Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.078280

ARTICLE

Multi-View Latent Imitation Learning with Mamba-Based Action Encoding for Unmanned Surface Vehicle Navigation

Manh-Tuan Ha[1], **Nhu-Nghia Bui**[2], **Dinh-Quy Vu**[1,*] **and Thai-Viet Dang**[2,*]

[1]Department of Vehicle and Energy Conversion Engineering, School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam

[2]Department of Mechatronics, School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam

*Corresponding Authors: Dinh-Quy Vu. Email: quy.vudinh@hust.edu.vn; Thai-Viet Dang. Email: viet.dangthai@hust.edu.vn

Received: 28 December 2025; Accepted: 15 April 2026; Published: 08 May 2026

ABSTRACT: The development of Unmanned Surface Vehicles (USVs) has become a key focus in marine robotics, fueling the need for navigation systems capable of performing complex and delicate tasks with speed and precision. However, the end-to-end path tracking process often encounters challenges in learning efficiency, and generalization, and varying environmental conditions. To achieve sample-efficient and robust USV navigation in dynamic maritime environments, the paper proposes a novel hierarchical multi-view latent imitation learning (IL) architecture. By formulating a latent IL objective, the framework disentangles diverse navigation modalities through continuous variables, preventing mode collapse and enhancing behavioral adaptability to non-stationary conditions. High-dimensional multi-view observations are transformed via a ViT-based backbone into compressed latent features to minimize redundant environmental information. These representations are processed by a Mamba-based action encoder, which leverages selective state-space modeling to capture long-term temporal dependencies with high computational efficiency. A UNet-based decoder subsequently forecasts optimal action sequences by synthesizing spatial maps to infer critical environment-agent relationships. This preliminary multi-view latent IL-based trajectory ensures precise tracking and dynamic stability while adhering to physical vehicle constraints. Experimental results validate that this end-to-end approach achieves robust path planning effectiveness, obstacle avoidance capability, and model training efficiency in complex, multi-modal maritime scenarios.

KEYWORDS: Latent imitation learning; unmanned surface vehicles; latent space model; state-space models; multi-modal feature fusion

1 Introduction

The advancement of unmanned surface vehicles (USVs) has become a focal point in marine robotics, necessitating navigation systems capable of handling complex and delicate tasks with high speed and precision [1]. To expand the practical utility of these autonomous agents, robots must understand and acquire complex human-like movements to operate effectively in unstructured environments. Traditional control methods often struggle with the high-dimensional state spaces inherent in marine navigation, leading researchers toward end-to-end data-driven approaches [2]. Among these, imitation learning (IL) has emerged as a vital technique, enabling USVs to learn sophisticated policies from expert demonstrations with minimal learning costs [3]. Unlike standard reinforcement learning, which often faces challenges with sparse rewards and high sample complexity, IL provides a reliable mechanism to learn directly from prior knowledge and historical experiences. Furthermore, integrating IL allows for the generation of trajectories

that maintain human-like smoothness while adhering to the physical constraints of the vehicle. By leveraging latent representations, IL can capture the underlying intent of a demonstrator, making it particularly suitable for long-term and diverse trajectory generation in dynamic maritime scenarios. Consequently, IL serves as a cornerstone for developing robust USV navigation strategies that balance operational efficiency with safety.

While traditional navigation algorithms like A* or Dijkstra rely on static environmental assumptions and predefined rules [4,5], they frequently fail in the real-time, dynamic scenarios encountered by USVs. Model predictive control (MPC) offers a more sophisticated alternative by accounting for future states and system constraints [6], yet its computational overhead and dependence on accurate mathematical models limit real-time deployment in high-dimensional continuous systems [7]. Deep reinforcement learning (DRL) has recently demonstrated superiority in optimal control, yet its practical implementation is often hindered by slow convergence and the difficulty of designing precise reward functions for unstructured environments [8]. To bridge these gaps, modern frameworks have integrated IL with DRL, such as through generative adversarial imitation learning (GAIL), which extracts implicit characteristics from demonstration data to achieve higher robustness and generalization [9]. A critical evolution in this domain is the transition to visual imitation learning (VIL), which processes raw pixel values from video demonstrations to determine actionable robot policies [10]. However, a single-view approach often suffers from information overload and an inability to discern critical spatial relationships. The adoption of bird's eye view (BEV) and multi-view features addresses this by providing a comprehensive global perspective, clearly displaying the spatial layout of surrounding obstacles and traffic participants [11]. Furthermore, multi-view latent imitation learning allows the system to focus on pertinent features while minimizing redundant information from buildings or environmental clutter. By utilizing latent variable models, such as those based on conditional variational auto-encoders (CVAE) [12], these systems can disentangle multiple modalities of expert behavior, preventing the "mode collapse" common in vanilla GAIL. This enables the USV to adapt its navigation strategy to diverse conditions, such as varying traffic patterns or adverse weather, by interpolating between learned latent variables. Multi-view latent IL also facilitates the extraction of condensed prior trajectories that capture the demonstrator's intent more effectively than pixel-to-action mapping [13]. Recent studies have shown that fusing BEV features with latent states significantly enhances the perception of dynamic environments, providing a more comprehensive understanding for decision-making [14]. This shift transforms the technology from purely data-driven to behavior-driven, allowing for global optimization through the backpropagation of loss across perception and planning modules. Additionally, integrating agent-agnostic rewards and efficient exploration around waypoints further improve sample efficiency in complex continuous scenarios. Compared to traditional modular pipelines, the multi-view latent approach reduces communication delays and enhances response speed. It also demonstrates better generalization ability by utilizing pre-training and feedforward prior experience. Ultimately, multi-view latent IL outperforms single-modality approaches by capturing the diverse and complex nature of real-world trajectories while maintaining high prediction accuracy over long horizons.

The efficiency of translating these high-dimensional latent features into executable actions depends heavily on the model's encoding-decoding architecture. Traditional structures, such as transformer decoders or recurrent neural networks (RNNs), are frequently employed in end-to-end planning to process sensor data into control commands like acceleration and steering [15]. However, these traditional models often encounter significant challenges with "covariate shift" and error accumulation when generating long-term trajectories. In complex robotic tasks, the system must reason over thousands of image frames, which can lead to an overload of information if the encoding is not sufficiently condensed. Emerging research suggests that extracting critical waypoints from demonstrator trajectories is more efficient than reasoning over every frame, as it allows the robot to focus on the intent rather than extraneous environmental details.

Mamba-based architecture, as a modern alternative to traditional sequence models, offers potential for more efficient action encoding by leveraging state-space models that scale linearly with sequence length [16]. This efficiency is crucial for USV navigation, where the agent must integrate perception, prediction, and planning in real-time within a single differentiable framework. Efficient encoding allows the system to construct a low-dimensional motion manifold, reducing the computational burden on the imitation learning model. By effectively mapping latent state features to action values, a Mamba-based encoder can enhance the stability and accuracy of path tracking even in transferred driving scenarios. Furthermore, this architecture facilitates dual-module collaborative updates, ensuring synergy between feature extraction and decision-making modules [17]. Such optimized action encoding is essential for real-time obstacle avoidance, enabling the USV to re-plan paths and adjust priorities dynamically in complex workshop or maritime environments. Modern encoding strategies are moving beyond simple state-action mapping toward behavior-driven frameworks that utilize fused features for global optimization. These frameworks benefit from pre-training on expert experiences, allowing agents to master planning in common scenarios quickly. Moreover, advanced imitation learning techniques now extend to action values estimated by value networks, further facilitating the acquisition of decision-making capabilities [18]. The integration of residual learners can also leverage insights into morphological differences and sensor noise to achieve few-shot learning on new tasks. Compared to traditional encoders, a Mamba-based approach can better maintain temporal consistency across long horizons, which is vital for maintaining high task success rates in high-speed operations. Therefore, a Mamba-based action encoding module provides the necessary scalability to handle the multi-modal and highly complex nature of marine navigation data. It supports the development of policies that display natural movements and higher robustness compared to traditional baseline models. This structural innovation represents a significant step toward achieving reliable and efficient autonomous navigation in dynamic, uncertain environments.

Driven by the need for more sample-efficient and robust navigation in dynamic maritime environments, the paper proposes a novel motion planning framework designed for USV, utilizing an imitation learning approach as its core mechanism. By utilizing a latent imitation learning objective, the system disentangles diverse navigation modalities through a continuous latent variable, preventing mode collapse and allowing the USV to adapt its behavior to varying maritime conditions. Data from scenarios in known environments is collected, and optimized simulation paths are generated to create an expert dataset. The input environment is observed from multiple viewpoints and transformed into feature vectors using the ViT network. These features are then compressed into a latent space and passed to the action encoder block. The Mamba model is employed to extract long-term action features from the latent space representation. The UNet block decodes this information to predict actions. A spatial map is synthesized, enabling action inference based on the relationship between the environment and navigation agents. The model forecasts a continuous sequence of actions corresponding to near-future positions, thereby forming a preliminary optimal navigation trajectory grounded in the expert data. Finally, experimental results demonstrate that our approach achieves superior path planning effectiveness, obstacle avoidance capability, and model training efficiency in complex, multi-modal environments. This study provides a novel and efficient control scheme for improving the end-to-end control performance of autonomous vehicles in highly uncertain maritime scenarios.

The main contributions of this work are as follows:

- Propose a long-term information-based synchronization learning framework applied to USVs based on the effectiveness of the Mamba architecture in exploiting the relationships between action chains in diverse navigation scenarios.

