SCIENCE CHINA
Information Sciences

• **Supplementary File** •

A Versatile Digital Twin Framework for Maritime Communications with Off-the-Shelf Components

Ye Li[1*], Yang Lu[1], Jue Wang[1*], Yang Lu[1], Xiaomin Chen[1], Yi Wu[2*] & Wei Feng[3*]

[1]*School of Information Science and Technology, Nantong University, Nantong, China*
[2]*College of Photonic and Electronic Engineering, Fujian Normal University, Fuzhou, China*
[3]*Department of Electronic Engineering, Tsinghua University, Beijing, China*

Appendix A Energy Efficiency Comparison and Convergence Performance of MATD3, MADDPG, and MAPPO

To better illustrate the performance gain enabled by the proposed framework, we added an energy efficiency improvement curve and compared the proposed MATD3-based method with MADDPG [1] and MAPPO [2]. In this context, EE is formulated as the ratio of the total effective communication throughput to the overall energy consumption, encompassing both the propulsion energy for movement and the transmission energy of the maritime agents. Thus, an improvement in EE reflects the algorithm's enhanced capability to achieve a superior trade-off: maximizing data transmission rates while minimizing energy expenditure during dynamic trajectory planning and resource allocation. The results show that MATD3 achieves the highest converged EE, improving the EE by approximately 11.02% and 3.92% compared with MADDPG and MAPPO, respectively. This comparison demonstrates that the proposed DTN framework can support quantitative algorithm evaluation and energy-efficiency-oriented decision optimization, as illustrated in Fig. 1.

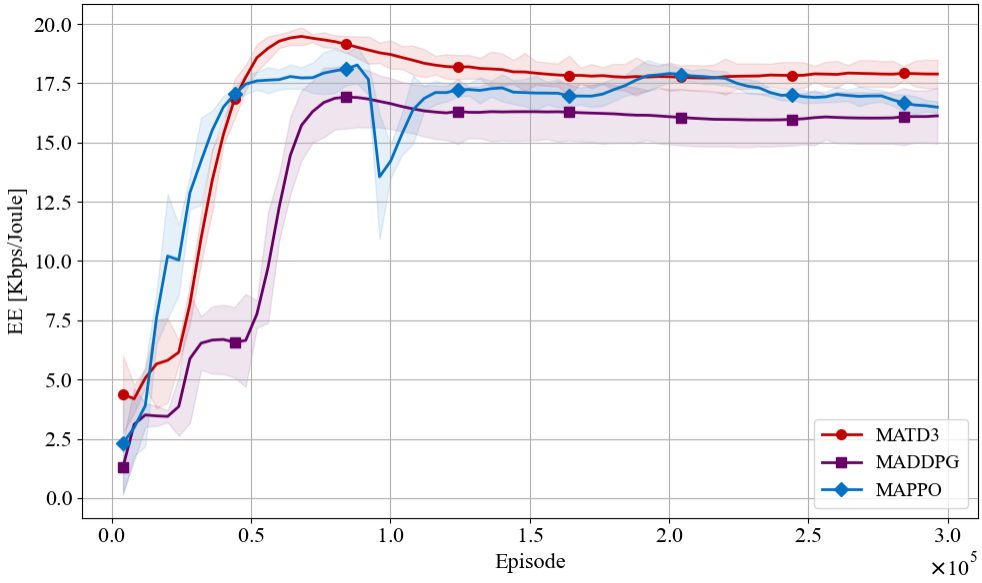

Fig. 1. Energy efficiency comparison among MATD3, MADDPG, and MAPPO

As shown in Fig. 2, MATD3 achieves better performance than MADDPG and MAPPO in terms of average reward. In our reinforcement learning framework, the average reward represents the cumulative system utility per episode, which is formulated as a weighted sum of the communication energy efficiency, connectivity maintenance, and safety constraints (e.g., collision avoidance). Specifically, a higher average reward indicates a more effective coordination strategy among maritime agents to maximize long-term transmission throughput while adhering to energy and environmental limitations.

The performance gain of MATD3 stems from its clipped double-Q learning mechanism, which mitigates the overestimation bias inherent in traditional Actor-Critic methods like MADDPG. Furthermore, by employing delayed policy updates and target policy

* Corresponding author (email: wangjue@ntu.edu.cn, wuyi@fjnu.edu.cn, fengwei@tsinghua.edu.cn)

smoothing, MATD3 exhibits superior robustness in handling the high-variance estimation caused by the dynamic and sparse maritime environment.

