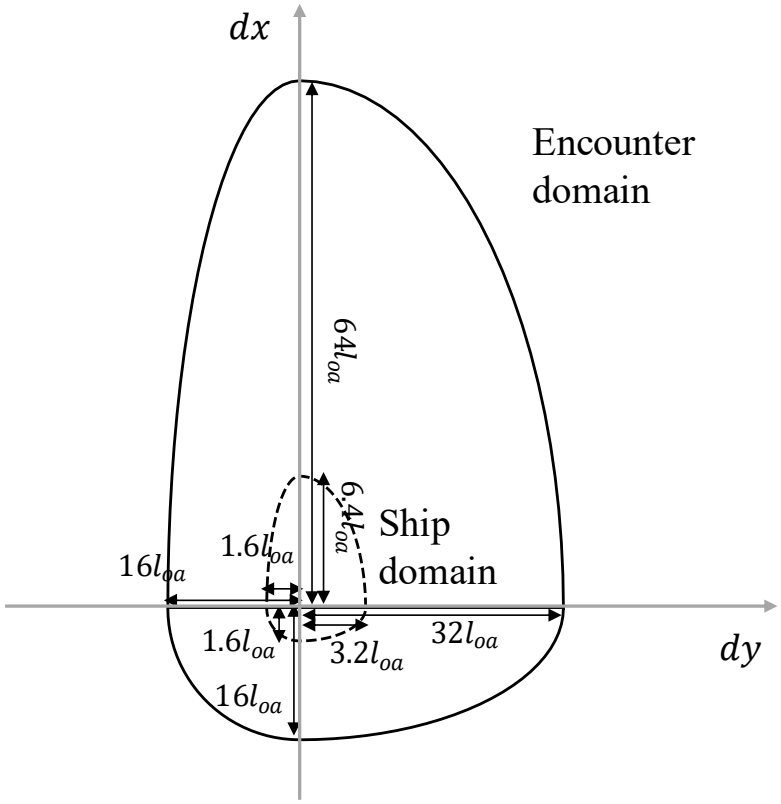
➢Encounter relationships are detected using a ship domain.

- The ship domain is a safety region around a ship that should not be invaded by other ships during navigation.

If another ship enters the ship domain, the situation is considered dangerous.

- An area defined as ten times the ship domain is referred to as the Encounter Domain.

Ships within the Encounter Domain are considered to be in encounter relationships, and edges are established between them.



Node features of Observation Graph

➢Observations are also represented as a graph.

- Information related to path following is embedded in the nodes.

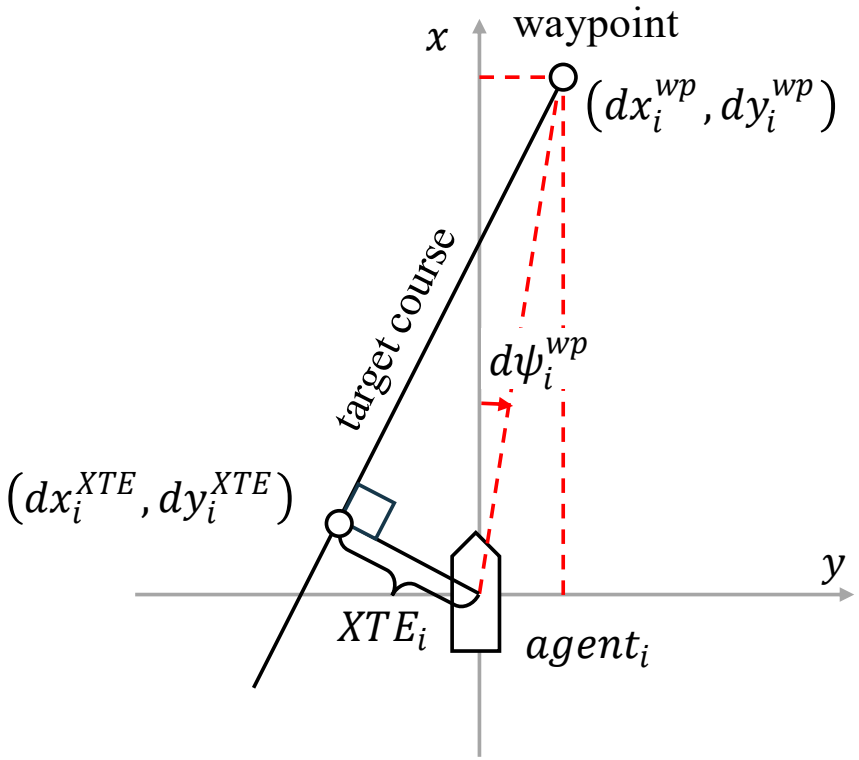
Relative position of the next waypoint: $\left(dx_{t,i}^{wp}, dy_{t,i}^{wp}\right)$

Relative bearing angle of the next waypoint: $d\psi_{t,i}^{wp}$

Relative position of the closest point on target course: $\left(dx_{t,i}^{XTE}, dy_{t,i}^{XTE}\right)$

Distance between the own ship and the closest point: $XTE_{t,i}$

$$n_{t,i}^{s} = \begin{bmatrix} dx_{t-4,i}^{wp}, dy_{t-4,i}^{wp,i}, d\psi_{t-4,i}^{wp,i}, dx_{t-4,i}^{XTE,i}, dy_{t-4,i}^{XTE,i}, XTE_{t-4,i} \\ \cdots \\ dx_{t,i}^{wp}, dy_{t,i}^{wp}, d\psi_{t,i}^{wp}, dx_{t,i}^{XTE}, dy_{t,i}^{XTE}, XTE_{t,i} \end{bmatrix}$$

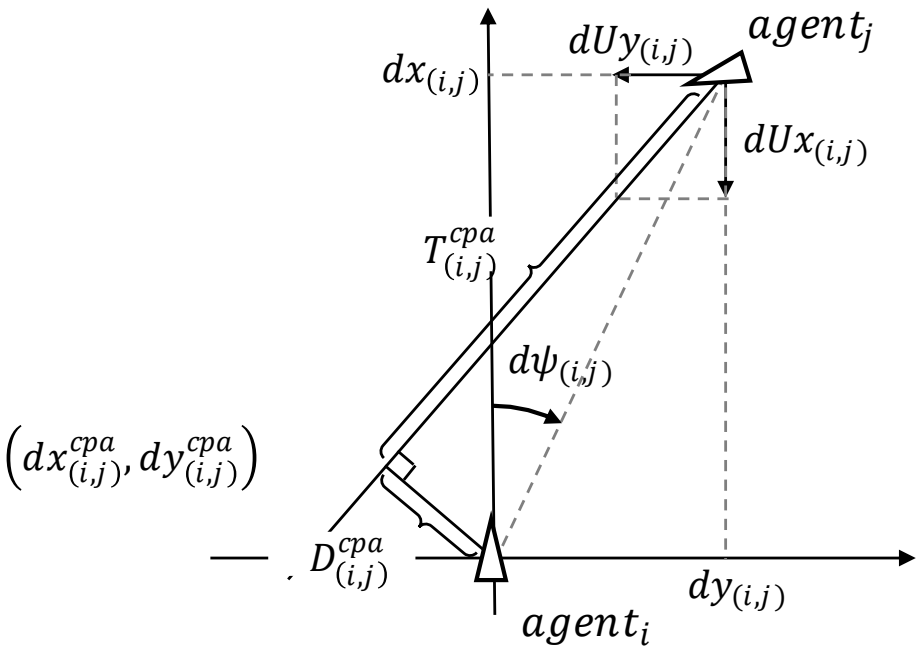


Edge features of Observation Graph

- Information related to encounter relationships is embedded in the edges.

Relative position: $\left(dx_{t,(i,j)}, dy_{t,(i,j)}\right)$
relative bearing angel: $d\psi_{t,(i,j)}$
relative speed: $\left(dUx_{t,(i,j)}, dUy_{t,(i,j)}\right)$
distance normalized by the ship domain: $D^{SD}_{t,(i,j)}$
information of closest point for approach (CPA):
$\left(D^{cpa}_{t,(i,j)}, T^{cpa}_{t,(i,j)}, dx^{cpa}_{t,(i,j)}, dy^{cpa}_{t,(i,j)}\right)$
distance normalized by the *encounter domain*: $D^{ED}_{t,(i,j)}$

$$e^{s,}_{t,(i,j)} = \begin{bmatrix} dx_{t-4,(i,j)}, dy_{t-4,(i,j)}, d\psi_{t-4,(i,j)}, dUx_{t-4,(i,j)}, dUy_{t-4,(i,j)}, \\ D^{SD}_{t-4,(i,j)}, D^{cpa}_{t-4,(i,j)}, T^{cpa}_{t-4,(i,j)}, dx^{cpa}_{t-4,(i,j)}, dy^{cpa}_{t-4,(i,j)}, D^{ED}_{t-4,(i,j)}, \\ \dots \\ dx_{t,(i,j)}, dy_{t,(i,j)}, d\psi_{t,(i,j)}, dUx_{t,(i,j)}, dUy_{t,(i,j)}, \\ D^{SD}_{t,(i,j)}, D^{cpa}_{t,(i,j)}, T^{cpa}_{t,(i,j)}, dx^{cpa}_{t,(i,j)}, dy^{cpa}_{t,(i,j)}, D^{ED}_{t,(i,j)} \end{bmatrix}$$

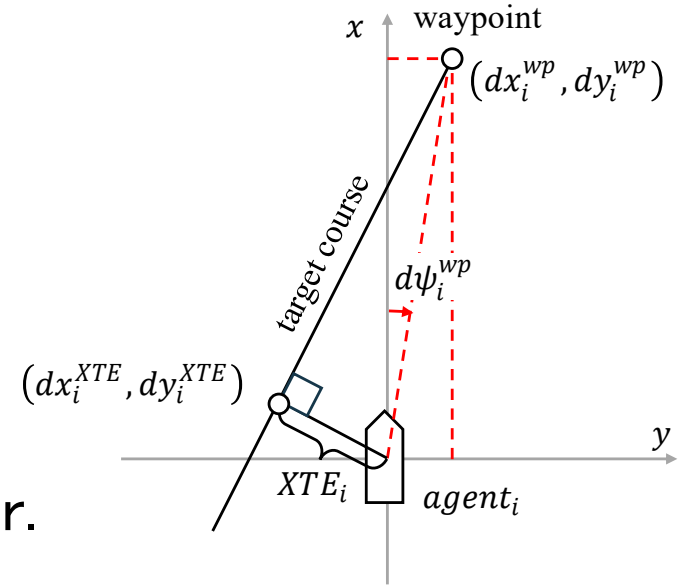


Reward Graph

➢Rewards related to path following are embedded in the nodes.