- A novel methodology for normalizing the environment using overlapping geometric features to transform state-action features in both simulated and real environments has been developed to improve the deployment capability of USVs in practical applications.
- Diffusion Denoising Policies (DDPs) have been integrated to enhance the robustness and adaptability of the model when handling noise in both the training dataset and during the inference of optimal real-world actions.

The remainder of this paper is organized as follows: Section 2 reviews related work in the field. Section 3 details the architecture of the proposed latent imitation learning with Mamba-based USV navigation architecture. Section 4 presents experimental results and comparative analyses. Finally, Section 5 offers concluding remarks and outlines directions for future research.

2 Related Works

2.1 Traditional Imitation Learning Approaches in Navigation

Traditional imitation learning has laid the groundwork for developing advanced robotic behaviors by mimicking expert demonstrations, thereby avoiding the complex task of manually designing reward functions required in reinforcement learning (RL) [19]. Early methods, such as behavior cloning (BC), treat policy learning as a supervised learning problem by directly mapping state-action pairs [20]. However, these approaches often suffer from "compounding errors" where small mistakes in sequential decisions cause the agent to encounter unfamiliar states outside the training distribution. In navigation, initial machine learning solutions typically rely on data generated by classical planners such as rapidly-exploring random trees (RRT), probabilistic roadmaps (PRM), or A* algorithms to create expert trajectories [21]. While effective, these methods have limitations in adapting to dynamic environments and handling complex vehicle kinematic constraints. Additionally, traditional BC models' inability to incorporate temporal historical information makes them vulnerable to "covariate shift", significantly reducing their real-world control performance.

2.2 Advanced Imitation Learning Techniques for Navigation

To overcome the limitations of traditional methods, modern imitation learning frameworks have evolved toward distributed architectures and latent variable matching models. GAIL exemplifies this advancement by using an adversarial training setup between a policy network and a discriminator network, optimizing the alignment between agent-generated and expert trajectories [9,22]. Given the inherently multimodal nature of navigation tasks, recent studies have incorporated latent variable models such as variational auto-encoders (VAEs) and conditional VAEs (CVAEs) to encode complex trajectory distributions into continuous, controllable latent spaces [23]. This approach helps prevent mode collapse and enables agents to generate diverse, adaptable trajectories responsive to varying environmental conditions. Moreover, hybrid frameworks that combine IL and RL, such as GAPPO [24] and SILP+ [25], have shown synergistic benefits by leveraging expert knowledge to guide exploration while using reward signals to refine policy performance. These advancements are especially important in USV navigation, where reward signals are often sparse and environmental uncertainties, like obstacles, are common.

2.3 Synergistic Integration of Planning, Action Encoding, and Control

Modern navigation systems increasingly focus on the seamless integration of feature recognition and efficient action encoding [26]. The use of BEV representations has become a key technique, providing a comprehensive spatial overview of environmental layouts and inter-entity relationships, which reduces redundant information and communication latency. In terms of action encoding, the Mamba architecture is

gradually replacing traditional Transformer models due to its ability to handle long-term action sequences with linear computational complexity, ensuring temporal contextual consistency during navigation [17,18]. To ensure trajectory safety and smoothness, intelligent hybrid path planners and controllers are often incorporated as an optimization layer, enabling adherence to machine learning-generated paths while strictly enforcing physical constraints [27]. This integrative approach creates a seamless process from visual perception to action-based decision-making, allowing holistic optimization of navigation systems in complex maritime environments. As a result, USV agents not only replicate expert trajectories but also gain the ability to proactively anticipate and adapt their behaviors in response to dynamic environmental changes.

3 Proposed Method

This section presents the Mamba-based architecture integrated with the UNet Decoder. The overall architecture consists of three main components: the Vision Encoder (ViT), the Action Encoder (state-space model), and the UNet Decoder, in Fig. 1. The ViT Encoder extracts features from the input environment, while the Action Encoder processes the corresponding action information to generate predictions tailored to the environmental context. Mamba employs state-space model blocks to establish strong temporal relationships between actions at different time steps and to link the action sequence with the corresponding image sequence. Finally, the UNet Decoder processes and synthesizes the spatial map, making final inferences based on the interplay between actions and the environmental context.

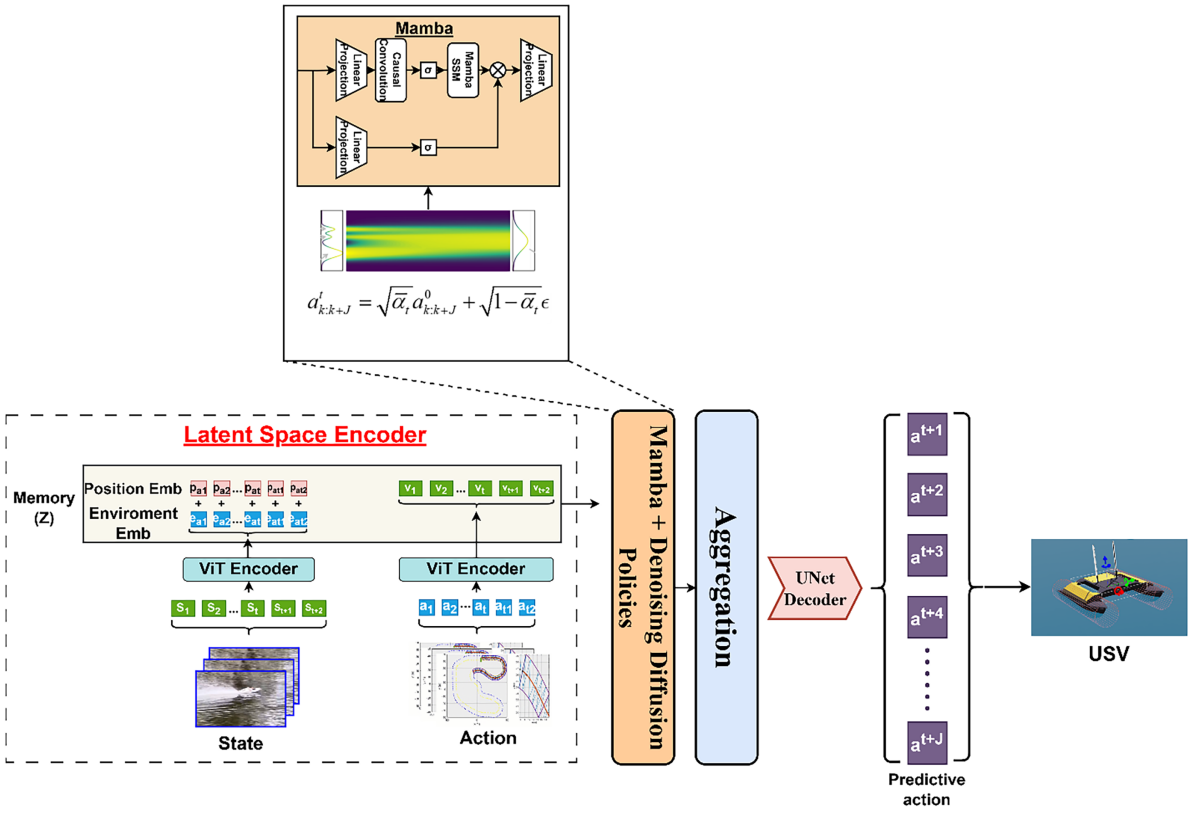

Figure 1: Proposed latent imitation learning with Mamba-based USV's navigation architecture.

The implementation of the imitation learning framework begins with creating an expert dataset in a simulated environment. State-action pairs are mapped into a shared latent space, enabling the integration of

discrete visual features and navigational data into unified mathematical representations. This alignment of feature dimensions helps reduce the gap between simulation and real-world scenarios [28]. The process is carried out through feature unification as described in Eq. (1):

$$\mathcal{Z}_{total} = \text{Aggregate}\left(z_{state}, v_{action}, \text{time}\right) \tag{1}$$

3.1 Latent Space Encoder

State Encoder

Observations captured from multiple perspectives are processed using the Vision Transformer (ViT) module to create a latent representation (Fig. 2). At each time step, the input comprises color images examined to extract relevant features. The input image is divided into N non-overlapping patches, each measuring $P \times P$ pixels. Each patch is then flattened into a vector and passed through a linear transformation that projects it into the embedding space, producing a projection matrix. This procedure is formally described by the following Eq. (2):

$$z_i = E \cdot Flatten\left(x_k^{(i)}\right) \tag{2}$$

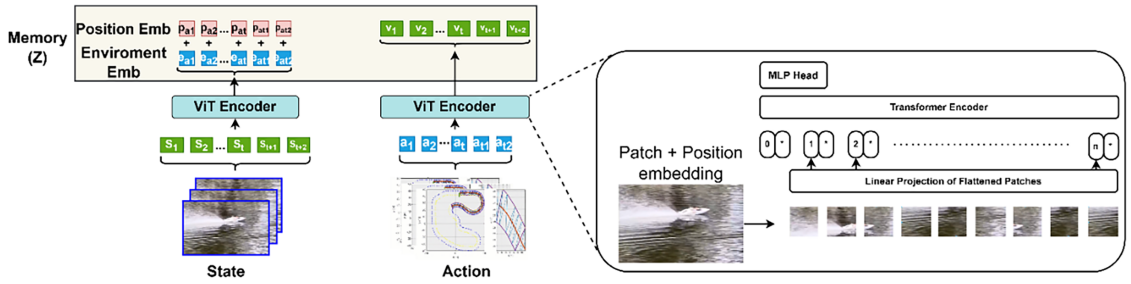

Figure 2: Latent space encoder.

