ShipNaviSim: Data-Driven Simulation for Real-World Maritime Navigation

Quang Anh Pham
Singapore Management University
Singapore
qapham@smu.edu.sg

Janaka Chathuranga Brahmanage
Singapore Management University
Singapore
janakat.2022@phdcs.smu.edu.sg

Akshat Kumar
Singapore Management University
Singapore
akshatkumar@smu.edu.sg

ABSTRACT

Maritime traffic management in busy ports faces growing challenges due to increased vessel traffic and complex waterway interactions. Strategies such as e-navigation by the International Maritime Organization aim to enhance navigation safety through traffic digitization. Maritime traffic simulation is essential for these systems, offering a virtual environment to model, analyze, and optimize traffic flows. Unlike road traffic, there are few simulators for maritime traffic, and they often lack realism and multi-ship interactions. In this paper, we (a) present ShipNaviSim, a data-driven maritime traffic simulator that utilizes a large-scale dataset over 2 years and electronic navigation charts to model vessel movements in Singapore Strait, one of the busiest ports in the world; (b) implement and evaluate different imitation learning algorithms such as behavior cloning to learn a policy that can accurately simulate real world vessel movements and multi-ship interactions; (c) develop vessel-specific metrics such as trajectory curvature, near miss rate, to validate the learned policy's alignment with human expert data. Extensive testing shows that our learned agents can behave like human experts, and thus can be used with the simulator for recommending routes for vessels in a hotspot region or generating diverse traffic scenarios to benchmark navigation systems.

KEYWORDS

Maritime Traffic Simulation; Imitation Learning; Reinforcement Learning

ACM Reference Format:
Quang Anh Pham, Janaka Chathuranga Brahmanage, and Akshat Kumar. 2025. ShipNaviSim: Data-Driven Simulation for Real-World Maritime Navigation. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Maritime transportation plays a crucial role in global trade and economics, with over 80% of world trade volume carried by sea [37]. Therefore, how we manage the maritime traffic is very important for operating global supply chains smoothly. This problem is challenging especially in important ocean thoroughfares. For example, Singapore strait (one of the busiest waterway in the world) is becoming more congested due to increasing number of arriving vessels over years [24, 27]. This leads to risks of collisions, which threaten human lives, and raise environmental issues such as oil spills [2].

To address vessel coordination for increasing navigation safety, several strategies have been proposed recently [8, 23, 32]. A key feature in such previous works is that they model *macro-level* vessel coordination where the maritime traffic separation scheme (analogous to sea highways) is divided into multiple large zones (geo-fenced sea space). Agents (vessels) only receive high-level guidance regarding when to arrive in each zone and which zone to go to next. Such macro simulation can simulate the traffic over longer duration (~5-10 hours) [6]. Our proposed work, in contrast, aims to address *micro-level* simulation of the maritime traffic over short time periods (10-20 minutes). This micro-level simulation is particularly important for port watch operators managing a port's vessel traffic information system (VTIS) for near term risks (10-20 minutes) [28]. There are several challenges in such micro-level simulation such as capturing real vessel movement patterns (e.g., vessels cannot turn sharply, speedup or slowdown quickly, or stop in the water completely unlike land vehicles). Furthermore, multi-vessel interactions capturing how other vessels in the nearby area affect a vessel's movement are critical to model and simulate. We next describe major control structures in maritime traffic management.

Vessel traffic information system. Most busy ports such as Singapore's, Tokyo bay, model ships movements using the *traffic separation scheme* (TSS) which are one-way sea lanes. These routes are created for managing traffic and minimizing near misses when vessels enter and exit the strait. Fig 1 illustrates TSS of Singapore strait using its electronic navigation chart. The TSS is further divided in to a set of zones, which are geo-fenced parts of the TSS (shown as different polygons in figure 1).

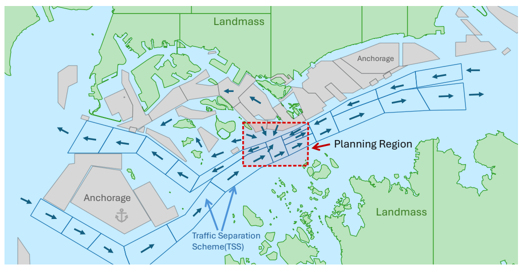

Figure 1: Traffic separation scheme (TSS) of Singapore strait. Red region is a *hotspot* region with high cross traffic.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

A port's vessel traffic information system (VTIS) that is manned by port watch operators [28]. Operators continuously monitor vessel traffic around the clock using radars and other sensors, taking action if a risky navigation situation is anticipated to develop in the near future (e.g., within the next 10-20 minutes). Certain areas within the TSS, like the red rectangle in Figure 1, are more prone to frequent hotspots as vessels navigate between the port area (pink zones) and TSS zones (blue zones). This crossing of vessels often leads to hazardous situations and numerous near misses. Simulating traffic accurately in this high cross-traffic region with multi-vessel interactions is the focus of our work.