- Reward based on the relative bearing angle: $0.0-0,3$

A larger reward is obtained as
the relative bearing angle becomes smaller.



- Reward based on the distance
  from the target course: $0.0-0.3$

A larger reward is obtained as the distance becomes smaller.

➢Rewards related to collision avoidance are embedded in edges.

- Reward based on collision risk: -1.0 – -0.5

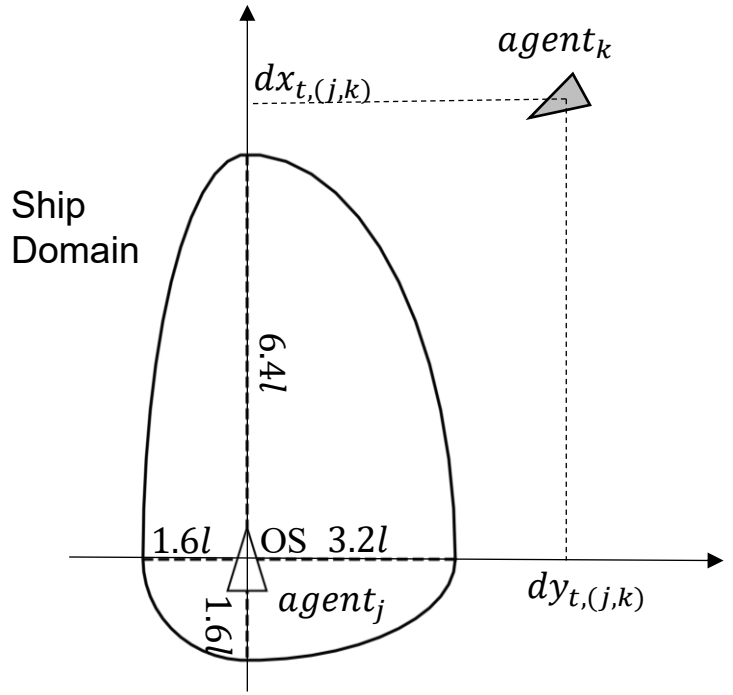
If the agent is in a dangerous state, a large penalty is imposed.

Graph Indicating Dangerous State

- Dangerous states are evaluated based on the ship domain.

The function is designed to output negative values when another agent enters the ship domain.

$$e^{h}_{t,(i,j)} = \sqrt{\left(\frac{dx_{t,(i,j)}}{Ax^{x}_{t,(i,j)}}\right)^{2} + \left(\frac{dy_{t,(i,j)}}{Ax^{y}_{t,(i,j)}}\right)^{2}} - 1$$



A graph indicating dangerous state has the evaluated value as edge features.

Learning of Graph Q-function

- Nodes (agents) do not disappear during an episode.
  Therefore, the Q-value of a node is the cumulative node reward during the episode.
- Edge (encounter relationship) can disappear during an episode.
  The edge Q-value is the cumulative edge reward until the edge disappears.

Node Q-value $n_t^q \coloneqq n_{t+1}^r + \gamma n_{t+2}^r + \gamma^2 n_{t+3}^r + \cdots$

Edge Q-value $e_t^q \coloneqq e_{t+1}^r + \gamma e_{t+2}^r + \gamma^2 e_{t+3}^r + \cdots \gamma^{T^{disap.}-t-1} e_{T^{disap.}}^r$ $T^{disap.}$: edge disappear

- The loss function is the TD error.

$\mathcal{L}^Q = \frac{1}{N^B} \sum_B \left\| n_t^q - \hat{n}_t^q \right\|^2 + \left\| e_t^q - \hat{e}_t^q \right\|^2$ $\hat{n}_t^q, \hat{e}_t^q$: predicted Q-values

Learning of Constraint

- For complex or unknown dynamics, it is difficult to obtain the safety constraint.

Since DRL can be applied to the unknown dynamic, it is needed that the safety constraint can be trained in the unknown dynamic to not limit the capability of DRL.

➢ We trained the CBF constraint using two-steps approximation.

•The CBF constraint is approximated by the sampled data.

$$L_f h(x) + L_g h(x) u + \alpha\big(h(x)\big) \approx \frac{(h_{t+1} - h_t)}{\Delta t} + \alpha(h_t)$$

•The value estimated using the sampled data is trained by neural networks.

$$c_t = \frac{(h_{t+1} - h_t)}{\Delta t} + \alpha(h_t)$$

Learning Constraint

- In the case that the own ship is imposed to danger even if the AI take the optimal action, the constraint should activate earlier. The training target is a minimum value of the constraint if the AI take the optimal action.

To emphasize constraints in the near future, a small positive value, Δ, is added to the future constraint.

$$\tilde{c}_t = \min(c_t, c_{t+1} + \Delta, c_{t+2} + 2\Delta \dots) = \min(c_t, \tilde{c}_{t+1})$$

The minimum values on future is estimated using the training network.

$$\tilde{c}_t = \min\left(\frac{e^h_{t+1} - e^h_t}{\Delta t} + \alpha\left(e^h_t\right), C(\mathbb{G}^s_{t+1}|\theta^C) + \Delta\right)$$

Loss function is a root mean squared error between the target and the predicted.

$\mathcal{L}^C = \sum_B \left\|\tilde{c}_t - \hat{e}^C_t\right\|^2$ $\hat{e}^C_t$: predicted value of the constraint

Learning of Policy

- Constraint and Q-values for the observation $\mathbb{G}_t^s$ are estimated using the training networks.

Constraint

$$e_t^{c,\pi} = C(\mathbb{G}_t^s)$$
$$l_t^{\pi,const} = \sum_{e \in e_t^{c,\pi}} \max(e, 0)$$

Q-values

$$\{n_t^{q,\pi}, e_t^{q,\pi}\} = Q(\mathbb{G}_t^s)$$
$$l_t^{\pi,q} = \sum_{n \in n_t^{q,\pi}} n + \sum_{e \in e_t^{q,\pi}} e$$

- Loss for the observation $\mathbb{G}_t^s$

$$l_t^{\pi} = \begin{cases} -l_t^{\pi,const}, & l_t^{\pi,const} < 0 \\ -l_t^{\pi,q}, & l_t^{\pi,const} \geq 0 \end{cases}$$

If constraint is violated, the policy is updated to satisfy it.

If constraint is satisfied, the policy is updated to maximizing Q-values.

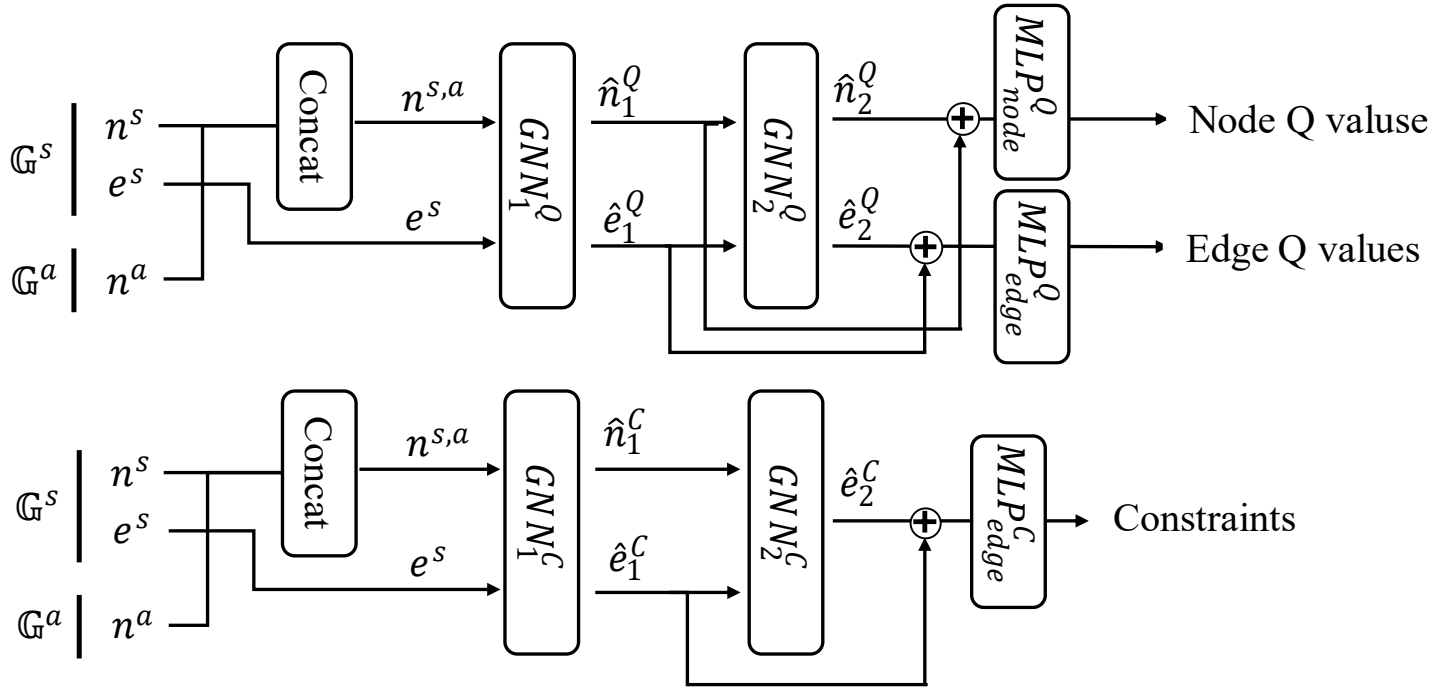
- Loss function is a mean of loss values for the batch sampled data.

Architecture of the Q-Network and Constraint Network

① Observation graph and action graph used as inputs to the network.

② Node and edge features are updated through two GNN blocks.