Let $i = 1, \ldots, N$ denote the indices, and let k represent the associated data reference point. The embedding matrix is denoted by E. Each patch is augmented with a corresponding embedding vector p, and the initial element of the sequence is concatenated with a designated token to integrate global attributes. These token sequences are subsequently processed through a series of Transformer encoder layers. Within each layer, the multi-head self-attention mechanism enables the model to capture global dependency relationships among patches within the environment. The interactions between tokens are modeled via the query (Q), key (K), and value (V) components. This specific mechanism is formally described by the accompanying Eq. (3):

$$Attention\left(Q, K, V\right) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

in which, query (Q), key (K), and value (V) are all inferred from the embedding image arrays through linear projection layers. These representations allow the model to capture the relationships between different spatial regions extracted from the initial training dataset and throughout the training process. The key (K) consists of a sequence of vectors that act as "labels" for the image regions, covering all recognizable areas within the image. The value (V) contains semantic information corresponding to these image regions, providing detailed data for each respective reference. The final decoder block used is a multilayer perception (MLP), which takes input tensors of appropriate shapes. Its output is a tensor with dimensions [B, num_action], where B denotes the batch size, and num_action corresponds to the predicted steering and spinning

commands. The resulting output (Z) is a vector that encapsulates an "intelligent summary" of the image, refined to highlight the most salient regions relevant to the robot's next action. The final MLP layer is used to extract output information into representations in the latent space. The vectors representing the two properties are the environmental representation (e_{at}) and the positional representation (p_{at}). The output of the transformer encoder undergoes a series of linear transformations followed by a non-linear activation function (ReLU). This process is illustrated in Eq. (4):

$$Z = \sigma\left(W_L \cdot \left(\ldots \sigma\left(W_1 h_{enc} + b_1\right)\ldots\right) + b_L\right) \tag{4}$$

where W represents the weight matrix of the hidden layers, and B denotes the bias vector. The nonlinear activation function σ is applied to the data. Z is the latent vector that contains the state information of the environment. Using Z, the model performs projection operations to manipulate the dynamic properties of the location through the projection matrix and slicing vector. Environmental features, which include information about the landscape, obstacles, and targets, are extracted. This process is described in Eq. (5):

$$e_{at} = W_{e_{at}} Z + b_{e_{at}} \tag{5}$$

where the environment embedding (e_{at}) encapsulates the characteristic memory structure of the environment at time t. Likewise, the extracted positional features (p_{at}) of the navigational agent convey information about the relative coordinates and orientation of the USV within the observation frame. This process is described in Eq. (6):

$$p_{at} = W_{p_{at}} Z + b_{p_{at}} \tag{6}$$

Therefore, both the e_{at} and p_{at} components are combined by concatenation to form a fully represented entity in the latent space, creating the final state memory as shown in Eq. (7):

$$Z_{e-p} = e_{at} \oplus p_{at} \tag{7}$$

3.2 Action Encoder

According to the architectural diagram (Fig. 1), the Action Encoder is responsible for encoding and synthesizing action attributes from the expert dataset to provide action information input for the Mamba block and the subsequent Aggregation block. It takes as input a sequence of known actions or actions to be predicted in the near future. These actions, represented as either discrete or continuous depending on the Sequence length parameter, are transformed into meaningful feature vectors that occupy the same vector space as state feature vectors and other learnable variables. This alignment is essential to enable the subsequent Mamba layers to jointly combine and process all types of information, including state, time, and action.

The Action Encoder processes a sequence of actions starting from the current action a_k^{t+i} at time t and extending through a series of predicted future steps. We denote the sequence of actions at time t as $a_k^t, a_k^{t+1}, \ldots, a_k^{t+J}$. The subsequent actions $a_k^{t+1}, \ldots, a_k^{t+J}$ represent either the model's predicted future actions or externally provided actions intended to guide the prediction of the state variable, known as the value. Secondly, the encoding process transforms raw input data: when actions are discrete (e.g., "turn left", "go straight") or sparse, the Action Encoder projects them into a higher-dimensional embedding space, converting each action a_k^{t+i} into a feature vector v_a^{t+i}. These encoded action vectors are then enhanced with Positional Encoding (PE), allowing the model to capture the relative temporal positions of actions within

the sequence $(t, t+1, \ldots, t+J)$. Hence, the computational process for estimating the action feature vector is illustrated in Eq. (8):

$$v_a^{t+i} = \left(E_{act} \cdot a_k^{t+i}\right) + \mathrm{PE}\left(t+i\right) \tag{8}$$

where PE is the vector that encodes the position corresponding to the time step in the sequence. This vector v forms the complete action feature vector, which is then ready to be fed into subsequent Mamba layers and Aggregation blocks. Integration with the overall model architecture proceeds as follows: the encoded action vectors, enriched with positional information, are input into the second processing stage alongside Learnable State Variables and Learnable Time Variables. This combined input is processed by the second Mamba layer. The primary goal is for the Mamba and Aggregation layers to effectively learn the complex relationships among past states (encoded by the first Mamba layer), learnable parameters representing general information, and future actions (encoded by the Action Encoder).

3.3 Mamba-Based Imitation Learning with Denoising Diffusion Policies

The goal of imitative learning is defined in terms of action sequences conditioned by latent state representations. Instead of directly regressing the expert's actions, this policy learns to denoise the biased action sequences within the diffusion framework, enabling robust modeling of the expert's multi-step behaviors. Modeling or simulating the Mamba architecture is a fundamental aspect of this methodology (see Fig. 3). Compared to other approaches, the unique characteristics of expert data in USV motion planning offer several advantages to this method. Convolutional neural network (CNN) architectures, which primarily rely on filter operations, often struggle to effectively analyze and capture localized information within the training dataset. In contrast, Transformer models tend to perform well mainly when applied to relatively large datasets.

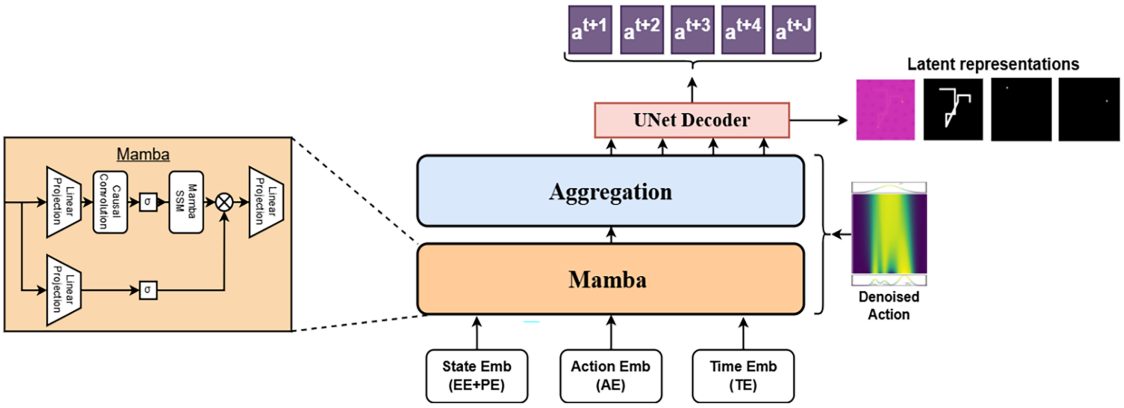

Figure 3: Mamba-based imitation learning with denoising diffusion policies.

Additionally, the scalability of self-attention mechanisms with long sequences poses significant challenges due to their computational complexity of $O(n^2)$. The Mamba architecture addresses these issues by catering to the specific features of action navigation systems for USVs. Its streamlined computational complexity of $O(n)$ makes it well-suited for environments with limited computational resources, while still delivering fast performance and inference speeds comparable to those of Transformer models. The particular nature of agent and state data in the operational environment requires the extraction of robust linear features, which can be effectively achieved through the stage space model. The coupling mechanism within the Mamba architecture is implemented via DDPs. DDPs model sequences of future actions through a process

of sequential diffusion and denoising, enabling the representation of multi-node action distributions and ensuring temporal consistency. Let $a^0_{k:k+J}$ represent a sequence of actions of length $j+1$ drawn from the expert dataset. During the forward diffusion process, Gaussian noise is gradually added to this sequence of actions:

$$a^t_{k:k+J} = \sqrt{\overline{\alpha}_t} a^0_{k:k+J} + \sqrt{1-\overline{\alpha}_t}\,\varepsilon \tag{9}$$

where $t \in \{1, \ldots, T\}$ is the diffusion step, and $\overline{\alpha}_t$ is the cumulative obfuscation coefficient. The goal of the policy is to learn the denoising function ϵ so that it can predict the noise that has been added to the sequence of actions. The diffusion index t is mapped to embedding through the time embedding function:

$$e_t = TE(t) \tag{10}$$

This embedding allows the model to know the noise level and adjust the noise reduction intensity accordingly. The Mamba mechanism is applied at each step i, and the hidden state is updated as follows:

$$\begin{cases} h_i = A(x_i)\, h_{i-1} + B(x_i)\, x_i \\ y_i = C(x_i)\, h_i \end{cases} \tag{11}$$

where x is the input token, h is the hidden state of the garbled signals on inputs A, B, and C. The model is trained using the standard diffusion loss function:

$$L_{DDP}(\theta) = E_{t,a_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(s_{k-K:k}, a^t_{k:k+J}, t\right)\right\|^2_2\right] \tag{12}$$

Through the training process, it is possible to accurately identify the noise added to the long-term action information chain.