Our work learns imitation learning based policies for vessels to accurately simulate the movement patterns found in real historical data in the planning region. Such policies can be used in several safety-focused settings. They can be used by port watch operators to predict accurately how a vessel (or a group of vessels) in a hotspot would move in the near future, which can help them manage dangerous situations proactively. In a control, our learned policy can be assigned to different vessels, while selected ego vessel(s) are controlled using an RL agent, for example, to mitigate hotspots.

Contributions. Our key contributions are:

- We first collect and process the real-world dataset of vessel movements in a hotspot region of Singapore strait over 2 years to create ShipNaviSim, a data-driven simulator for *micro-level* maritime navigation. ShipNaviSim provides a flexible design for different feature settings such as log-play (all agents moving as per historical data), self-play (some agents controlled using a policy), and a graphical interface.
- We adapt several offline reinforcement learning and imitation learning methods to run on ShipNaviSim. We design relevant state features, action space specific to the maritime setting that enable such algorithms to train policies that align well with human experts (i.e., historical data), capture multi-vessel interactions, and generalize well on the unseen test data.
- We develop several vessel-specific metrics such as trajectory curvature, near miss rate, to quantitatively validate the learned policy's alignment with human expert data.
- Extensive testing shows that our learned agents can behave like human experts, and thus can be used with the simulator for recommending routes for vessels in a hotspot region or simulating diverse traffic scenarios.

2 RELATED WORK

We next review some related works for vessel traffic management. An expert system is introduced in [20] for the Turkish strait including components like vessel traffic flow simulation models, hydrographic prediction model. Popular optimization methods such Mixed Integer Linear Programming (MILP) and Constraint Programming (CP) are also applied for optimizing the navigation efficiency as well as satisfying safety constraints [3, 8]. Seeking the collision-free paths of vessels can be considered as a multi-agent path finding problem which is studied [34]. Reinforcement learning methods [32, 33] are proposed for scaling the number of vessels and dealing with the uncertainty in the environment. Modeling travel time of vessels is presented in [6] using Wasserstein Generative Adversarial Networks [4]. Despite such progress, the problem setting is still limited to the macro-level management ignoring how vessels voyage inside a specific zone capturing historical movement patterns and multi-ship interactions. There have been relatively few works for modeling micro-level trajectory simulation. In [7], a generative model based on LSTM [19] is learned to predict a set of future trajectories for each vessel. However, this approach uses an MILP to be solved at each time step to decide a vessel's future course, which is not scalable when large number of agents are involved in a simulator. In contrast, our method learns a neural net based policy, that once trained, is fast to execute.

Another relevant research direction is building simulators for autonomous vehicles, especially self-driving cars (SCs). According to [13], there are three types of agents that can be inside SC simulators: rule-based agents such as the popular IDM model [36] which reduce the vehicle speed or stops to prevent crashes, *log-play* agents using the recorded human driving logs to replay, and learning-based agents. Latest SCs simulators [17, 22] contain all these types of agents or provide an interface to use rule-based and log-play agents to train learning-based agents. Training methods are also classified into three categories: trajectory prediction [38, 41], open-loop imitation learning (IL) like behavior cloning (BC) [5, 31] without requiring environment interactions, and close-loop approaches such as adversarial imitation learning [9, 10] or reinforcement learning [13, 21].

Such SC simulators cannot be directly applied to the maritime setting due to major differences in the movement patterns of cars and ships. A vessel can not stop completely like cars when facing collision risk; vessels cannot make sharp turns, and other movement restrictions apply such as no sudden (de)acceleration. There are other unique factors, such as course-over-ground (COG), heading (the way a vessel is oriented), which need to be incorporated in the maritime traffic simulation. As these maritime traffic constraints cannot be described analytically easily, our goal is to learn such movement patterns using imitation learning from the data.