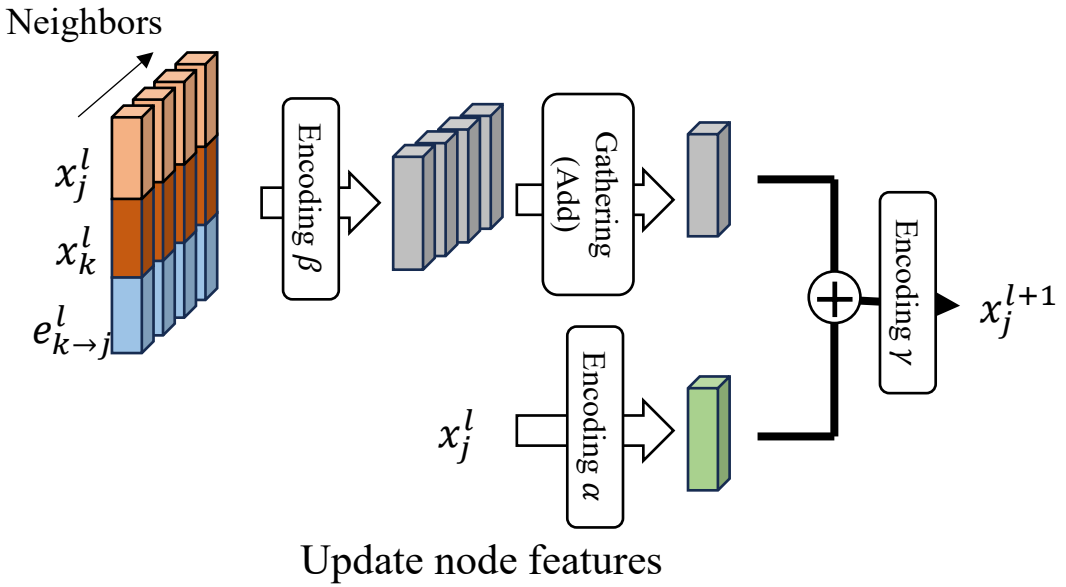
③ Q-values or constraint values are estimated from updated features.



Processing in the Graph Neural Network

➢Update node features

1. The neighboring node features are concatenated with the node's own feature and the edge feature between them.
2. They are encoded by a neural network, $\boldsymbol{\beta}$, and aggregated by summation.
3. The node's own features are updated by a neural network, $\boldsymbol{\alpha}$.
4. Encoded features of the neighboring nodes and the own node are added and encoded by the neural network $\boldsymbol{\gamma}$.

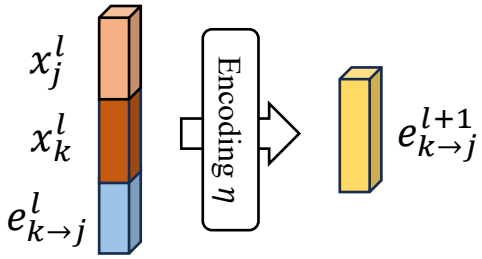


$$x_j^{l+1} = \gamma\left(\alpha\left(x_j^l\right) + \sum \beta\left(x_j^l, x_k^l, e_{k\to j}^l\right)\right)$$

Processing in the Graph Neural Network

➢Updating edge features

1. The edge features are concatenated with the features of the connected nodes.
2. The concatenated features are encoded by a neural network, η.

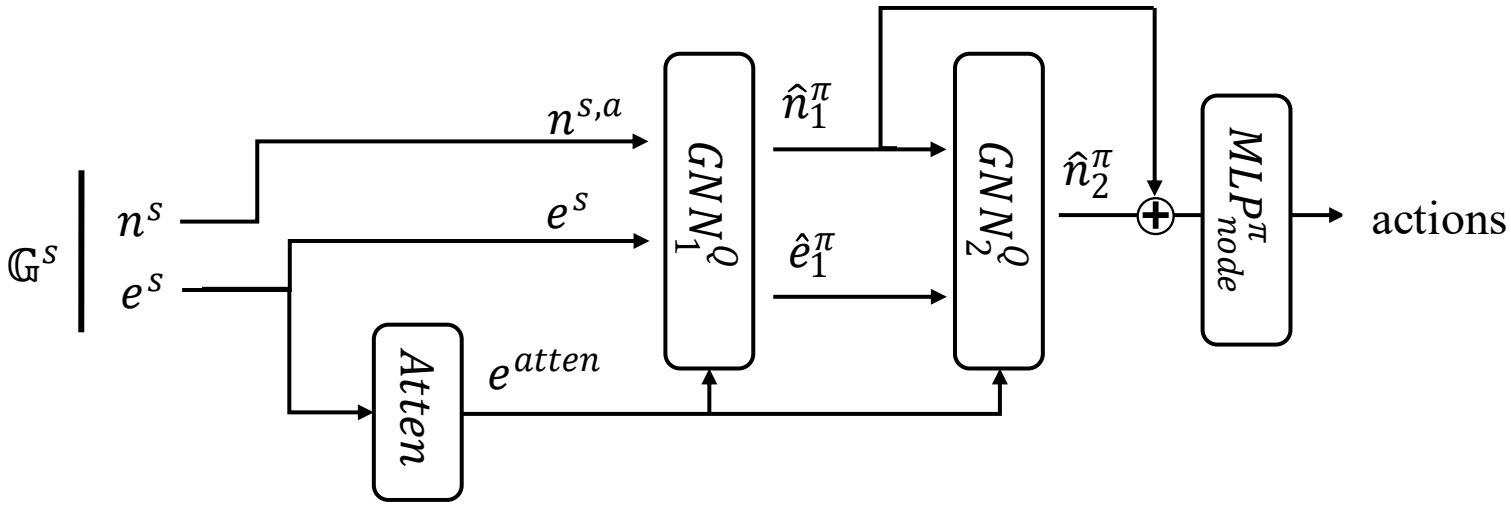


Update edge features

$$e_{k\to j}^{l+1} = \eta(x_j^l, x_k^l, e_{k\to j}^l)$$

Architecture of the Policy Network

- The policy network consists of two GNN blocks and a multilayer perceptron.
- An attention mechanism is incorporated into the aggregating neighboring nodes.



Attention mechanism

➢Prior-defined attention function

The attention mechanism learns the importance of neighboring nodes using a neural network.

In collision avoidance decision-making, the importance of another ship increases as the relative distance decreases.

To reflect domain knowledge in collision avoidance, a distance-dependent monotonic function is predefined, and its shape is learned from data.

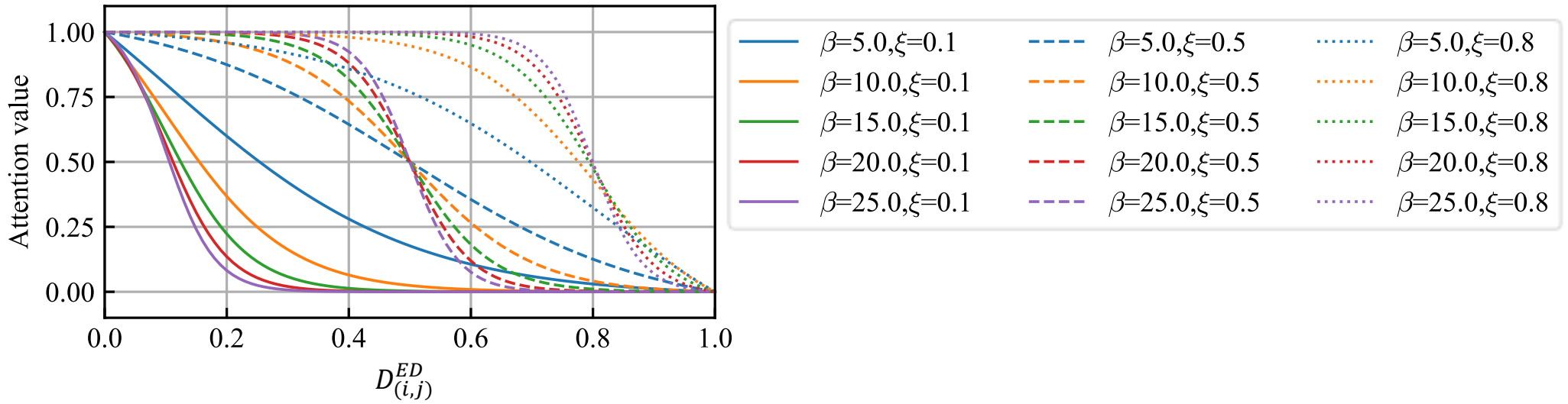
$$\alpha_{(i,j)} = \frac{\text{sigmoid}\left(\beta\left(D_{(i,j)}^{ED} - \xi\right)\right) - \text{sigmoid}(\beta(1-\xi))}{\text{sigmoid}(-\beta\xi) - \text{sigmoid}\left(\beta(1-\xi)\right)}$$