MATD3 gradually increases during training and reaches a stable convergence state after approximately 75,000 training episodes. In contrast, MADDPG converges to a lower reward level, while MAPPO shows faster early-stage improvement but exhibits noticeable fluctuations before convergence.

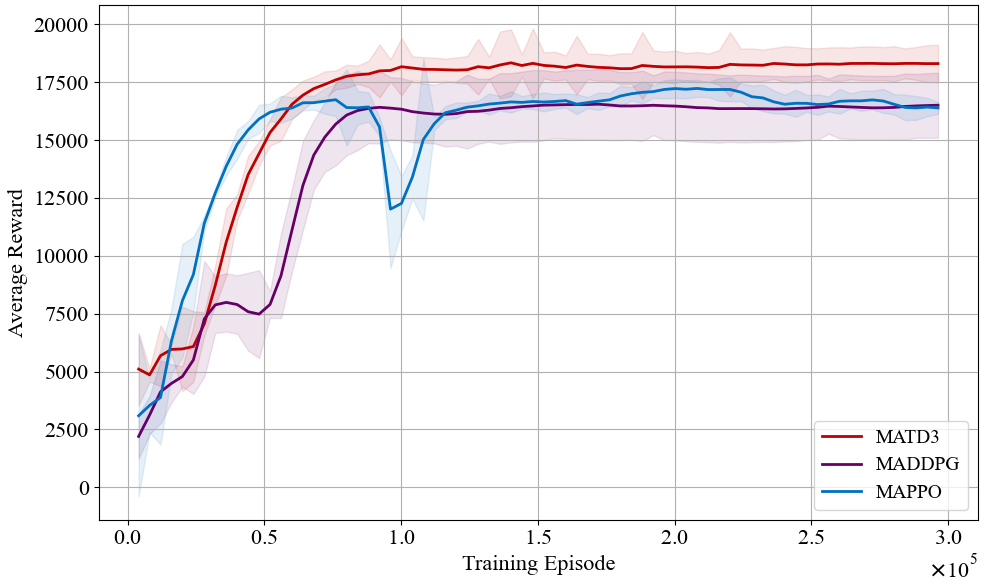

Fig. 2. The convergence of the MATD3, MADDPG and MAPPO

Appendix B Detailed Implementation of the Data Interface and Synchronization Mechanism

To ensure the seamless integration of the visualization layer (Unity3D), the network emulation layer (NS-3), and the intelligent decision-making module, we developed a multifaceted data interface mechanism. This architecture enables a robust closed-loop data flow, which is executed through the following four interconnected procedures.

1) Forward Link: Parameter Transfer and Simulation Triggering. The forward link facilitates the transmission of user-defined maritime scenarios from the visualization interface to the cloud-based emulation engine. Specifically, for coordinate transformation, since the maritime scenario involves global coordinates, the visualization engine first converts user-input geographic coordinates (Longitude λ and Latitude ϕ) into a local Cartesian coordinate system (x, y) using a linear approximation based on a reference origin (λ_0, ϕ_0). The transformation is given by:

$$\begin{cases} x = \frac{(\lambda-\lambda_0)\cdot R\cdot\cos(\phi_0)\cdot\frac{\pi}{180}}{M_{unit}} \\ y = \frac{(\phi-\phi_0)\cdot R\cdot\frac{\pi}{180}}{M_{unit}} \end{cases} \tag{B1}$$

where R denotes the Earth's radius, and M_{unit} is the meters-per-unit scale in the rendering environment, thereby ensuring spatial consistency between the visual rendering and the mobility model. Subsequently, for HTTP-based remote triggering, the transformed coordinates and navigation parameters are serialized into a JSON configuration. An asynchronous HTTP POST request is then dispatched to a Flask-based middleware deployed on a Huawei Cloud virtual machine. The middleware parses the payload and programmatically triggers the execution of the emulation engine, passing the parameters as command-line arguments.

2) Network Emulation. The network emulation layer (NS-3) performs high-fidelity simulations based on the received parameters. Within the simulation environment, the system continuously monitors network-layer metrics including Throughput, Packet Loss Rate (PLR), and Latency. Instead of traditional file-based logging, we designed a database interfacing mechanism by integrating a C++ MySQL connector directly within the simulation process. As the emulation progresses, the system establishes a connection to a remote MySQL database and executes SQL to persist the performance data. Each entry includes a unique synchronization ID and a high-precision timestamp to ensure data traceability.