In contrast to SCs, there are much fewer studies for developing simulators for surface vehicles (ASVs) such as vessels [40]. Most of them only contain rule-based agents and do not rely on real-world data [26], or do not consider multi-vessel interactions [30]. Some of them consider only small amount of data and the simulation platform is not detailed [18]. Therefore, our ShipNaviSim provides a high-fidelity benchmark for micro-level maritime navigation.

3 SHIPNAVISIM

3.1 Dataset for constructing scenarios

The maritime navigation data for large vessels are recorded in the Automatic Identification System (AIS). We utilize a dataset from the Singapore strait, which includes vessel trajectories with 59.6 million data points (AIS records detailing vessel traffic) over a period of 2 years. The dataset consists of two types of data: static information and trajectory information.

The static information includes the following fields: (a) *Ship ID*: A unique identifier for the ship; (b) *MMSI*: The Maritime Mobile Service Identity number of the vessel; (c) *Length*: The length of the vessel; (d) *Width*: The width of the vessel; (e) *Vessel Type*: Specifies whether the vessel is a cargo ship, oil tanker or any other.

We also have *dynamic* trajectory information for each vessel, which includes the following data fields: (a) *Timestamp*: The timestamp of the record in UTC; (b) *Status*: The current status code of the vessel, indicating whether it is underway using engine, anchored, aground, etc.; (c) *Heading*: The direction the vessel is facing; (d) *Course Over Ground*: The actual direction in which the vessel is moving over the ground; (e) *Speed*: The speed of the vessel in nautical miles per hour; (f) *Latitude*: The latitude of the vessel; (g) *Longitude*: The longitude of the vessel.

Selection of Planning Region. Our dataset covers the entire Singapore Strait, as shown in Figure 1. The green areas represent landmasses, while the gray areas indicate anchorages. Each small polygon represents a sea zone. The Traffic Separation Scheme (TSS) defines the sea lanes within these zones, specifying the directions vessels should follow. The arrows indicate the TSS direction for each zone. In most areas, the lanes are clearly separated, reducing the need for detailed planning. However, the area highlighted by the dotted red rectangle experiences heavy cross-traffic, as it serves as a crossing point within the TSS. Consequently, vessels frequently navigate across traffic separation areas in this region to avoid congestion and hotspots. Therefore, we select this area for our simulator.

Pre-processing. In the pre-processing phase, we filter out outlier records and irrelevant information. We select only vessels that are underway using their engines (i.e., they are not being towed). Additionally, we include only tankers and cargo vessels based on the vessel type field, excluding smaller vessels such as tugboats and fishing boats; tankers and cargos are the riskier class as they are much larger in size (200-300 meters) and have less navigation agility than smaller vessels. Then we transform our dataset into the following form which contain M historical trajectories, $D = \langle \tau_1, \tau_2, \ldots \tau_m, \ldots, \tau_M \rangle$. Each trajectory $\tau_m = \langle l_1^m, l_2^m, \ldots, l_t^m, \ldots, l_T^m \rangle$ contains a sequence of data points, where T is the length of the trajectory. $l_t^m = (x_t^m, y_t^m, v_t^m, h_t^m)$ represents the **state** of the ship at time step t and includes four features: x_t^m and y_t^m denote the vessel's location in a two-dimensional plane (the planning region is treated as a 2D plane, ignoring Earth's curvature due to the relatively small area). v_t^m denotes the vessel's velocity relative to the ground, and h_t^m indicates the vessel's heading angle, representing its orientation. All trajectories are linearly interpolated into δ_T (= 10 second) intervals for improved granularity.

3.2 Ego Agents and Log-play Agents

We implement our environment using a log-play setting, featuring two types of agents: *ego agents* and *log-play agents*. The environment consists of n_E ego agent and n_L log-play agents, each associated with a historical trajectory. The Figure 2 demonstrates a scenario with one ego agent (red) and five log-play agents (blue). The ego agents are controlled by the RL/imitation learning algorithm, while the log-play agents passively replay their historical trajectories within the environment. If an ego agent is assigned the historical trajectory τ_m, it is initialized at the starting location (x_0^m, y_0^m) with speed v_0^m and heading h_0^m. Its objective is to reach