$D_{(i,j)}^{ED}$: Distance

Predefined Attention Function

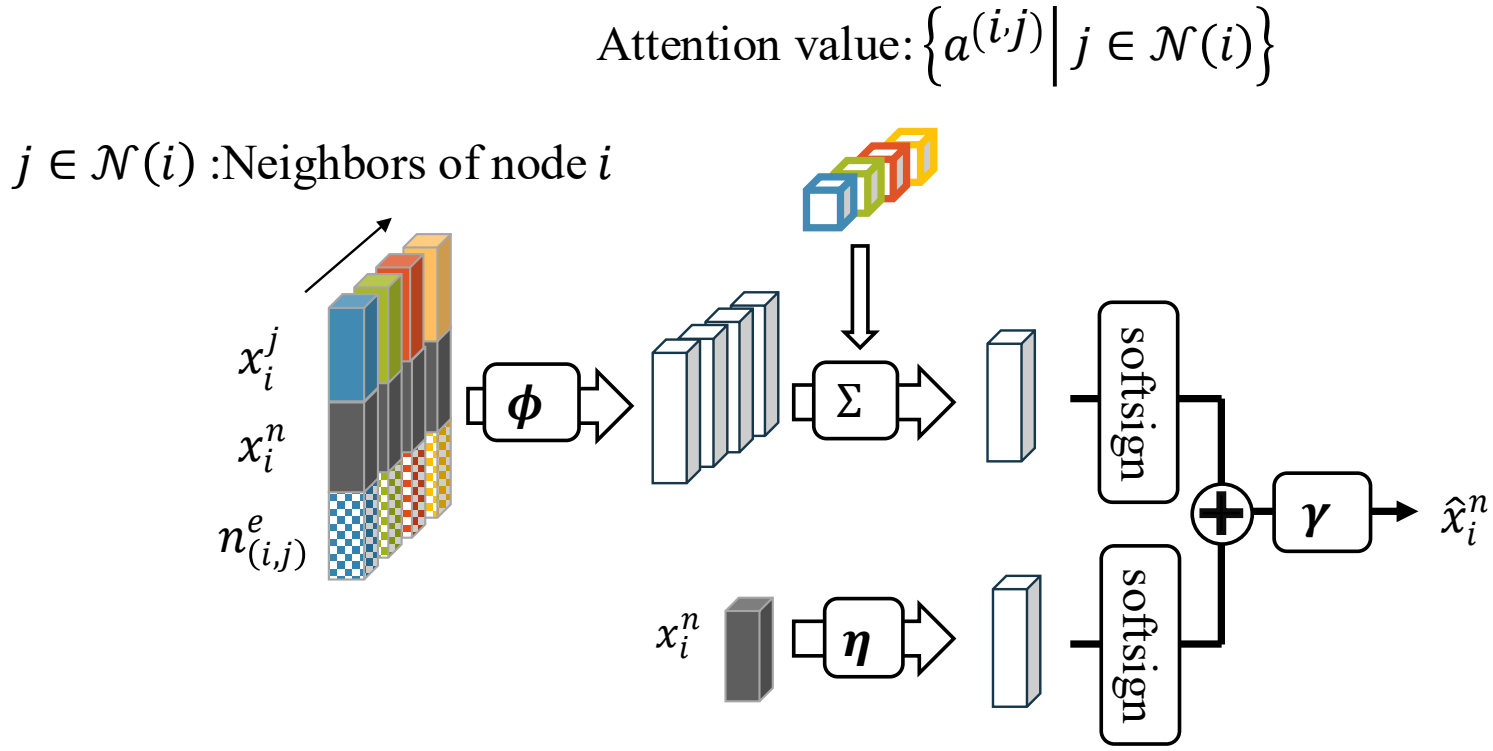
$$\alpha_{(i,j)} = \frac{\text{sigmoid}\left(\beta\left(D^{ED}_{(i,j)} - \xi\right)\right) - \text{sigmoid}(\beta(1-\xi))}{\text{sigmoid}(-\beta\xi) - \text{sigmoid}\left(\beta(1-\xi)\right)}$$

β	The point at which the attention value begins to decrease farther from the agent
η	The rate of change of the attention value



GNN Block with Attention Mechanism

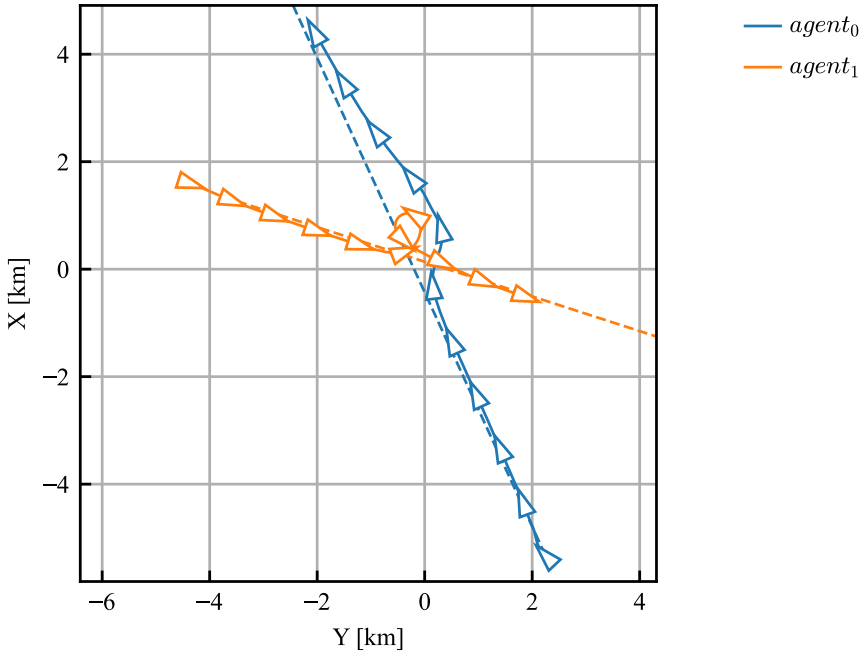
- The encoded features of the neighboring nodes are aggregated using a weighted average based on the attention values.

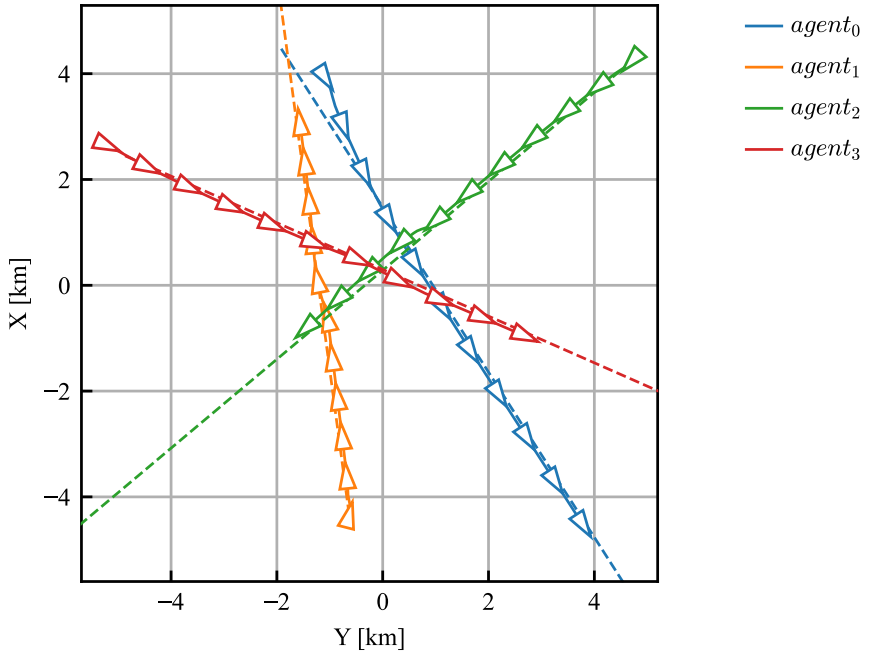


Update node features introduced attention mechanism

Absolute Trajectories in the Simple Scenario

- Absolute trajectories

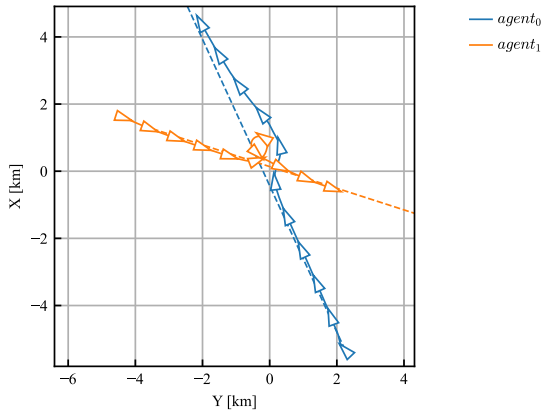


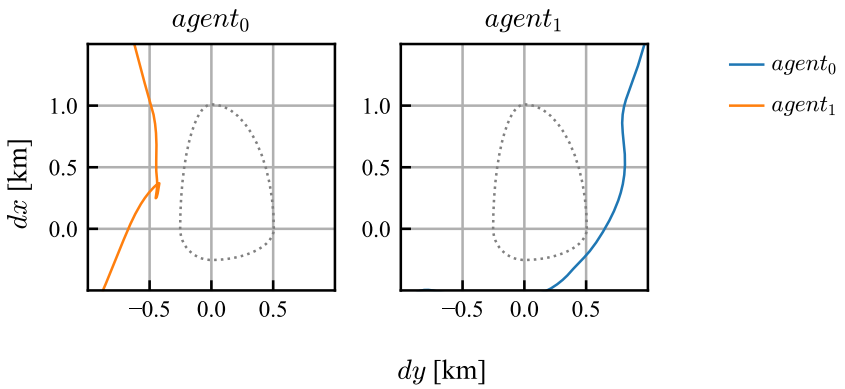


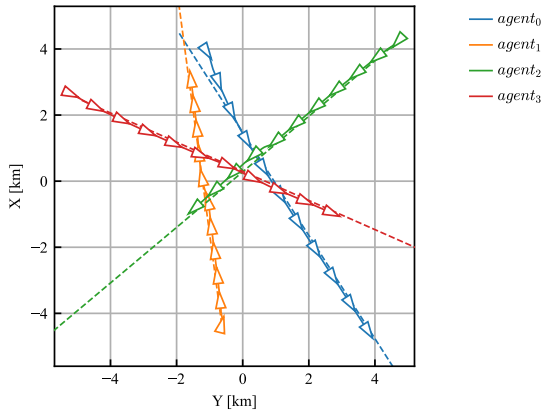
The AI-controlled ships took collision avoidance actions.

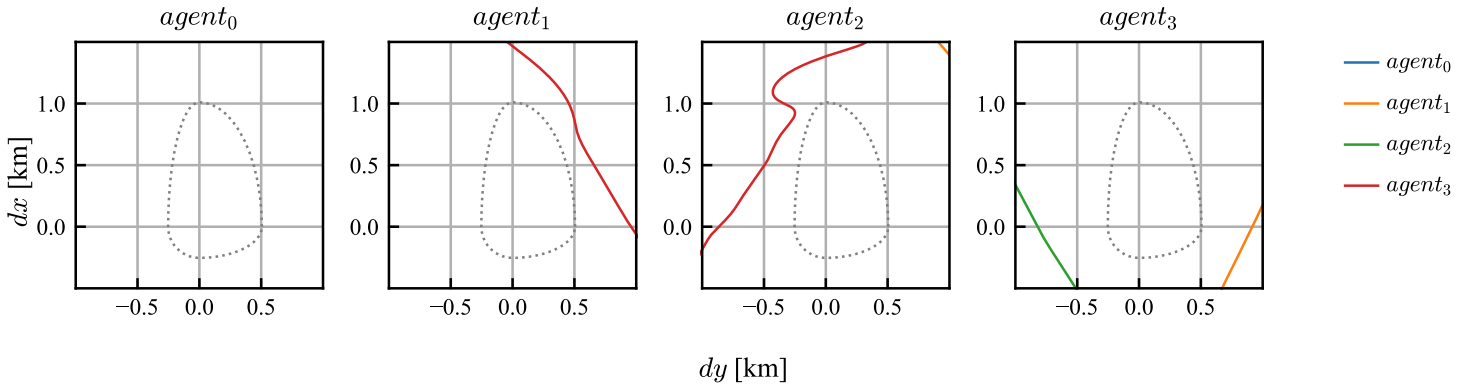
Each ship followed the target path when no danger is present.