3) Backward Link: Data Synchronization and Real-time Visualization. To close the loop, the visualization layer must retrieve and render the emulation results in real-time. To this end, an asynchronous database access module is implemented in the frontend. To avoid UI blocking, the system utilizes a dedicated polling logic that queries the database for newly generated records based on the last-read ID and the simulation start time:

```
SELECT * FROM log_test WHERE id > @lastId AND timestamp >= @startTime
ORDER BY id ASC LIMIT 1;
```

The retrieved metrics are parsed and dynamically fed into the visualization component. This allows the DTN framework to display dynamic performance curves (e.g., energy efficiency and throughput), providing users with immediate visual feedback.

4) Interface for Intelligent Algorithms. Finally, the intelligent algorithm module (Python-based MATD3) interacts with the system through the same MySQL-based data bus. It reads historical system states from the database for model training and writes optimized decision parameters back to the database. These parameters are subsequently fetched by the visualization engine for executive control, achieving high modularity and scalability for the proposed maritime DTN framework.

Appendix C Cross-Layer Synchronization Latency Analysis

Table C1 Representative measured cross-layer synchronization latency samples.

Sample No.	SyncId	Simulation time (s)	Synchronization latency (ms)
1	1	1.001	209
2	2	2.001	220
3	3	3.001	111
4	4	4.001	109
5	5	5.001	112
6	6	6.001	112
7	7	7.001	123
8	8	8.001	104
9	9	9.001	82
10	10	10.001	106

Note: The table lists representative samples from the measured synchronization records. The simulation time values appear with a .001s offset because the sender application is configured to start at 1 ms. A total of 29 samples were collected in the full 30 s run and used for the statistical analysis in Table C2.

Table C2 Cross-layer synchronization latency statistics. Unit: ms.

Set	N	Mean $\pm$ SD	Median	Range	P95
All	29	130.3 ± 42.5	110.0	82–220	212
Steady	27	124.1 ± 36.9	109.0	82–212	207

Note: $T_{\text{sync}} = T_{\text{Unity,aligned}} - T_{\text{ns3}}$ after clock-offset calibration. The calibrated clock offset was 485 ms. The steady-state set excludes the first two start-up samples.

Table C2 summarizes the measured cross-layer synchronization latency. Over all 29 synchronized records, the average synchronization latency was 130.3 ms, with a median of 110.0 ms and a 95th percentile of 212 ms. After excluding the first two start-up transient samples, the steady-state average latency decreased to 124.1 ms, with a median of 109.0 ms and a 95th percentile of 207 ms.

Appendix D Scaling Factor γ in Eq. (4)

To ensure the fidelity of the DTN, the framework also inherently permits dynamic environment to be driven by real-time meteorological data and incorporate real-world historical AIS trajectories (from open sources or certain maritime administration entities). Specifically, we may acquire time-series radar grayscale images of the target sea area. Let $I_t \in \mathbb{R}^{H\times W}$ denote the grayscale image matrix at time t. We extract the pixel value at a fixed observation coordinate $p(u, v) = (153, 240)$ to generate the echo intensity vector $\boldsymbol{E}$, defined as:

$$\boldsymbol{E}(t) = I_t(153, 240) \times \gamma, \tag{D1}$$

where $\gamma = 10$ is the scaling factor. This vector $\boldsymbol{E}$ encapsulates the temporal variation of wind speed and precipitation.

The scaling factor γ is introduced to recover the actual physical quantity from the normalized grayscale value. In the radar meteorological data visualization, the original grayscale image maps physical quantities to grayscale values ranging from 0 to 255. After normalization to [0,1], the pixel value is multiplied by γ to obtain the corresponding physical value. In Eq. (4), γ is set to 10 because the corresponding precipitation intensity range is 0–10 mm/h. For other physical quantities, different scaling factors can be used, for example, $\gamma = 35$ for a wind speed range of 0–35 m/s.

Appendix E Evaluation of Weather Effects on Maritime Communication Channel Gain

In complex and dynamic maritime environments, the weather effects on communication channel gain fluctuations can be evaluated by incorporating weather-induced attenuation into the channel model. Specifically, meteorological parameters such as rainfall rate are used to calculate the additional attenuation in dB, and this attenuation is added to the original path-loss model. Consequently, the received power and channel gain decrease with the increase of weather-induced loss. In addition, the effect of wind speed is indirectly represented by antenna oscillation or fast-fading models, which can capture the short-term channel fluctuations caused by vessel motion and dynamic sea conditions.