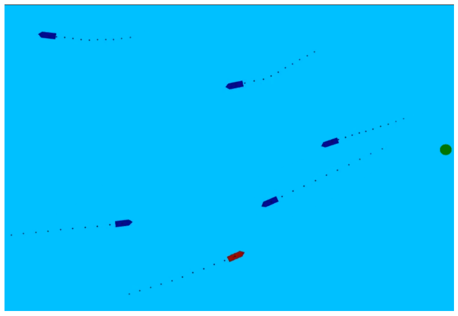

Figure 2: An example scenario from the environment showing five log-play agents (in blue) and one ego agent (in red). The dots behind the ships indicate the observable past states of the trajectories, and the green dot marks the goal location of the ego agent.

the endpoint $g_m = \langle x_T^m, y_T^m \rangle$ of the trajectory, defined as its goal. During training, ego agents are expected to reach their goals, avoid collisions with other vessels, and maintain minimal deviation from their original trajectories. An environment episode concludes when all ego agents have either reached their goals or exited the planning region. For ease of exposition and to avoid notation clutter, the following sections present the formulation in a single ego agent setting (i.e., $n_E = 1$). Once our imitation learning based policy is trained, we can use it to control multiple ego agents; also shown empirically.

3.3 Observation Space

The observation space is defined in the perspective of the ego agent. At time step t, the agent can observe the past H (a hyperparameter, we test for different H values) steps of its own trajectory and the past H-steps of closest 10 (a hyperparameter) nearby ships, and its goal location. This captures multi-ship interactions.

3.4 Action Space

In real-world scenarios, a ship's action space can be complex, involving multiple factors related to its dynamics. To simplify this, we model the ship's behavior using a straightforward, 3-dimensional continuous action space defined as $a = \langle d_x, d_y, d_h \rangle$, where d_x and d_y are the changes in the x and y coordinates, and d_h is the change in heading. This allows the next state of the ego vessel to be calculated as $l_{t+1} = \langle x_t + d_x, y_t + d_y, v_{t+1}, h_t + d_h \rangle$. The speed at the next time step, $v_{t+1} = \sqrt{d_x^2 + d_y^2}/\delta_T$, is determined by dividing the distance traveled by the time interval δ_T. This representation is also known as delta action space [17] which can be used for any kind of moving object.

3.5 Reward Function

We define the reward function as a sparse reward for the vessel. The agent receives a reward of 0 for each step until it reaches the goal. If the agent comes within a distance of ψ_G from the goal, it is considered to have reached it. Upon reaching the goal, the agent receives a reward of 1. To limit the trajectory length, we also truncate the episode at 1000 steps.

3.6 Evaluation Metrics

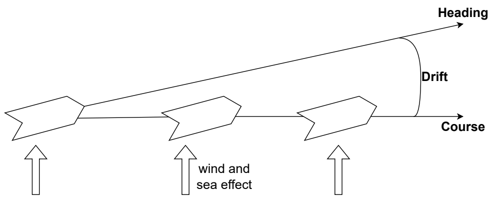

Figure 3: Illustration of drift

In this section, we present below metrics used for evaluating navigation policies in ShipNaviSim.

Average Displacement Error (ADE). : In our work, we use the Goal Conditioned ADE (GC-ADE) as defined in [13] to measure the difference between the behavior of our agent and that of the expert. Given an agent's original historical trajectory τ_m of length T_m and the actual trajectory generated by the learned policy, τ_p, with length T_p, the GC-ADE quantifies the deviation of the current trajectory from the original trajectory in the 2D plane:

$$\text{GC-ADE} = \frac{1}{\min(T_m, T_p)} \sqrt{\sum_{t=1}^{\min(T_m,T_p)} (x_t^m - x_t^p)^2 + (y_t^m - y_t^p)^2}$$

Goal Rate. This metric indicates the percentage of times the ego agent successfully reaches its goal. We set a threshold of $\psi_G = 200$ meters; if the agent comes within this radius of the final location, it is considered a success. Compared to the size of tankers and cargos (around 300 meters), ψ_G is relatively small.

Near Miss Rate. Domain experts consider that if a ship approaches another vessel within 3 cable lengths (555 meters), it qualifies as a near-miss scenario [8]. The near-miss rate represents the percentage of time-steps during which the ego agent encounters such scenarios.

Drift. Drift illustrated in Figure 3 is defined as the angle difference between the course-over-ground (COG) and the heading of the vessel. For a moving vessel, this value should typically be no more than a few degrees. We compare this metric with expert values to evaluate how closely the RL agent imitates expert behavior. Drift indirectly captures the effect of ocean currents, wind, and other weather condition on the vessel movement.