Relative Trajectories in the Simple Scenario





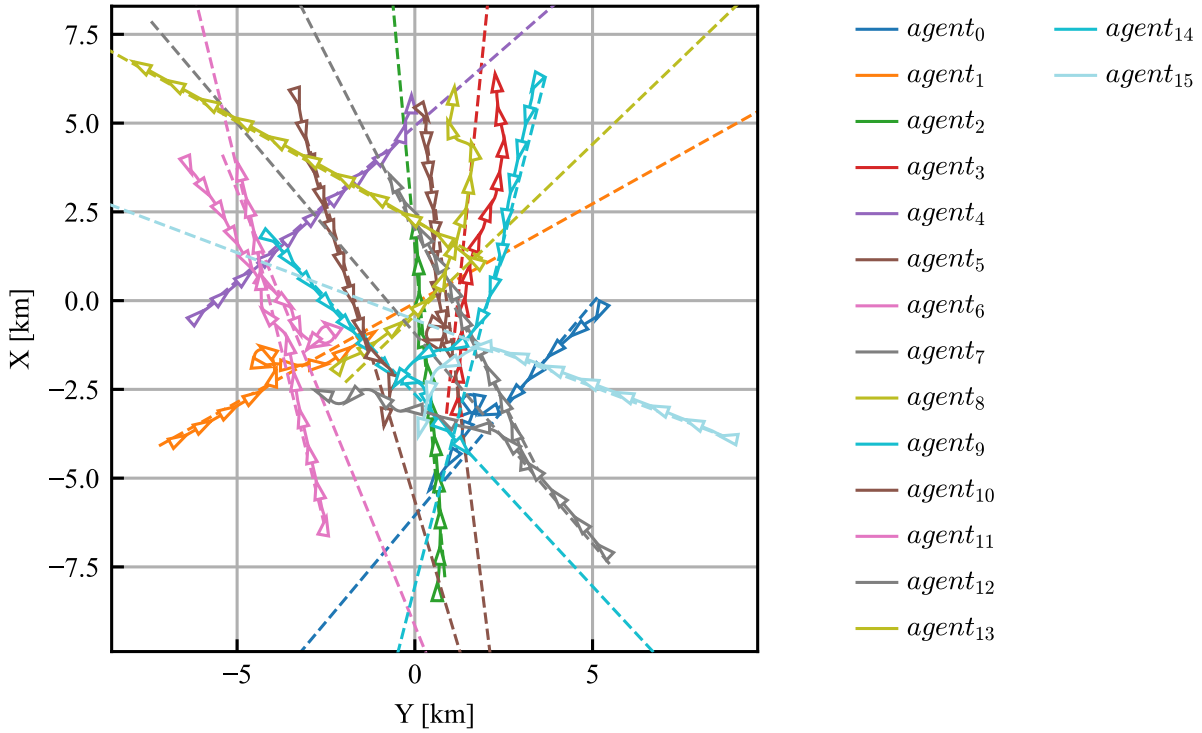




No ship-domain violations were occurred.

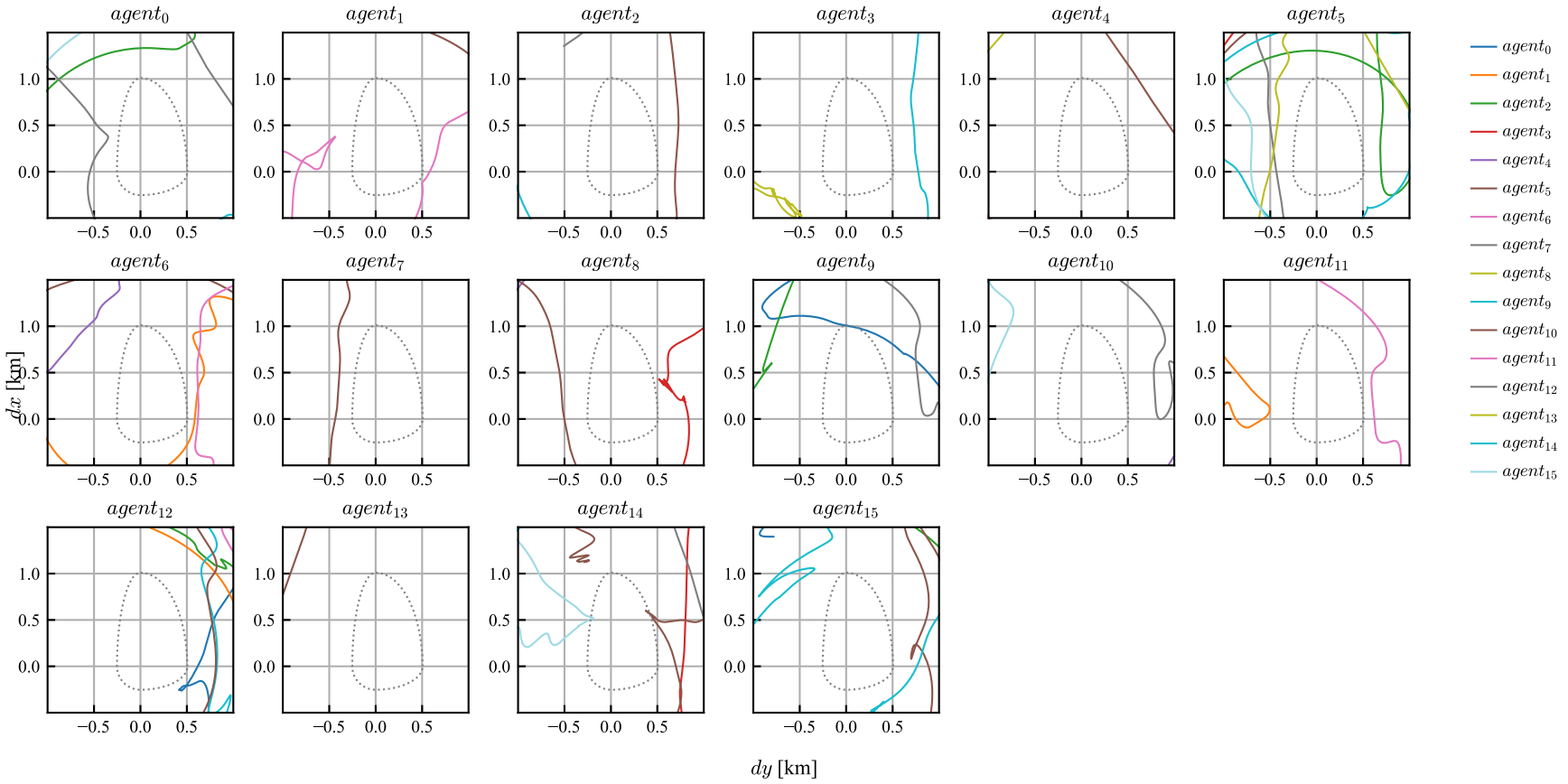
The trained controlled ships without the risk of collision.

Absolute Trajectories in the Congested Scenario



In congested scenario, the AI-controlled ships avoided collisions while following their target paths.

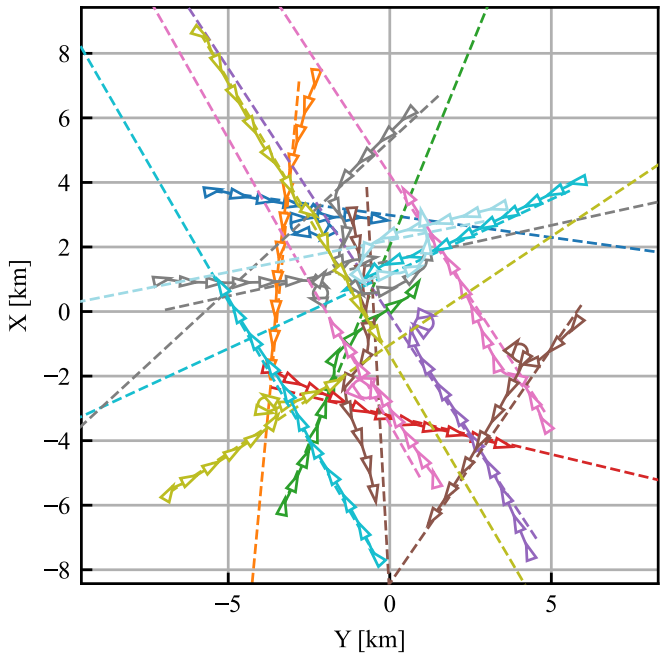
Relative Trajectories in the Congested Scenario