3.4 Unet Decoder

The Unet decoder architecture is designed as a dedicated image decoder, tasked with reconstructing detailed segment masks from abstract features provided by Mamba. This process begins at the "bottleneck" where a global feature vector containing contextual information about the entire scene is projected and reshaped into a small spatial feature map (7×7 with 512 channels). From here, the decoder begins a series of "upsampling" steps to progressively increase resolution. On the first layer, a powerful Unet Decoder Block performs the crucial upsampling step from 7×7 to 14×14. The unique aspect of this block is that it not only upsamples the feature map from the bottleneck but also integrates detailed spatial information from "skip connections", which are features patched directly from ViT. This combination allows the model to maintain global understanding while incorporating local details about obstacle locations. Following the initial U-Net block, a series of four consecutive ConvTranspose2d layers further doubles the feature map size at each step ($14 \times 14 \rightarrow 28 \times 28 \rightarrow 56 \times 56 \rightarrow 112 \times 112 \rightarrow 224 \times 224$), while gradually reducing the number of feature channels. Finally, a 1×1 convolution layer (final_conv) is used to condense the last 16 feature channels into three channels, creating a final prediction mask of the same size as the input image. Based on this model, we need to predict the USV position, the Item position, and the Goal position, so the UNet decoder will have 3 channels, each representing a binary segmentation mask for a specific semantic class: the USV, the target item, and the navigation goal. Each channel is supervised independently using its corresponding ground-truth mask. The predicted segmentation masks are encoded as small navigation features. Geometric and spatial signals such as space regions, obstacle boundaries, and relative target positions are retrieved through this encoding step. The motion commands of the USV are generated by the control module, which integrates the encoded features.

4 Results and Discussion

Firstly, the authors collected and constructed an expert dataset within the simulator. The environmental scenarios were varied, each paired with corresponding state-action data. The dataset's parameters and details can be found at: https://github.com/buinghia3101/USV_dataset.git. The scenario features the USV navigating the environment under overhead observation from two drones (Fig. 4). In reality, drones are simply used as observation devices and to provide input data for the navigation process. For cases of diverse navigation systems, they can be completely replaced by fixed environmental monitoring stations or image data from ships or nearby moving objects. Relying on synchronized multi-view image inputs allows for the exploitation of additional sensors and communication requirements.

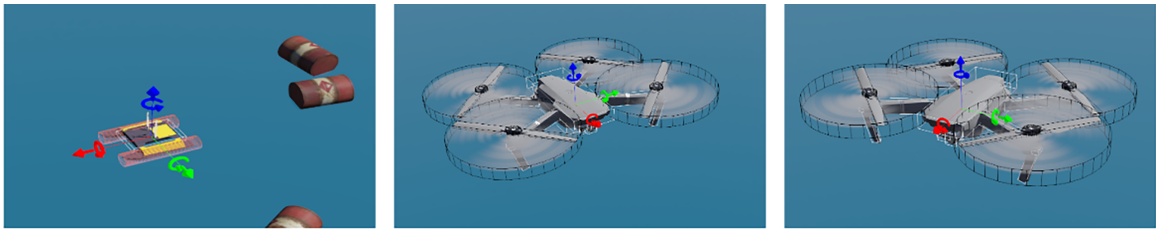

Figure 4: Expert dataset collection environment in a Webots simulation environment with (**a**) USV and (**b**) the first observation drone, and (**c**) the second observation drone, respectively.

Secondly, the training process of the proposed model was carefully monitored, with all parameter variations thoroughly documented. The training process for assessment was conducted on 120 scenarios. Each training sample collected includes the USV position, environmental characteristics, and the trajectory of actions recorded continuously, along with the action parameters of each motor. Special attention was paid to the convergence behavior and the corresponding convergence criteria. The expert dataset was divided into three subsets of training, testing, and validation, allocated in proportions of 60%, 20%, and 20%, respectively. The training process was conducted on a computer with Ubuntu 22.04 operating system, 64 GB RAM, Intel Core i9 11900K CPU, and RTX 3090 GPU. Model performance was evaluated using metrics such as loss, precision, recall, and Intersection over Union (IoU). The IoU metric was calculated by measuring the overlap between the predicted trajectory and the reference trajectory.

Fig. 5 illustrates the training parameters alongside the corresponding results. Although trajectories are represented as sequences of points, we evaluate trajectory similarity using the Intersection over Union (IoU) metric by rasterizing both the predicted and ground-truth trajectories into binary occupancy maps. Each trajectory is converted into a region by buffering the polyline with a fixed width. IoU is then computed as the overlap ratio between the two rasterized regions. The plot comparing training and validation loss reveals a stable learning process, with the model gradually converging throughout training. In all configurations, the training loss decreases rapidly during the initial epochs (around 10–20), indicating effective learning of fundamental problem features. After this phase, the rate of loss reduction slows and approaches a plateau, reflecting the model's progressive refinement in capturing more complex data patterns. The validation loss generally follows the training loss trend but shows greater fluctuations, especially in the early stages, likely due to the heterogeneity of the validation data and challenging environmental conditions. Importantly, the gap between training and validation losses does not widen significantly in later stages, suggesting effective control of overfitting and preservation of the model's generalization ability. The precision and recall metrics show a steady improvement in predictive quality over the course of training. Initially, both metrics are low and highly variable, particularly recall, which indicates the model's limited capacity to fully

recognize attributes derived from expert data. As training continues, these metrics improve consistently, converging within the range of 0.6 to 0.75. This reflects not only a reduction in average error but also a better balance between accuracy and coverage, which is crucial in navigation and decision-making contexts where omissions or incorrect predictions can lead to task failure. Analysis of the intersection over union (IoU) histogram as a function of input sequence length highlights the significant impact of sequence length on model performance. The configuration using a sequence length of 40 achieves the highest and most stable IoU values, approximately 0.640 to 0.757 during the final training phase. This underscores the benefit of incorporating long-term contextual information, enabling the model to more effectively capture the spatio-temporal structure of trajectories. In contrast, shorter sequence lengths (10 to 20) produce substantially lower and more volatile IoU scores, indicating that short-term information alone is insufficient for accurately reproducing predicted regions or target trajectories. These results emphasize the critical importance of modeling long-term dependencies within the problem domain. The learning rate diagram illustrates a tuning strategy that combines an initial warm-up phase with a subsequent linear decay. Gradually increasing the learning rate at the start helps avoid suboptimal solutions and promotes stable learning, while the later decrease allows fine-tuning of parameters and improved convergence. This approach effectively reduces oscillations in loss during the later training stages and enhances overall performance, as reflected in improvements across loss and IoU metrics. In summary, the experimental results demonstrate a stable training process and provide compelling visual evidence that architectural design choices, input sequence length, and optimization strategies collectively contribute to performance improvements. These findings align with prior quantitative evaluations and validate the effectiveness and broad applicability of the proposed method.

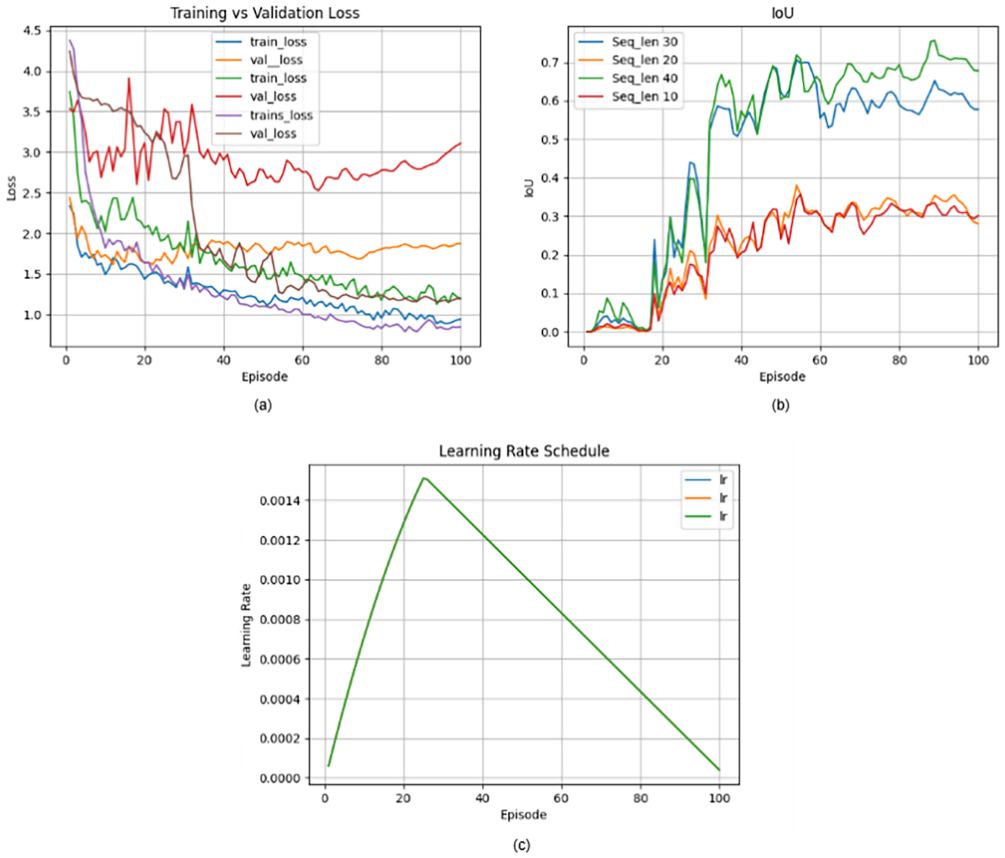

Figure 5: Training parameters of the proposed method on three datasets: train, validation, and test, respectively.