Curvature. Curvature measures the angle change in the vessel's course over ground at each time step, indicating the sharpness of the turn[1]. It helps assess the maneuverability and ensures the vessel's behavior aligns with expected patterns, particularly when comparing the RL agent's performance to that of an expert.

4 IMITATION LEARNING FOR SHIPNAVISIM

In this section, we focus on how we adapt various imitation learning algorithms such as Behavior Cloning (BC) [35], ODICE [25], and Inverse Soft-Q Learning (IQL) [14] for training agents in ShipNaviSim.

As mentioned in Section 3.1, we have the dataset D containing M historical trajectories which are sequences of ship states. For each trajectory in D, we can easily convert it into a sequence of environment states in Section 3.3. Based on these sequences, we can construct an environment expert demonstration $\mathcal{D}_{exp} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^{|\mathcal{D}_{exp}|}$ where s_i and s_i' are two consecutive ShipNaviSim states, a_i is the action taken in state s_i for reaching s_i', and r_i is a reward at s_i' obtained from the reward function (Section 3.5). By using delta action space, a_i in ShipNaviSim can be obtained from only s_i and its next state s_i'. Because BC only uses state-action information, we create another static dataset $\mathcal{D}_{exp}^{BC}$ consisting all pairs (s_i, a_i) from $\mathcal{D}_{exp}$.

Behavior Cloning. The goal of BC is to find a policy $\pi(s, a)$ optimizing the following function:

$$max_\pi \sum_{(s,a)\in\mathcal{D}_{exp}^{BC}} \log \pi(s, a) \tag{1}$$

This objective function of BC is quite popular in previous works and is used in modern imitation learning implementation [16] in which π is represented by a Gaussian policy. Problem (1) is then optimized by training the mean and covariance of π. Another way for modeling the policy in BC is using deterministic policy $\pi(s)$ which outputs the action directly. The objective function in this case become:

$$min_\pi \sum_{(s,a)\in\mathcal{D}_{exp}^{BC}} (\pi(s) - a)^2 \tag{2}$$

Interestingly, this objective function is very close to optimizing ADE, an important metric in our setting because we use the delta action space. For example, assuming that for a pair $(s, a) \in \mathcal{D}_{exp}^{BC}$ in which the expert state s is equal to x-axis and y-axis positions (x, y) and the expert action a based on the delta action space will be (d_x, d_y). The expert next state $s' = (x', y') = s + a$ and a will be equal to $s - s'$.

We denote $s_\pi' = (x_\pi', y_\pi')$ as the next state if action $\pi(s)$ is taken in state s. Given the delta action space, $\pi(s) = s_\pi' - s$. Therefore in this case equation (2) can be equivalent to:

$$min_\pi \sum_{(s,a)\in\mathcal{D}_{exp}^{BC}} (\pi(s) - a)^2 \tag{3}$$

$$=min_\pi \sum_{(s,a)\in\mathcal{D}_{exp}^{BC}} (s_\pi' - s - s' + s)^2 \tag{4}$$

$$=min_\pi \sum_{(s,a)\in\mathcal{D}_{exp}^{BC}} (s_\pi' - s')^2 \tag{5}$$

$$=min_\pi \sum_{(s,a)\in\mathcal{D}_{exp}^{BC}} [(x_\pi' - x')^2 + (y_\pi' - y')^2] \tag{6}$$

This is clearly a form of ADE metric which leads to an observation that in case of using delta action space for 2d coordinates, if we optimize the equation 2, we also minimize the ADE. Thus instead of using the common objective function (1), we choose deterministic policy $\pi(s)$ and equation (2) for learning BC in our experiments.

Policy network architecture. In our ShipNaviSim, the state space have multiple types of inputs so using only the normal feed-forward network was not enough based on our preliminary experiments.

Therefore, we adapt the late fusion architecture from self-driving cars work [29] to design our policy network as in Figure 4. Specifically, we separate a state into 3 parts: goal features containing goal location, ego features including only ego-agent information and ships-interaction features consisting 10 nearby ships state. These parts are processed by different extractors which are one or two linear layers with activation functions. All the hidden features obtained from three extractors will be inputted into the last layer with activation function. Variants of this architecture are used for ODICE and IQL to allow for stochastic policies.