The proposed framework possesses the capability to integrate environmental factors. To mathematically quantify these dynamic weather effects, the channel model can be formulated as:

$$G_{\text{total}}(t) = G_{\text{tx}} + G_{\text{rx}} - L_{\text{path}}(d) - L_{\text{rain}}(t) - L_{\text{wind}}(t) + F_{\text{fading}}(t) \tag{E1}$$

where G_{tx} and G_{rx} denote the antenna gains of the transmitter and receiver, respectively. $L_{\text{path}}(d)$ represents the distance-dependent path loss. The impact of weather conditions is quantified through the following two primary components:

- **Rain-induced Attenuation (L_{rain}):** Following the ITU-R P.838 standard, the specific attenuation γ_R (dB/km) is calculated based on the rainfall rate R (mm/h):
$$\gamma_R = kR^{\alpha} \tag{E2}$$
where k and α are power-law coefficients depending on the frequency and polarization. The total time-varying rain attenuation is given by $L_{\text{rain}}(t) = \gamma_R \times d_{\text{eff}}(t)$, where d_{eff} is the effective propagation path length.

- **Wind-induced Fluctuation (L_{wind}):** Wind speed primarily affects the stability of maritime vessels, leading to antenna misalignment. Based on the physical kinematics simulated in the DTN visualization layer, the instantaneous pointing loss $L_{\text{wind}}(t)$ is modeled as:
$$L_{\text{wind}}(t) \approx 12 \left(\frac{\theta_{\text{error}}(t)}{\theta_{\text{3dB}}} \right)^2 \quad \text{(dB)} \tag{E3}$$
where $\theta_{\text{error}}(t)$ is the instantaneous misalignment angle caused by wind-induced vessel roll and pitch, and θ_{3dB} is the half-power beamwidth of the antenna.

By integrating these dynamic factors, the proposed framework can be utilized to simulate the impact of weather effects on communication channel gain fluctuations.

Appendix F Sources, Composition, and Preprocessing of AIS and Meteorological Data

(1) Data sources. The AIS data were provided by the Marine Project Department of Fujian Weixing Developer Company., Ltd. The meteorological data were obtained from radar grayscale image analysis.

(2) Data composition. The AIS data include the MMSI number, longitude, latitude, timestamp, sailing speed, course over ground, and ship heading. The meteorological data include the data ID, wind speed, and rainfall.

(3) AIS Data Preprocessing.

- Acquire the AIS data table by querying the database.
- Handle missing and abnormal values using the mean-value method.
- Remove abnormal values and supplement missing values, considering spatiotemporal continuity of trajectories.
- Parse and sort the trajectory data:
  - Convert the timestamp into a DateTime object.
  - Arrange the data in ascending temporal order.
- Transform longitude and latitude into Unity world coordinates based on predefined geographic boundaries.

(4) Meteorological Data Preprocessing.

- Read the original PNG grayscale radar images.
- Normalize pixel values to the range [0,1].
- Apply geometric corrections (e.g., vertical flipping) to ensure consistency with geographic orientation.
- Recover physical quantities by multiplying normalized grayscale values by corresponding scaling factors.
- Extract target pixel values for the selected region.
- Batch-process time-series images and store them as arrays for subsequent modeling and simulation input.

Appendix G Communication Resources Optimized by the MATD3 Algorithm

In the proposed MATD3-based framework, the optimization focuses primarily on the UAV acceleration and heading angle, which directly determine the UAV trajectory, velocity, and propulsion energy consumption. By adjusting these variables, MATD3 effectively optimizes the spatial-temporal allocation of UAV communication services, thereby improving the utilization efficiency of communication resources.

In the current implementation, the UAV transmit power and bandwidth are fixed simulation parameters. However, the achievable transmission rate for each UAV-vessel link is indirectly influenced by the optimized UAV trajectory. Specifically, as the UAV moves, the relative distance between the UAV and vessels changes, which affects the channel gain, signal-to-noise ratio (SNR), and achievable data rate. Under the minimum rate constraint, MATD3 maximizes the system energy efficiency, defined as the total transmitted bits divided by the total UAV energy consumption over all time slots, to improve the overall utilization of communication resources.