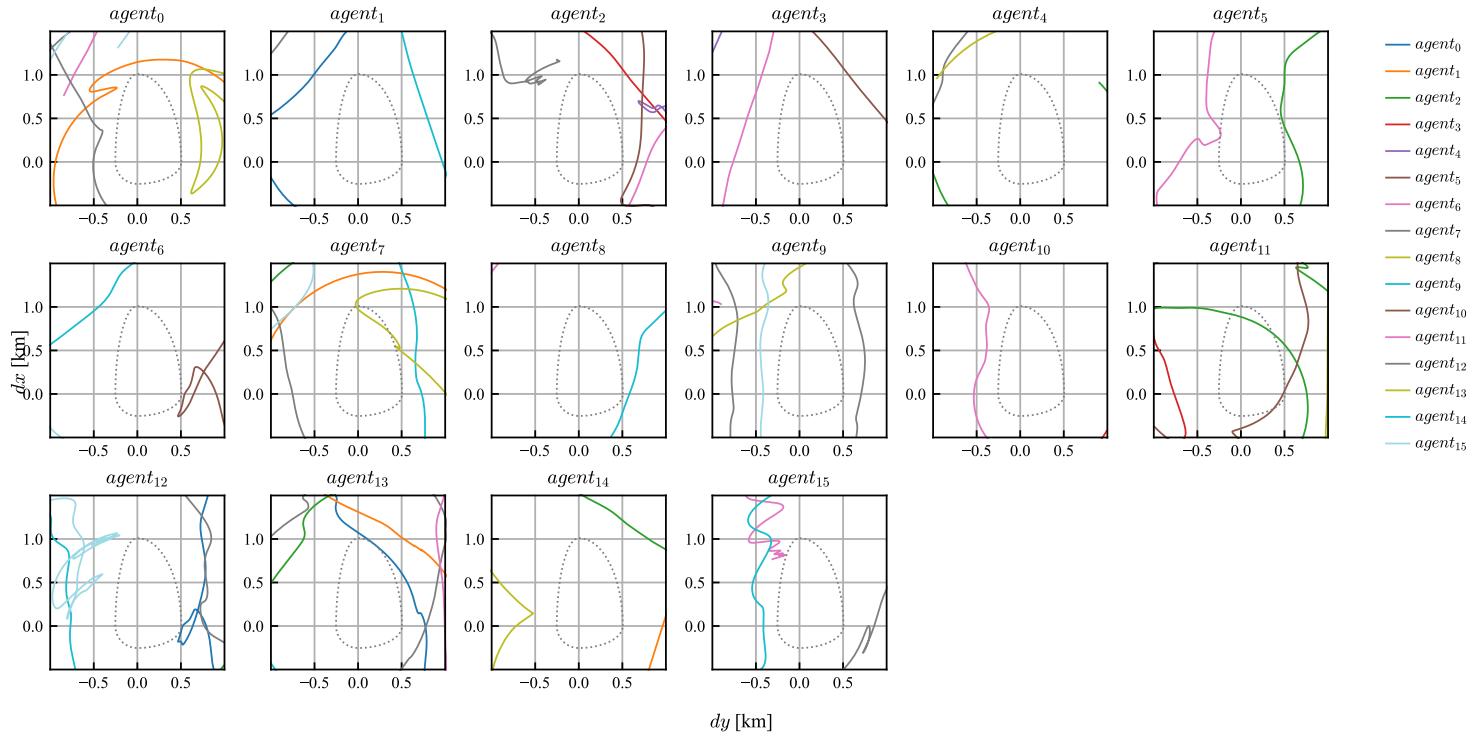


The ship domains were not violated by other ships.

Even in congested situations, all ships remained in the safe state.

Results in the Highly Congested Scenario





Although minor ship-domain violations were observed,
they occurred only near the boundary of the safety domain.

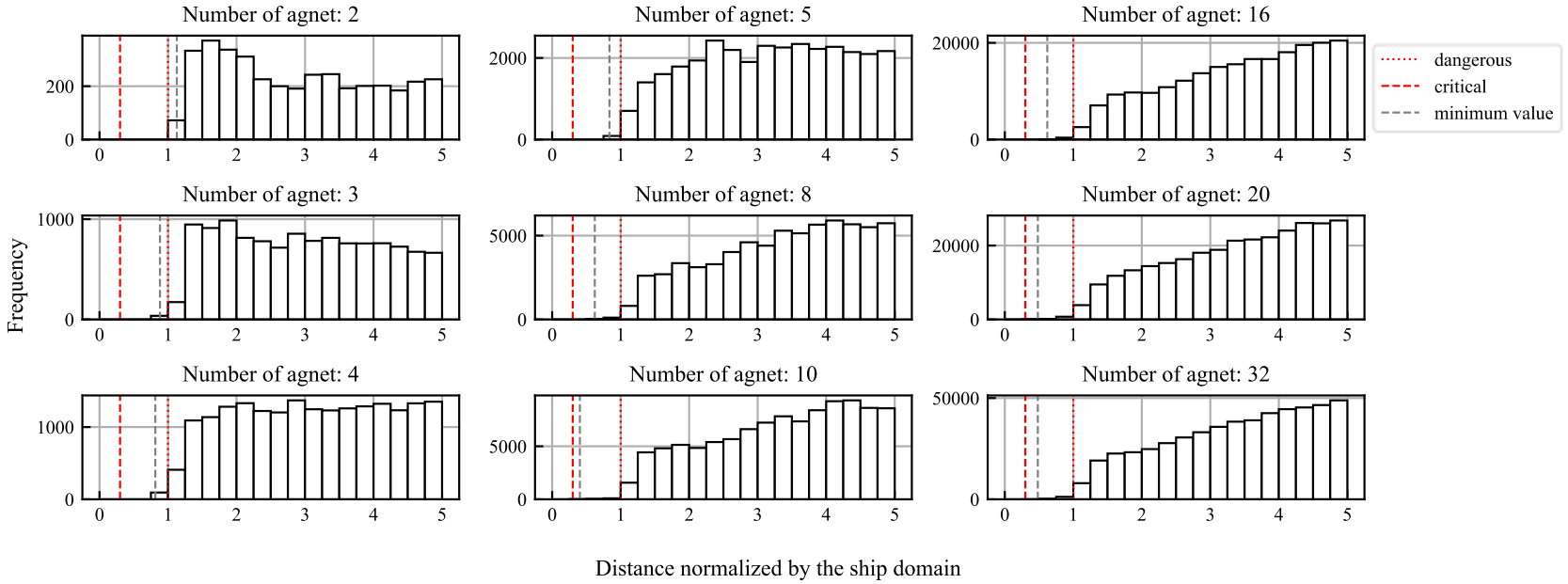
These results can be interpreted as
the AI-controlled ships minimizing danger.

Validation of Scalability

- Comparison of the frequency of dangerous states by the number of agents

$D_{(i,j)}^{ED} \leq 1.0$ | The ship domain is invaded (danger)

$D_{(i,j)}^{ED} \leq 0.3$ | Relative distance is less than half of the length over all (critical)



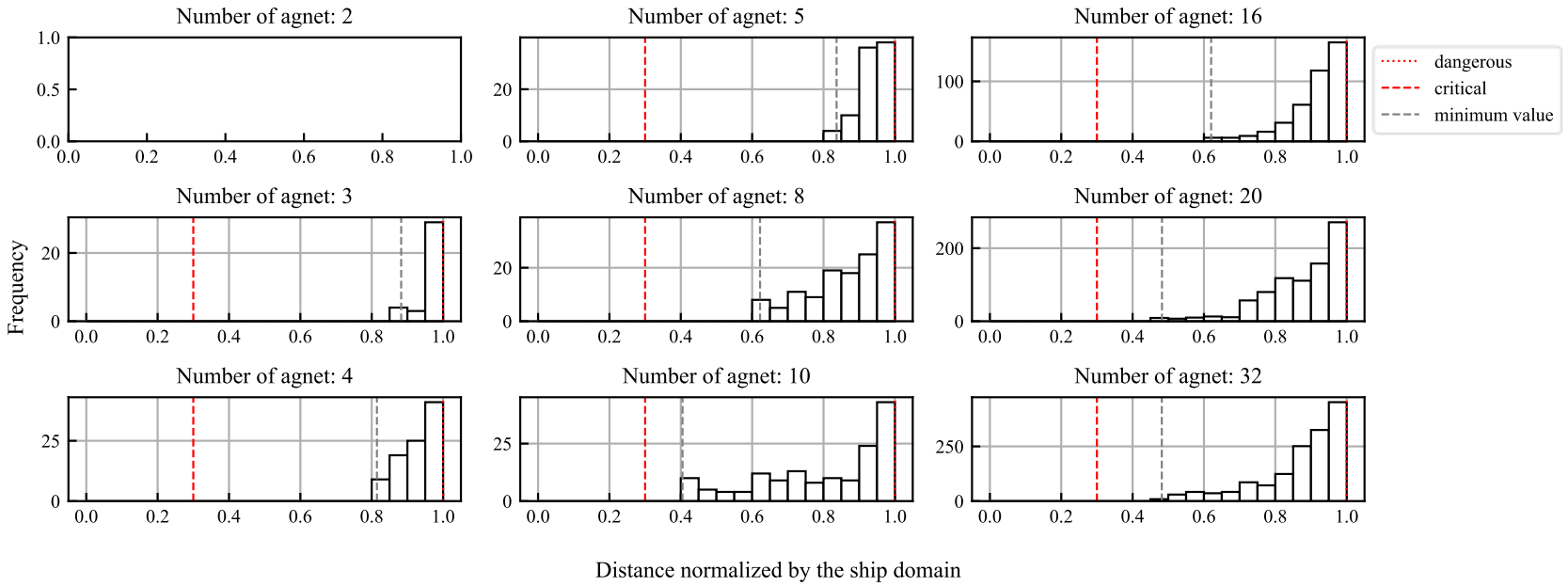
Few dangerous states occurred even as the number of agents increased.

Validation of Scalability

- Comparison of the frequency of dangerous states in the dangerous range.

$D_{(i,j)}^{ED} \leq 1.0$ | The ship domain is invaded (danger)

$D_{(i,j)}^{ED} \leq 0.3$ | Relative distance is less than half of the length over all (critical)



If the number of agents increased, no critical dangerous is occurred.

Comparison with RL and D-GCBF

- To investigate the effectiveness of the deep reinforcement learning with safety constraint, we trained three AIs: one using RL with D-GCBF, one using RL alone, and one using D-GCBF alone.
- In the case of RL alone, both of collision avoidance and path following were trained using RL.
- In the case of D-GCBF alone, collision avoidance was trained using D-GCBF and path following is trained using RL.