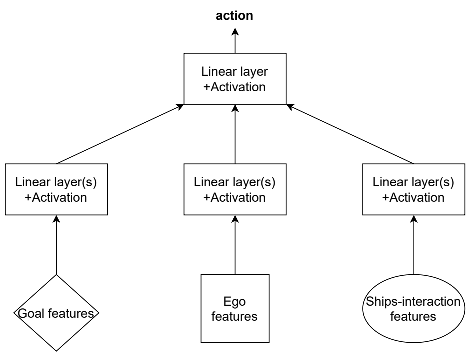

Figure 4: Policy Network architecture

ODICE and IQL. Because ODICE can be considered as both offline reinforcement learning and imitation learning method, we run both the two variants, which we name as ODICE-RL and ODICE-IL. The ODICE-RL needs to ultilize the reward information due to its offline RL setting. Based on our experiments, instead of using sparse reward like in our reward function, changing zero reward inside expert demonstration to -1 helped boost the performance of ODICE-RL. For IQL, we used their off-the-shelf implementation.

5 EXPERIMENTS

Our navigation data consist the information about vessels movement from mid 2017 to mid 2019 inside the hotspot planning region shown in Figure 1; other larger planning regions in TSS are also tested as shown in Figure 9. For a moving vessel, there is one AIS record for each 1-2 minute interval, which provides a granular picture of vessel movements in the Singapore strait.

The chosen planning region shown in figure 1 has a length of 6.68 km and a breadth of 7.77 km. We split these data into test and training dataset. The test dataset with a total of 2948 trajectories contains mid-2019 data and the remainder is used for training dataset with a total of 8437 trajectories.

Figure 5 illustrates the distribution of episode length in these datasets. Most of vessels move inside the planning region for more than 10 minutes which is equivalent to 600 steps in our simulation (each time step in our simulation is 10 seconds). The maximum trajectory length of training and test dataset can be up to 652 and 559, respectively. We use *minari* package [39] to create the offline expert demonstrations from training dataset for offline RL and

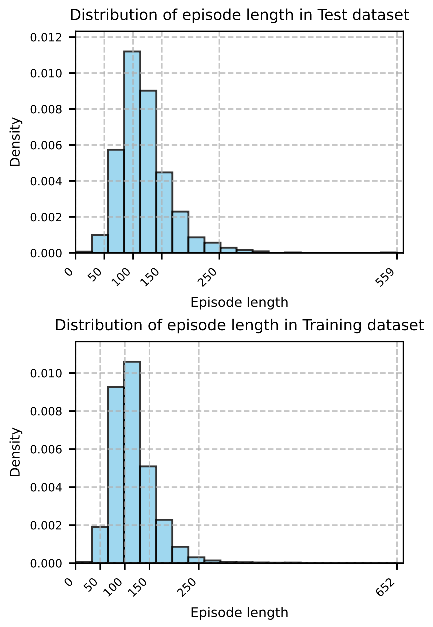

Figure 5: Distribution of episode length in the dataset

IL methods. Our simulator, trained policies and demo videos are publicly available can be found at our project website [1].

5.1 Simulation results

We train and test the BC with two settings, varying parameter H, that controls how much information from the past time steps is part of the observation space: $H = 0$ and $H = 5$. We call these two BC settings as *BC-0histL* and *BC-5histL*. Figures 6 and 7 illustrate the performance of 5 methods over 4 important metrics on training and test dataset, respectively.

In terms of Goal Rate, three methods BC-5histL, BC-0histL and ODICE-RL have good results which are always greater than 75% on both training and test modes. Both BC-5histL and IQL have the lowest near-miss rate of around 20% which is very close to the expert performance (15%). However, IQL has a very large GC-ADE value (over 10m on both datasets) which is more than about 20-40m compared to other approaches. This indicates that it rarely follows the expert trajectories. Furthermore, the average drift of IQL is also the biggest across different methods and datasets. In the worst case, this value can go up to 1 radian (about 57 degrees) which is highly infeasible in real-world scenarios. For ODICE methods, ODICE-RL is clearly better than ODICE-IL on all metrics with the help of reward augmentation as described in the previous section.

Overall, these results show that BC-5histL shows good, robust performance across different metrics in both training and testing modes. This is because the BC loss function with deterministic policy is similar to the GC-ADE metric, as discussed in section 4. Notice that GC-ADE achieved by BC-5histL of around 60m is fairly small given that we consider only large tankers and cargos with their length around 300 meters.