The system energy efficiency (EE) is expressed as

$$\text{EE} = \frac{\sum_{t=1}^{T} \sum_{u=1}^{N_U} R_{u,ms,t}\, \lambda_t}{\sum_{t=1}^{T} \sum_{u=1}^{N_U} E_u(t)} \tag{7}$$

where $R_{u,ms,t}$ denotes the achievable transmission rate of UAV u to vessel ms at time slot t, $E_u(t)$ represents the energy consumption of UAV u at time t, and λ_t is the duration of the time slot. While the UAV transmit power and bandwidth are fixed, the optimized UAV trajectory changes the relative positions to vessels, which directly influences $R_{u,ms,t}$ via the distance-dependent channel gain. By maximizing EE under the minimum rate constraints, MATD3 improves communication resource utilization in terms of both spatial and temporal allocation.

Appendix H Decision Layer Aggregation in Multi-UAV-Assisted Maritime Communication

In the multi-UAV-assisted maritime communication scenario, the decision layer aggregates system states and network metrics using a *centralized training with decentralized execution* (CTDE) paradigm, which is widely adopted in multi-agent reinforcement learning.

During the centralized training stage, the states of all UAVs and maritime vessels are collected and stored in a shared experience buffer. Specifically, the state of each UAV u at time t can be expressed as

$$s_u(t) = \{\boldsymbol{p}_u(t), \boldsymbol{p}_{\text{vessels}}(t), \boldsymbol{p}_{o\neq u}(t), v_u(t), a_u(t), \theta_u(t)\}, \tag{H1}$$

where $\boldsymbol{p}_u(t)$ is the position of UAV u, $\boldsymbol{p}_{\text{vessels}}(t)$ are the positions of all target vessels, $\boldsymbol{p}_{o\neq u}(t)$ are the positions of other UAVs, $v_u(t)$ is the UAV speed, $a_u(t)$ is its acceleration, and $\theta_u(t)$ is its heading angle.

The UAVs' actions are the acceleration and heading angle, denoted as

$$a_u(t) = \{a_u(t), \theta_u(t)\}, \tag{H2}$$

which determine the UAV trajectories, velocities, and corresponding energy consumption.

The reward function integrates multiple network and task-related objectives, including a flight reward r_u^{flight}, a coverage reward r_u^{cov}, and an energy-efficiency reward r_u^{EE}, i.e.,

$$r_u(t) = r_u^{\text{flight}} + r_u^{\text{cov}} + r_u^{\text{EE}}. \tag{H3}$$

The energy-efficiency reward is calculated based on the ratio between transmitted bits and UAV energy consumption over the time slot, reflecting the system-level objective.

After training, the learned actor network parameters are distributed to each UAV. During decentralized execution, each UAV makes decisions solely based on its local observation $s_u(t)$, without requiring real-time centralized control. This approach enables cooperative maritime coverage and energy-efficient communication in a scalable manner.

Appendix I Extending the Framework with Physical-Layer Transmission Details

The proposed framework adopts a modular architecture, where the visualization layer, network emulation layer, and intelligent decision layer are connected through a unified data interface. Therefore, additional physical-layer transmission details can be integrated without changing the overall framework structure. Specifically, environmental and motion-related states generated in the visualization layer can be further converted into channel-related parameters and passed to the network emulation layer.

For example, precipitation intensity can be mapped to an additional propagation attenuation term and incorporated into the path-loss model. Wind- and wave-induced vessel motion can be mapped to antenna attitude variation, link-distance variation, and channel-gain fluctuation. Sea-surface reflection can be further modeled through two-ray or multipath propagation models. These physical-layer factors can then be used to update the channel gain, received power, SINR, achievable rate, and packet-level performance metrics in the network emulation layer.

From the implementation perspective, this extension can be achieved by adding corresponding physical-layer fields, such as rain attenuation, antenna orientation, Doppler shift, or reflection loss, into the JSON/database-based interface, and then extending the channel or propagation-loss models in NS-3. In this way, the proposed framework can support more fine-grained physical-layer simulation while maintaining its modular and extensible structure.