Next, the authors conducted experiments to evaluate the effectiveness of the proposed Mamba architecture in capturing and representing the latent space. Simultaneously, alternative backbone architectures, including Transformer [29], ResNet, and DenseNet were employed for comparative analysis. The primary goal was to assess the efficiency and advantages of integrating the Mamba and DDP architectures in shaping the attributes encoded within the latent space. The comparative results are presented in Fig. 6. Then, to visualize the latent space structure learned from state-action sequences during navigation, t-SNE diagrams were used. The findings reveal a clear distinction between the proposed method and the comparator models in how they organize and preserve the geometric structure of data within the feature space. Specifically, the proposed approach arranges data points in latent space into highly consistent, coherent, and continuous geometric formations. The latent representations maintain the original trajectory after encoding and training, indicating that the model has acquired a representation that is both informative and sequential, effectively preserving geometric and kinematic relationships between successive states throughout motion. In contrast, the transformer model, while capturing the general trajectory trend, exhibits discrete clustering of points within the latent space. Trajectory disruptions and mixing occasionally occur, particularly in complex experimental datasets. Although the self-attention mechanism facilitates the capture of long-range dependencies, its limitations in encoding continuous information result in latent representations that lack temporal smoothness and consistency. Regarding ResNet, the distribution of latent points is more dispersed, and trajectories fail to retain linear information effectively. The data points do not form continuous curves or shapes corresponding to the original trajectory, suggesting a significant degradation of temporal progression and dynamic inter-state relationships. This outcome highlights the limitations of pure convolutional neural networks when encoding extended state sequences, especially in navigation tasks that demand temporal continuity. For DenseNet, latent points cluster almost randomly within a confined spatial region, making it difficult to identify trajectory shape or temporal order. This lack of discernible structure implies challenges in encoding the dynamics and sequential relationships inherent in the data, resulting in latent representations with low discriminability in the context of navigation based on continuous state sequences. Overall, a visual examination of the t-SNE plots in Fig. 6a–c demonstrates the proposed method's effectiveness in learning structured and stable latent spaces, while preserving the geometric and kinematic properties of the trajectory. This observation helps explain the model's improved performance in quantitative evaluations. Special orbital properties such as figure-eight shapes, right angles, and arcs from the proposed framework are extracted and are highly stable, with little disturbance to the noisy representations in the weight matrix. The evaluations in the hypothetical cases of the corresponding trajectories in the latent space further reinforce effective accessibility. Moreover, the method's capacity to incorporate the continuous kinematic constraints of the USV within the state-space model framework highlights its suitability for the task.

Table 1 presents the comparison between the proposed method and the state-of-the-art method. Firstly, to consider the success rate (SR), methods such as BC (MLP) and BC (LSTM), which rely on simple imitation learning, yielded limited results of 41.3% and 53.7%, respectively. This indicates their inefficiency in modeling long-term dependencies and complex environmental scenarios. By employing the Transformer architecture, the chain model becomes a crucial component in problem-solving, boosting the success rate of BC (Transformer) to 61.5%, which is noteworthy. Further improvements in SR are observed with methods like GAIL, PPO (LSTM), and Decision Transformer, with DDP (Transformer) achieving a maximum SR of 72.4%. The proposed technique demonstrates a strong performance, attaining a success rate of 71.1%, only 1.3% lower than DDP, indicating its efficiency is nearly on par with the largest existing Transformer model, though not significantly better. It is important to note that SR alone does not accurately represent orbital quality, necessitating a more comprehensive analysis using spatial error indices. Regarding ADE, the proposed method achieves 1.09 m, and for FDE, 2.03 m, delivering the best results across the entire comparison set.

BC (Transformer) showed a reduction of approximately 34.7% in ADE, from 1.67 to 1.09 m, while FDE decreased by 32.6%, from 3.01 to 2.03 m. Although DDP (Transformer) had the highest orbital accuracy in terms of SR, it was significantly less accurate in spatial error metrics, with ADE and FDE values of 1.18 and 2.21 m, respectively. Furthermore, MLP and LSTM models exhibited low latency but delivered poor inference performance. In contrast, large Transformer models such as Decision Transformer and DDP have latencies of 22.4 and 31.7 ms, respectively, which are too high for real-time applications. The proposed approach achieves a latency of 9.6 ms, approximately 57% lower than Decision Transformer and about 70% lower than DDP while maintaining comparable or efficient performance. This architecture successfully integrates the sequence model without significantly increasing computational complexity. DDP (Transformer) requires 42.5 million parameters, whereas the proposed method uses only 14.8 million, representing a reduction of over 65%. Although Decision Transformer, with 21.3 million parameters, is smaller than the proposed model, it still underperforms in ADE/FDE metrics. Although DDP (Transformer) achieves the highest SR, it exhibits significantly higher inference latency and model size. In contrast, the proposed method achieves comparable SR while delivering efficient trajectory accuracy (ADE/FDE) and real-time performance, making it more suitable for deployment on resource-constrained platforms. This demonstrates the efficiency of the architectural design, enabling the model to learn rich feature representations without excessive scaling. In summary, experimental results indicate that the proposed method strikes an excellent balance between success rate, trajectory accuracy, and computational efficiency. It not only meets the highest success rate standards among Transformer-based methods but also excels in ADE/FDE with low latency and a moderate parameter count. These qualities make the proposed method especially well-suited for real-time automated operation on hardware with limited resources.

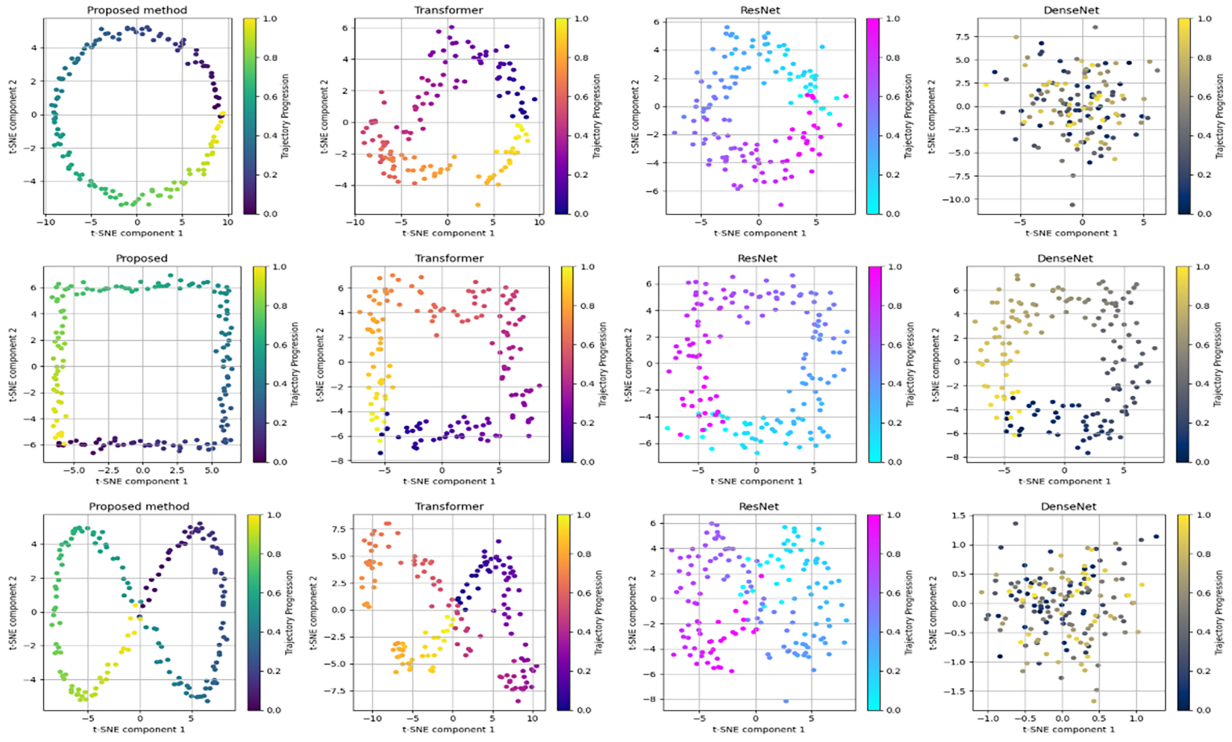

Figure 6: Training parameters of the proposed method on three datasets: train, validation, and test, in the comparison between the proposed Mamba with Transformer [29], ResNet and DeseNet architectures in capturing and representing the latent space of (**a**) circle trajectory, (**b**) square trajectory and (**c**) 8-number shaped trajectory, respectively.

Table 1: Comparing the effectiveness of motion planning with modern methods.