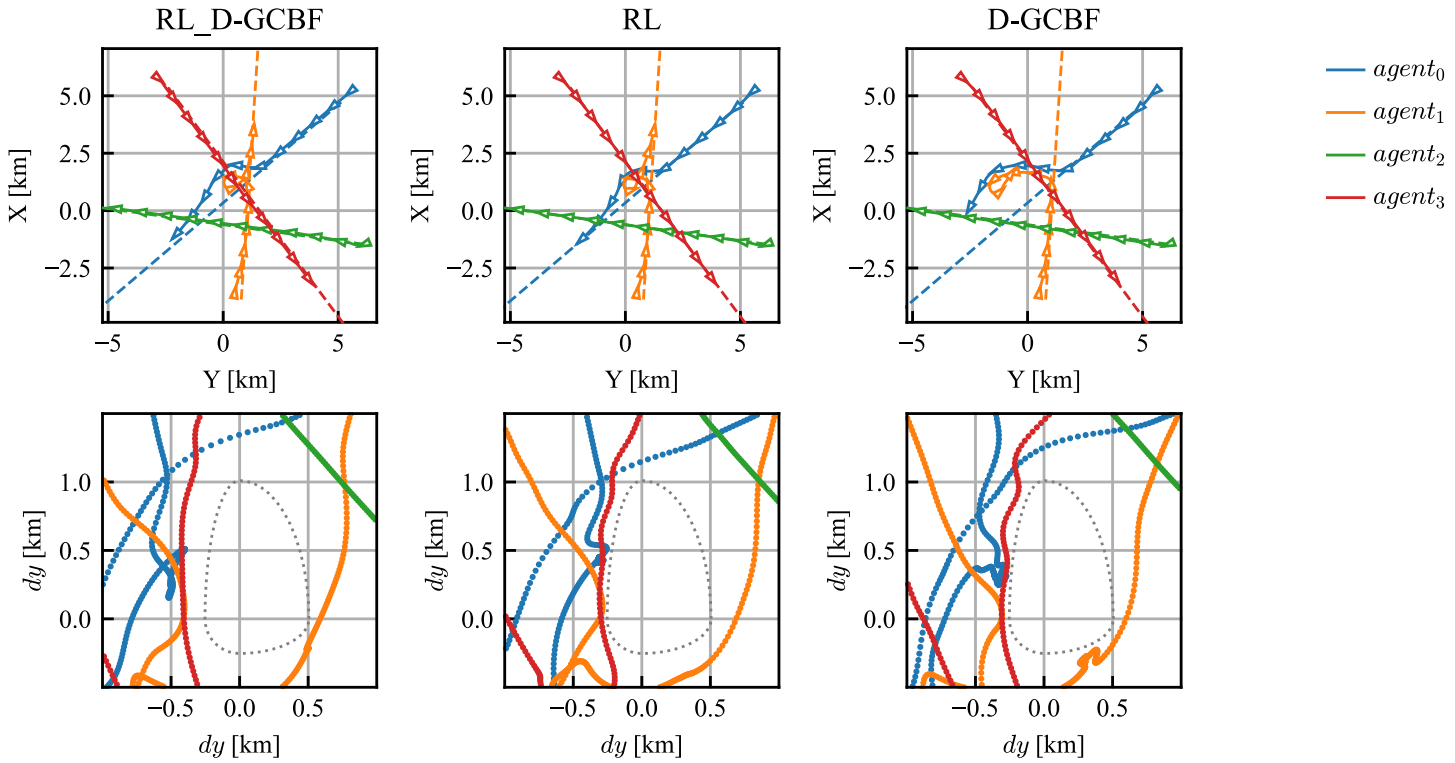
RL alone

$$\mathcal{L}^{\pi,RL} = \frac{1}{N^B} \sum_{\{\mathbb{G}_t^s\} \in B} l_t^{\pi,const}(\mathbb{G}_t^s)$$

D-GCBF alone

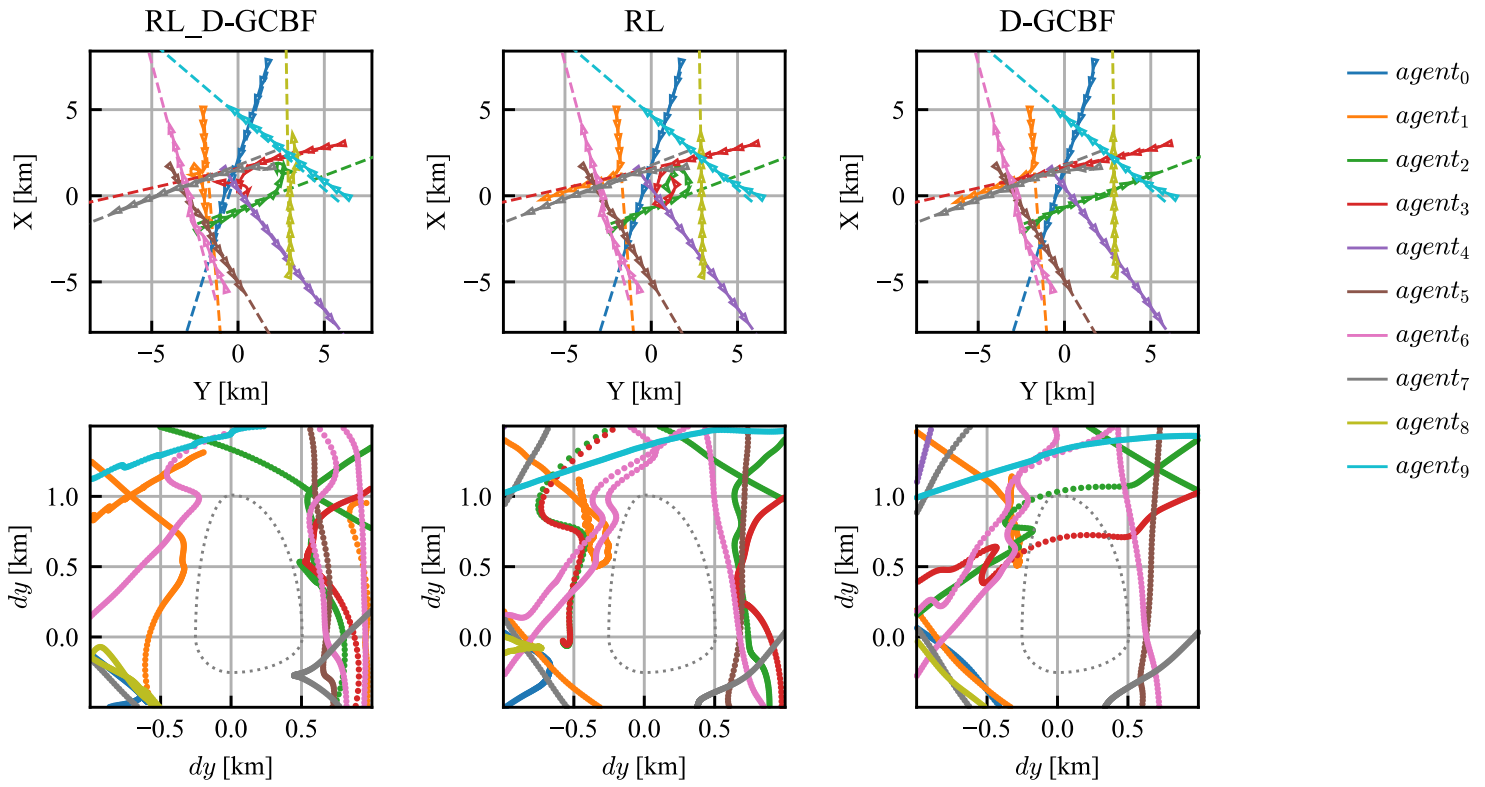
$$\mathcal{L}^{\pi,D-GCBF} = \frac{1}{N^B} \sum_{\{\mathbb{G}_t^s\} \in B} \begin{cases} -l_t^{\pi,const}(\mathbb{G}_t^s), & l_t^{\pi,const}(\mathbb{G}_t^s) < 0 \\ -\sum_{n \in n_t^{q,\pi}(\mathbb{G}_t^s)} n, & l_t^{\pi,const}(\mathbb{G}_t^s) \geq 0 \end{cases}$$

Comparison Result in the Simple Scenario



All AIs controlled the ships without
any ship-domain violations in the simple scenario.

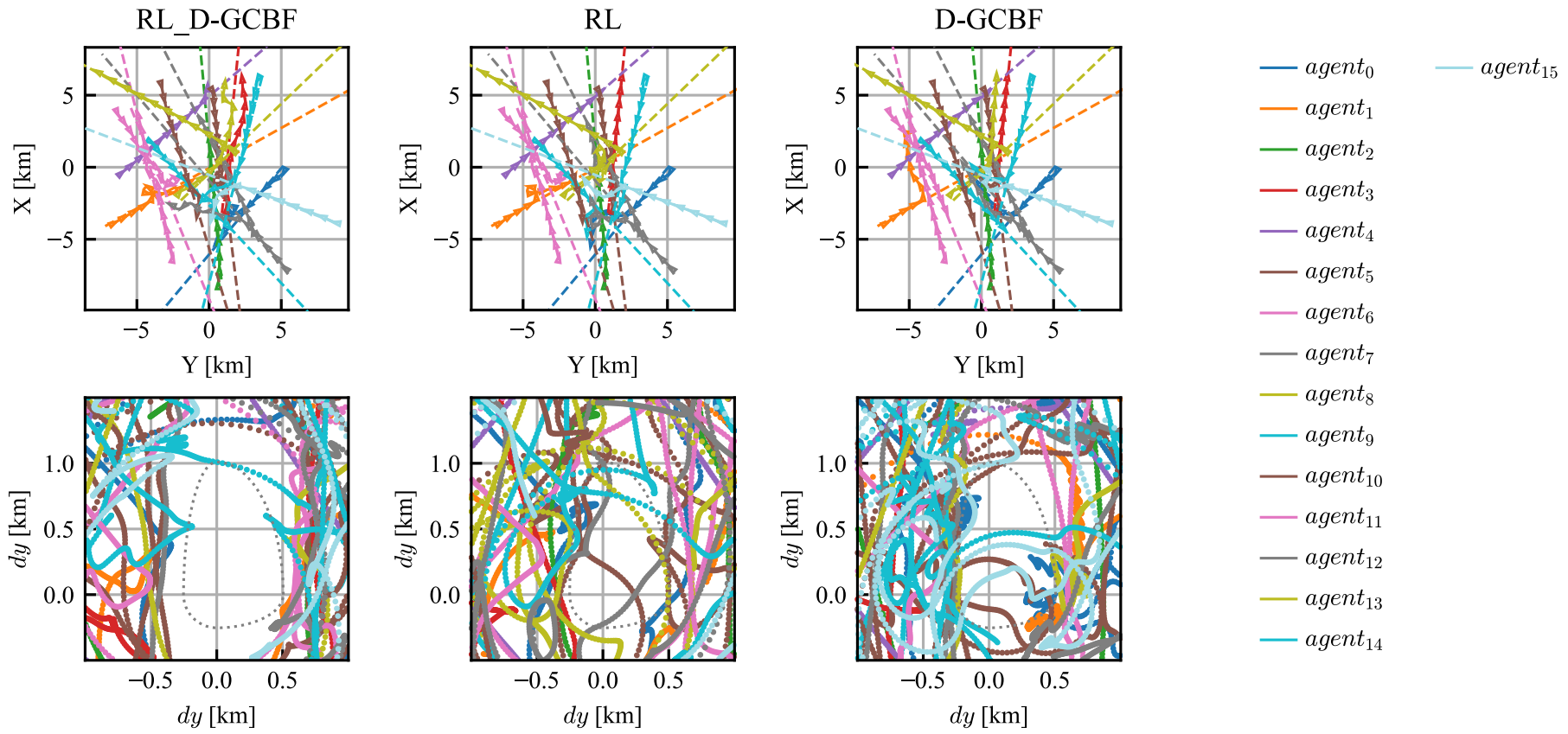
Comparison Result in the Congested Scenario



The AI that trained collision avoidance using D-GCBF did not prevent other ship from entering the ship domain.