Appendix J Role of Vessel Motion Equations in Visualization and Potential Physical-Layer Modeling

The construction of DTN starts with drag-and-connect of pre-configured 3D modules in Unity 3D to specify the position and movement of the objects. To accurately twin the maritime scenario, we can model wave-induced vessel motions by decomposing the hull geometry. Specifically, the hull buoyancy is approximated by a set of primitive volumes (e.g., spheres) whose submerged areas can be efficiently computed and aggregated. The buoyancy force on a primitive is modeled as

$$\boldsymbol{F}_f = \begin{cases} \mathbf{0}, & \xi(p) \leqslant p_y, \\ Sg\rho\big(\xi(p) - p_y\big)\, n_y\, \mathbf{y}, & \xi(p) > p_y. \end{cases} \tag{1}$$

where $\boldsymbol{F}_f$ denotes the buoyancy force vector, $\xi(p)$ denotes the water level at position p, p is the centroid position of the submerged sphere, S is the submerged area, ρ is the fluid density, g is the acceleration of gravity, n is the centroid vector, n_y is its y-component, and $\boldsymbol{y}$ is the unit vector along the vertical direction.

$$\boldsymbol{F}_{\text{lift}} = \frac{1}{2}\rho C_L |S_{\text{face}}||\boldsymbol{U}| \left(\boldsymbol{U} \times \frac{\boldsymbol{n} \times \boldsymbol{U}}{|\boldsymbol{n} \times \boldsymbol{U}|}\right), \tag{2}$$

$$\boldsymbol{F}_{\text{drag}} = -\frac{1}{2}\rho C_D |S_{\text{face}}||\boldsymbol{U}|\, \boldsymbol{U}. \tag{3}$$

where C_L and C_D are the lift/drag coefficients, $\boldsymbol{U}$ is the relative velocity between the primitive and the fluid, $\boldsymbol{n}$ is the surface normal, and S_{face} is the effective projected area. These models can be conveniently implemented in the visualization layer (e.g. Unity3D in this particular case), enabling the scene to update vessel pose (position/attitude) at each simulation tick while remaining computationally tractable.

Equations (2) and (3) are used to model the lift force and drag force acting on the vessel in fluid, respectively. These equations are mainly implemented in the visualization layer, such as Unity3D, to update the vessel posture, including its position and attitude, at each simulation step.

In the current prototype, the above vessel-motion-induced effects have not been fully coupled into the physical-layer link quality calculation of the network emulation layer. It should be clarified that the use of these equations is not limited to visual rendering. They can generate physical motion states of the vessel, which can further provide physical inputs for communication modeling. For example, vessel attitude variations induced by lift, drag, wind, and waves may lead to changes in antenna position and orientation. These changes can affect the link distance, antenna alignment, and sea-surface reflection path, thereby causing channel gain fluctuations [3]. The proposed framework supports such an extension through its modular data interface. Specifically, the updated antenna position and attitude obtained from the visualization layer can be passed to the channel model, where models such as the two-ray propagation model can be used to evaluate their effects on received power, channel gain, SINR, and achievable rate.

Appendix K Role of Vessel Motion Equations in Visualization and Potential Physical-Layer Modeling

In addition to shipborne base-station coverage enhancement and multi-UAV-assisted maritime communication, the proposed platform can serve as a cost-effective testbed for validating next-generation maritime communication strategies. The potential application scenarios include the following:

(1) Maritime emergency rescue and search-and-rescue operations. By incorporating real-time ocean current and wind direction data into the physical engine and combining them with intelligent algorithms, the proposed framework can be used to predict the drifting trajectory of distressed vessels or floating targets. This can support emergency communication planning and rescue resource deployment.

(2) IoT management for offshore infrastructures. Due to its modularity and scalability, the proposed DTN framework can be applied to smart marine ranches, offshore wind farms, and other maritime IoT scenarios. It can help optimize data collection trajectories and edge-computing offloading strategies for massive and sparsely distributed IoT sensors.

(3) Smart port automation and blockage management. The visualization layer can be used to construct port and dock models, while the network emulation layer can simulate wireless communication performance in complex metallic environments. This enables the evaluation of physical blockage, multipath effects, and communication reliability for automated vehicles, cranes, and port-side devices.

References

1 Lowe R, Wu Y I, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. Adv Neural Inf Process Syst, 2017, 30: 6379–6390
2 Yu C, Velu A, Vinitsky E, et al. The surprising effectiveness of PPO in cooperative multi-agent games. Adv Neural Inf Process Syst, 2022, 35: 24611–24624
3 Sun S, Guo Y, Tao M, et al. Modeling and analysis of land-to-ship maritime wireless channels at 5.8 GHz. IEEE Trans Wireless Commun, 2026, 25: 10051–10065.