Method	Type	SR ↑ (%)	ADE ↓ (m)	FDE ↓ (m)	Latency ↓ (ms)	Params (M)
BC (MLP) [30,31]	IL	41.3	2.84	4.91	1.2	0.12
BC (LSTM) [32]	IL	53.7	2.01	3.62	4.8	0.54
BC (Transformer) [33,34]	IL	61.5	1.67	3.01	18.6	7.9
GAIL [9,22]	IL + RL	58.2	1.79	3.28	6.2	1.8
Decision Transformer [35]	IL	65.9	1.52	2.87	22.4	21.3
PPO (MLP) [36]	RL	55.4	1.95	3.41	2.3	1.4
PPO (LSTM) [32]	RL	63.8	1.61	2.94	7.1	3.6
A2C [37]	RL	51.6	2.14	3.89	2.0	1.2
DDP (Transformer) [34]	IL	72.4	1.18	2.21	31.7	42.5
Proposed Method	IL	71.1	1.09	2.03	9.6	14.8

The authors compared navigation simulations in unknown environments using the proposed method and common navigation algorithms (Fig. 7). Two scenarios were created: a static unknown environment and an unknown environment with high noise. The noise was modeled as dynamic obstacles—specifically, other USVs moving randomly—that needed to be avoided. The evaluation metric used was the lateral deviation error, which measures the difference between the executed trajectory and the optimal trajectory determined when the environmental information is fully known. This lateral deviation error is calculated using Eq. (13):

$$e\perp(t) = min_s \left\| p(t) - pref(s) \right\| \perp \tag{13}$$

where $p(t)$ is the actual position of the vehicle, $pref(s)$ is the reference trajectory, and $\| . \| \perp$ is the distance perpendicular to the tangent of the trajectory. With the set of lateral errors during the measurement process $\{e\perp(t1), e\perp(t2), \ldots, e\perp(t_N)\}$. The median point is determined by Eq. (14):

$$Median(e\perp) = \begin{cases} e\perp,(N+1)/2 & N\ is\ an\ even\ number. \\ \dfrac{e\perp,(N/2) + e\perp,(N/2+1),}{2} & N\ is\ an\ odd\ number. \end{cases} \tag{14}$$

In both environmental conditions, the proposed method demonstrated the best and most stable performance. In Scene 1, the median lateral error was approximately 0.03 m, significantly lower than that of traditional and hybrid methods (Fig. 7a). The narrow interquartile range (IQR) and absence of outliers indicate that the method can reliably track its trajectory with a high degree of accuracy, even in environments with partial or no obstructions. In Scene 2, characterized by high uncertainty and noise, the proposed method showed only a slight increase in error, whereas other methods experienced a sharp decline in performance, underscoring their generalizability and resilience to environmental changes. In both scenarios, the A* algorithm [4] exhibited a higher mean and variance of lateral error in lateral error. Its primary function on discrete maps is global planning, which bypasses the dynamic constraints of USV and environmental noise. Scene 2 represents a transitional phase, during which the error distribution of A* increased significantly, reflecting instability in an unknown and highly dynamic environment. Both RRT*-DWA [38] and BIT*-TEB methods [39] outperformed the A* algorithm, as they integrate planning with local trajectory optimization to achieve better results. By continuously optimizing the trajectory, BIT*-TEB achieved a narrower error distribution and lower average error than RRT*-DWA. However, both methods exhibited numerous outliers in Scene 2, especially under high noise levels and unstable obstacle configurations, highlighting limitations in modeling the long-term dependencies of system states (Fig. 7b). Hybrid methods in Scene 1 matched the

average error of traditional reinforcement learning (RL) [5], but their large variance and skewed distributions made stability during inference difficult to predict. Domain changes in Scene 2 significantly impacted RL, leading to increased errors and frequent local failures. Incorporating global planning information into the RRT*-RL method [5] greatly improved its efficiency compared to traditional RL by reducing median error and outliers; however, its performance remained highly dependent on sampling quality and computational cost. Overall, the results demonstrate that the proposed method achieves an excellent balance between accuracy and stability. It is more efficient than traditional planning algorithms, minimizing lateral error and oscillations. Its error distribution is more consistent and narrower than those of RL-based or hybrid methods, even in unexplored environments. Employing a Mamba-based chain model to capture long-term dependencies and hidden dynamics of the entire USV could further enhance navigation efficiency in complex real-world scenarios.

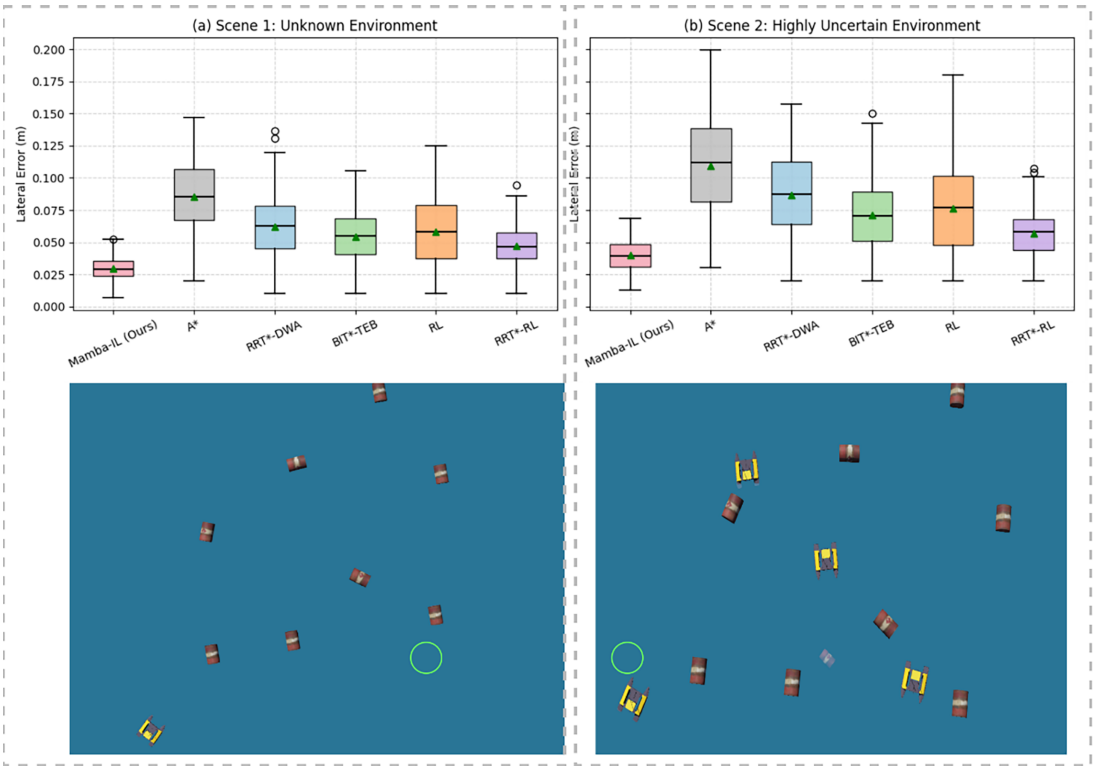

Figure 7: The comparison of the proposed method with classical directional programming algorithms based on median lateral deviation with respect to the optimal trajectory in (**a**) the unknown environment and (**b**) the highly uncertain environment.

To reinforce its validity and ensure its effective practical implementation, the authors collected experimental cases with the USV sample in both simulated (Fig. 8) and real-world environments (Figs. 9–11). The multi-view settings and the actual USV model are provided in Fig. 9a,b, respectively. The experiment to navigate USV on a coastal patrol is presented in Figs. 10 and 11. At each inference, the model generates a short sequence of 30 consecutive actions $a_t = [v_t, \omega_t]$. The parameters of interest include the speed and angular velocity corresponding to the motor. According to the receding-horizon strategy, the initial actions are performed within the physical limitations of the USV. The process is illustrated by the Eq. (15):

$$v_t \in [v_{min}, v_{max}], \quad \omega t \in [-\omega_{max}, \omega_{max}] \tag{15}$$

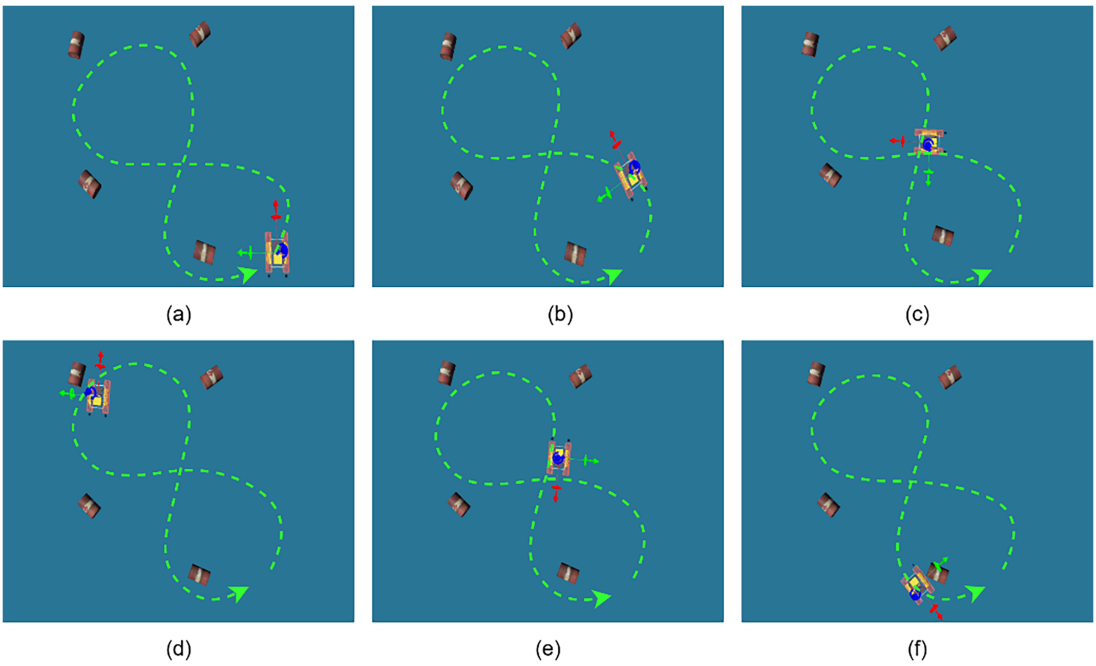

Figure 8: Simulate USV motion planning on ROS2-Webots environment. The figures are recorded in chronological order (**a**–**f**), respectively.