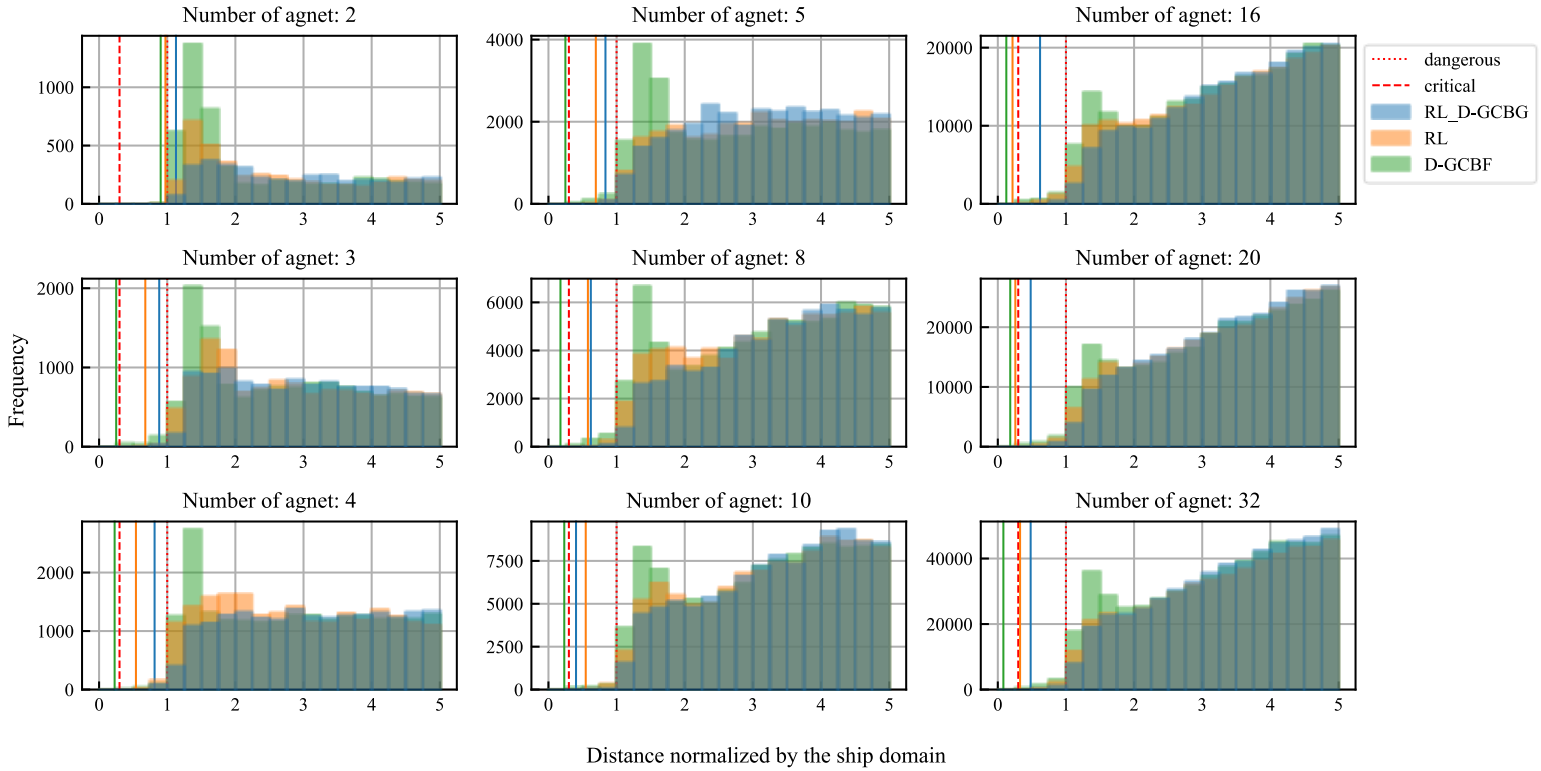
This is because the D-CBF constraint is activated only when other ships approach the boundary of the ship domain.
As a result, in situations where a ship was surrounded by others, it could not prevent domain intrusion and was exposed to danger.

Comparison Result in the Highly Congested Scenario



Only the AI trained RL with D-GCBF controlled ships without ship domain invasion.
Deep reinforcement learning with safety constraint is effective method
for developing a marine traffic control AI.

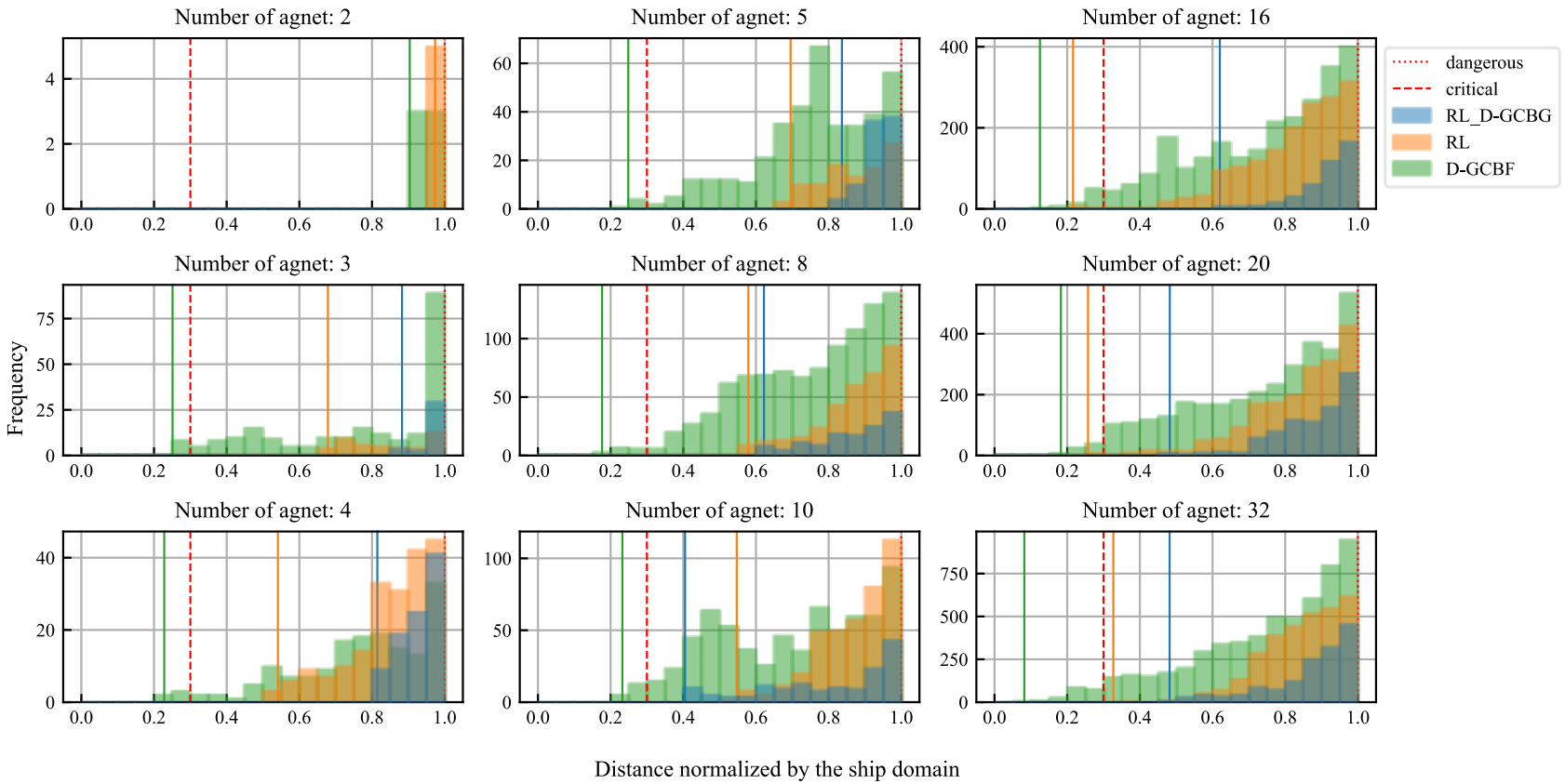
Comparison of the Frequency of Dangerous States



In the D-GCBF results, peaks in the number of dangerous states appear near the danger threshold. It shows a characteristic of CBF.

Since RL maximize the cumulative reward, collision-avoidance action is earlier by predicting future danger.

Comparison of the Frequency of Dangerous States



Among the three AIs, the results of RL with D-GCBF showed the lowest frequency of dangerous states and the largest minimum distance.

This demonstrates that integrating RL with CBF allows safety guarantees on collision avoidance.

Conclusion

We developed a marine traffic control AI trained using deep reinforcement learning with a safety constraint.

- The observations and reward functions were represented using a graph structure.
- A deterministic graph control barrier function (D-GCBF) was introduced, which can be applied to unknown dynamics and off-policy learning methods.
- The safety constraint was incorporated into the DRL framework.
- The proposed AI was validated through numerical experiments.
- Furthermore, it was compared with AIs trained using RL alone and D-GCBF alone.
- The results demonstrate that the proposed framework is an effective method for developing a safe marine traffic control AI.