Figure 9: Set up a verification experiment to navigate the USV in a real environment. (**a**) shoreline multi-view setup, (**b**) USV, respectively.

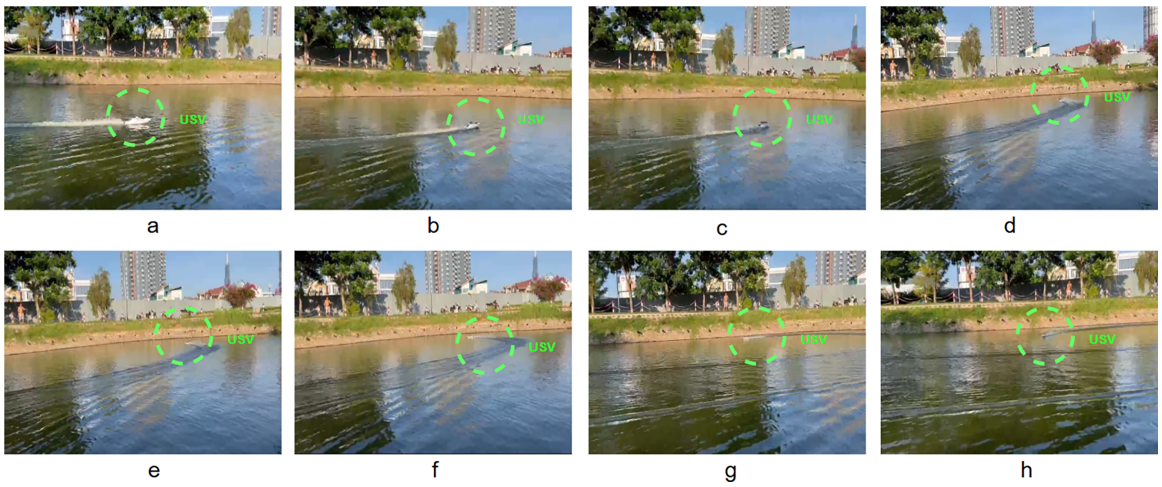

Figure 10: Results of coastal patrol experiments, in the medium environment with (**a**) start point, (**b**) turn left towards the shore along the IL-based trajectory, (**c**) continue turning left towards the shore, (**d**) turning point for shore patrol, (**e**) starting point of the orbit close to the shore, (**f**) coastal patrol, (**g**) the turning point at the end of the patrol process, (**h**) the exit turn marks the end of the test run, preparing to move to a new orbit, respectively.

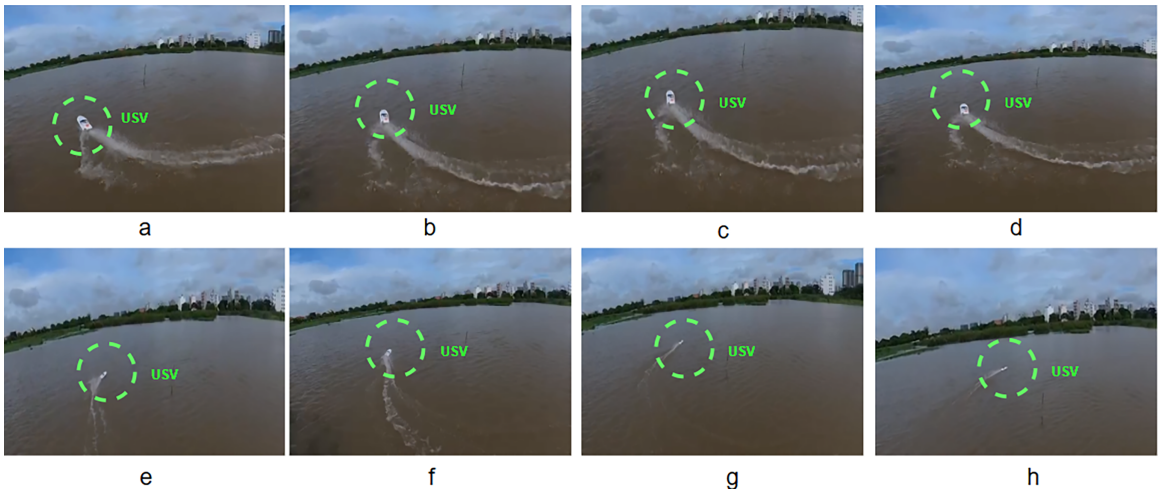

Figure 11: Results of coastal patrol experiments, in the large environment with (**a**) start point, (**b**) turn left towards the shore along the IL-based trajectory, (**c**) continue turning left towards the shore, (**d**) turning point for shore patrol, (**e**) starting point of the orbit close to the shore, (**f**) coastal patrol, (**g**) the turning point at the end of the patrol process, (**h**) the exit turn marks the end of the test run, preparing to move to a new orbit, respectively.

To smooth the USV's trajectory, the smoothed function $\tilde{a}_t$ is designed to reduce high-frequency oscillations:

$$\tilde{a}_t = \alpha a_t + (1-\alpha)\,\tilde{a}_{t-1} \tag{16}$$

where a_t is the raw control command generated by the IL model at time t, and $\tilde{a}_t$ is the smoothed control command. α is the smoothing coefficient. The resulting command is tracked by a low-level controller that computes the surge force F_t and yaw moment τ_t, which are allocated to the left and right thrusters by Eq. (17):

$$\begin{bmatrix} T_L \\ T_R \end{bmatrix} = \begin{bmatrix} 1 & -d \\ 1 & d \end{bmatrix} \begin{bmatrix} F_t \\ \tau_t \end{bmatrix} \tag{17}$$

where T is the thrust and d is the length. The estimated trajectory was effectively completed throughout most of the trials. The USV's activities in a real-world environment yielded positive results, with no collisions with obstacles, and it consistently maintained a set distance during coastal patrols. Phenomena such as loss of orbit or sudden deviations did not occur throughout the experiment. The authors hope these results will provide opportunities for deployment in port monitoring, the marine environment, and other tasks, using flexible navigation systems like USVs.

The experimental process was conducted on a USV prototype designed by the authors. The observation cameras used are Intel RealSense D435. The embedded computer Jetson Nano is equipped, along with an accompanying ESP32 microprocessor. The signal is continuously communicated between the coastal observation system and the Intel Core i7 6920HQ processor, Quadro M2000M GPU. The A2212 brushless motor and the 30A ESC control unit were installed as navigation devices in the experiment. The snapshot sequence from Fig. 10a–h shows the USV's trajectory based on the established mimicry learning. The trajectory ensures a figure-eight path with cornering to keep the USV close to the shore, performing exploration and data collection. At the turning point (Fig. 10g), the USV changes direction and moves diagonally to prepare for the second mission trajectory in the established figure-eight pattern. Experiments demonstrate the effectiveness of this approach, as in real-world environments, data collection and vision-perception system training for USVs require extensive experimental data and lengthy, complex training time. However, using the proposed IL structure, the USV's navigation trajectory has been designed and ensured to allow the USV to navigate in medium-sized environments.

The author continues to experiment and implement a practical navigation problem for the USV in a wider water environment, as shown in Fig. 11. The experimental USV demonstrates stable movement and ensures the morphological form of path planning based on the proposed method. Furthermore, with supporting mechanisms for noise reduction and vibration damping, the movement is smooth, and the speed is stable while following the designed trajectory.

5 Conclusions

The paper proposes an imitation learning model for the USV motion planning problem in an environment. The basic framework is built upon encoding multi-view observation information through a latent space model. This process is handled by the ViT network. The contexts are compressed and pushed to the imitation learning model to capture continuous action sequences from the previously collected expert dataset. The state-space model based on the Mamba architecture is used in conjunction with the Denoising Diffusion Policies mechanism, enabling the model to achieve high generality with continuous action sequences. The planning of information from all sources into the same latent spatial environment to bring the characteristics in the simulated environment closer to reality. Qualitative and quantitative tests were conducted to demonstrate the feasibility and competitiveness of the method. The IoU score reached the maximum threshold of 0.757 with a sequence length of 40. Meanwhile, comparative tests with deep learning methods are consistently at competitive parameter levels, both in terms of effectiveness and speed. Benchmark tests with traditional path-planning algorithms in unknown environments yielded efficient results. The number of parameters used is 14.8 million, which is much smaller with a high-efficiency model like the Transformer, while the efficiency is clearly competitive. The visualizations illustrating the preservation of information in the expert dataset using t-SNE for evaluation demonstrate flexibility with

action-state data specific to USVs. Comparisons with conventional algorithms applied in information-scarce environments further strengthen the competitiveness of deployment on navigation targets. Real-world testing with USVs shows promise. The proposed method demonstrates potential in the motion planning problem with a USV as the object. Future developments will include applying a hierarchical world model to optimize the agent's overall understanding of the system's characteristics. In addition, establishing links and creating close relationships between environmental monitoring points is a worthwhile area of study.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by the Ministry of Education and Training of Vietnam under the research project B2024-BKA-13.

Author Contributions: Conceptualization, Manh-Tuan Ha, Thai-Viet Dang; Methodology, Nhu-Nghia Bui, Manh-Tuan Ha, Thai-Viet Dang; Software, Nhu-Nghia Bui, Dinh-Quy Vu; Formal analysis, Manh-Tuan Ha; Investigation, Nhu-Nghia Bui, Dinh-Quy Vu; Resources, Nhu-Nghia Bui, Thai-Viet Dang; Data curation, Nhu-Nghia Bui; Writing—original draft, Dinh-Quy Vu, Thai-Viet Dang; Writing—review & editing, Dinh-Quy Vu, Thai-Viet Dang; Visualization, Dinh-Quy Vu; Funding, Manh-Tuan Ha; Supervision, Thai-Viet Dang; Project administration, Thai-Viet Dang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available on request from the corresponding author, Thai-Viet Dang.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang Y, Liu X, Zhou W, Wen G, Xu X. Global-local hybrid path planning and path point tracking control for unmanned surface vehicles. IEEE/ASME Trans Mechatron. 2025;1–12. doi:10.1109/TMECH.2025.3618682.
2. Li C, Yao L, Mi C. Fusion algorithm based on improved A* and DWA for USV path planning. J Mar Sci Appl. 2025;24(1):224–37. doi:10.1007/s11804-024-00434-1.
3. Wang J, Xiang S, Shen T, Fang Z, Niu S, Pan X, et al. Imitation learning from observation for ROV path tracking. Intell Mar Technol Syst. 2025;3(1):20. doi:10.1007/s44295-025-00069-0.
4. Dang TV, Tan PX. Hybrid mobile robot path planning using safe JBS-A*B algorithm and improved DWA based on monocular camera. J Intell Rob Syst. 2024;110(4):151. doi:10.1007/s10846-024-02179-z.
5. Pham HL, Bui NN, Dang TV. Hybrid path planning for wheeled mobile robot based on RRT-star algorithm and reinforcement learning method. J Robot Control. 2025;6(4):2045–51.
6. Dirckx D, Bos M, Vandewal B, Vanroye L, Swevers J, Decré W. ASAP-MPC: an asynchronous update scheme for online motion planning with nonlinear model predictive control. Auton Rob. 2025;49(1):8. doi:10.1007/s10514-025-10192-w.
7. Sun Y, Cao Y, Shan Q, Zhang H. Event-triggeredbased multi-objective model predictive control for ship path following with velocity assignment. J Mar Sci Technol. 2025;30(2):364–76. doi:10.1007/s00773-025-01055-1.
8. Li S, Li C. Path planning for unmanned surface vehicles in dynamic environments based on artificial potential field and global guided reinforcement learning. J Mar Sci Appl. 2025;1–12. doi:10.1007/s11804-025-00697-2.
9. Chen T, Zhang Z, Fang Z, Jiang D, Li G. Imitation learning from imperfect demonstrations for AUV path tracking and obstacle avoidance. Ocean Eng. 2024;298:117287. doi:10.1016/j.oceaneng.2024.117287.
10. Vasconcellos EC, Negreiros ÁPF, de Araújo APD, Guerra R, Preux P, dos Santos DH, et al. Yara: an ocean virtual environment for research and development of autonomous sailing robots and other unmanned surface vessels. J Intell Rob Syst. 2025;111(3):78. doi:10.1007/s10846-024-02212-1.

11. Dang TV, Tran DMC, Bui NN, Tan PX. ELDE-net: efficient light-weight depth estimation network for deep reinforcement learning-based mobile robot path planning. Comput Mater Contin. 2025;85(2):2651–80. doi:10.32604/cmc.2025.067500.
12. Cong S, Zhou Y. A review of convolutional neural network architectures and their optimizations. Artif Intell Rev. 2023;56(3):1905–69. doi:10.1007/s10462-022-10213-5.
13. Boretti A. Navigating the future: the expanding role of unmanned surface vehicles in maritime security. J Transp Secur. 2025;18(1):24. doi:10.1007/s12198-025-00314-x.
14. Peng JW, Hu MC, Chu WT. An imitation learning framework for generating multi-modal trajectories from unstructured demonstrations. Neurocomputing. 2022;500(4):712–23. doi:10.1016/j.neucom.2022.05.076.
15. Jbene M, Chehri A, Saadane R, Tigani S, Jeon G. From RNNs to transformers and beyond: a deep dive into intent detection in goal-oriented conversational agents. Cogn Comput. 2025;17(3):116. doi:10.1007/s12559-025-10470-w.
16. Wang Y, Jin J, Chen X, Wu Z, Zhang L. A lightweight crack segmentation network based on the importance-enhanced Mamba model. Sci Rep. 2025;15(1):41549. doi:10.1038/s41598-025-25504-4.
17. Sandooghdar A, Yaghmaee F. Enhancing image restoration: parameter-assisted and edge-based inpainting with U-Net mamba. Signal Image Video Process. 2025;19(6):466. doi:10.1007/s11760-025-04045-3.
18. Zhu W, Ma Y, He L, Zhang B. MambaSkill: mamba-inspired robotic skill abstraction and dual-level generation for long-horizon control. In: Advanced intelligent computing technology and applications. Singapore: Springer Nature; 2025. p. 77–88.
19. Xia J, Wu X, Xu M. BEV-fused imitation and reinforcement learning for autonomous driving planning. J Shanghai Jiaotong Univ Sci. 2026;31(1):154–66. doi:10.1007/s12204-025-2851-3.
20. Antonelo EA, Couto GCK, Möller C, Fernandes PH. Investigating behavior cloning from few demonstrations for autonomous driving based on bird's-eye view in simulated cities. In: Intelligent systems. Cham, Switzerland: Springer Nature; 2025. p. 155–68.
21. Nguyen VT, Duong DN, Dang TV. Optimal two-wheeled self-balancing mobile robot strategy of navigation using adaptive fuzzy controller-based KD-SegNet. Intell Serv Robot. 2025;18(3):661–85. doi:10.1007/s11370-025-00606-0.
22. Tan J, Chen G, Huang Z, Liu H, H.Ang M Jr. E-GAIL: efficient GAIL through including negative corruption and long-term rewards for robotic manipulations. Appl Intell. 2025;55(7):633. doi:10.1007/s10489-025-06335-2.
23. Chen N, van derSmagt P, Cseke B. Fast preserving local distances and topology in auto-encoders. In: Neural information processing. Singapore: Springer Nature; 2025. p. 415–30.
24. Chen W, Lin X, Zhou Z, Qiao Y, Li P, Yu D. GAPPO-ERW: an imitation reinforcement learning approach for AGV path planning. J Supercomput. 2025;81(8):883. doi:10.1007/s11227-025-07361-0.
25. Luo S, Schomaker L. Reinforcement learning in robotic motion planning by combined experience-based planning and self-imitation learning. Robot Auton Syst. 2023;170(4):104545. doi:10.1016/j.robot.2023.104545.
26. Dang TV, Bui NN, Tan PX. Imitation learning-based mobile robot navigation using semantic segmentation and depth estimation. In: Proceedings of the 2025 24th International Symposium on Communications and Information Technologies (ISCIT); 2025 Oct 16–18; Hanoi, Vietnam. p. 403–8.
27. Bui TL, Nguyen VT, Bui NN, Dang TV. Four-wheeled omnidirectional mobile robots strategy of navigation using hybrid HPSO-GA-PID controller-based YOLO3D-CSSA network. Intell Serv Robot. 2025;19(1):3. doi:10.1007/s11370-025-00669-z.
28. Hafner D, Lillicrap T, Fischer I, Villegas R, Ha D, Lee H, et al. Learning latent dynamics for planning from pixels. arXiv:1811.04551. 2018.
29. Zhang Y. Transformer-based adversarial imitation learning: enhancing long-term decision making with time series. In: Proceedings of the 3rd International Conference on Machine Learning, Cloud Computing and Intelligent Mining (MLCCIM2024). Singapore: Springer Nature; 2025. p. 84–92.
30. Pomerleau DA. Efficient training of artificial neural networks for autonomous navigation. Neural Comput. 1991;3(1):88–97. doi:10.1162/neco.1991.3.1.88.
31. Sumanth U, Punn NS, Sonbhadra SK, Agarwal S. Enhanced behavioral cloning-based self-driving car using transfer learning. In: Data management, analytics and innovation. Singapore: Springer; 2021. p. 185–98.

32. Heess N, Hunt JJ, Lillicrap TP, Silver D. Memory-based control with recurrent neural networks. arXiv:1512.04455. 2015.
33. Mineault P. Is attention all you need? In: From human attention to computational attention. Cham, Switzerland: Springer Nature; 2025. p. 297–314.
34. Pelluri N. Transformers for image-goal navigation. arXiv:2405.14128. 2024.
35. Chen L, Lu K, Rajeswaran A, Lee K, Grover A, Laskin M, et al. Decision transformer: reinforcement learning via sequence modeling. arXiv:2106.01345. 2021.
36. Li C, Zhang Z, Wang J, Yuan S, Wang Z, Chai L, et al. Improved multi-agent proximal policy optimization algorithm for resource allocation with radar-perception in UAV-assisted VEC. ACM Trans Sen Netw. 2025;21(6):1–28. doi:10.1145/3771094.
37. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, et al. Asynchronous methods for deep reinforcement learning. arXiv:1602.01783. 2016.
38. Su Y, Xin J, Sun C. Dynamic path planning for mobile robots based on improved RRT* and DWA algorithms. IEEE Trans Ind Electron. 2025;72(10):10595–604. doi:10.1109/TIE.2025.3546349.
39. Xie Y, Ma Y, Cheng Y, Li Z, Liu X. BIT*+TD3 hybrid algorithm for energy-efficient path planning of unmanned surface vehicles in complex inland waterways. Appl Sci. 2025;15(7):3446. doi:10.3390/app15073446.