[1] https://shipnavisim.github.io

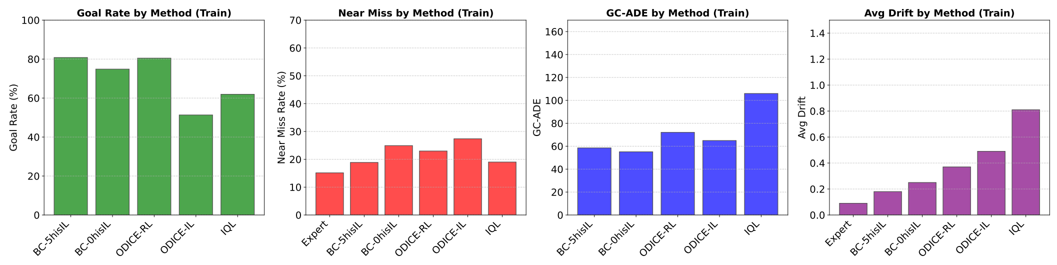

Figure 6: Comparisons on training data

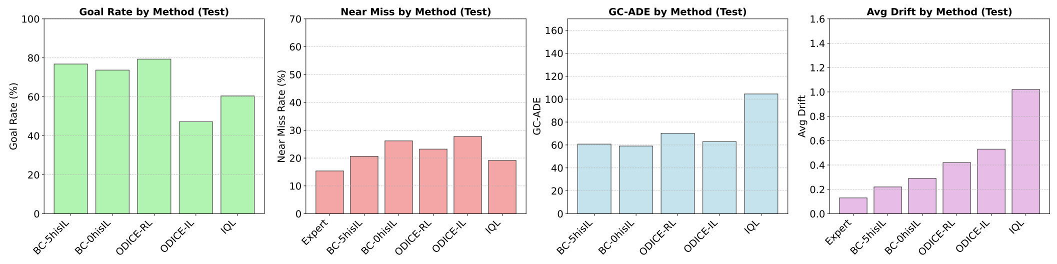

Figure 7: Comparisons on test data

Table 1: Additional kinematic metrics

Method	Data type	$\min_{\text{curvature}}$	$\max_{\text{curvature}}$	$\min_{\text{acceleration}}$	$\max_{\text{acceleration}}$	$\min_{\text{drift}}$	$\max_{\text{drift}}$
Expert	Train	-0.29	0.39	-2.00	2.35	-0.41	0.34
	Test	-0.24	0.32	-1.02	1.25	-0.47	0.34
BC-5histL	Train	-0.12	0.14	-0.98	0.78	-0.49	0.52
	Test	-0.07	0.15	-1.02	0.73	-0.57	0.61
BC-0histL	Train	-0.15	0.27	-1.05	0.84	-0.72	0.55
	Test	-0.19	0.21	-0.53	0.39	-1.00	0.68
ODICE-RL	Train	-0.25	0.11	-0.40	0.57	-0.92	1.1
	Test	-0.19	0.11	-0.35	0.51	-0.99	1.15
ODICE-IL	Train	0.00	0.21	-0.61	0.50	-1.90	0.53
	Test	0.00	0.22	-0.51	0.40	-2.34	0.55
IQL	Train	-12.42	2.25	-4.94	4.73	-3.14	3.14
	Test	-11.47	1.29	-5.02	4.96	-3.14	3.14

We further evaluate all agents more on some other kinematic metrics as shown in Table 1. Most methods have the reasonable bounding values of curvature except ODICE-IL and IQL. The minimum curvature value of ODICE-IL is zero which means that it always turns on one side of the vessel, which is not a reasonable behavior. In terms of IQL, its curvature bound values are very large, thus its turning angle is really sharp in the worst case, that is unrealistic for vessels. For acceleration and drift, IQL still has outlier values compared to other approaches and expert. The outstanding performance of BC-5histL clearly shows when we look at drift metrics. The worst drift angles of other methods are always bigger than 0.9 radian (about 51 degrees) while this value for BC-5hist is only about 0.61 radian (about 35 degrees).

Drift and weather effects. We also note AIS data does implicitly capture the effect of weather conditions on the ship movements (see [12]). The drift (defined on Section 3.6) captures the effect of weather, sea currents on vessel movements. In calm waters without any external forces, a vessel's heading and course-over-ground (COG) would be the same. However, this is not the case in our setting; external forces causes a vessel's heading to be different from COG, resulting in non-zero drift (see Figure 6, Table 1). As our IL algorithms capture both COG and heading, they capture the vessel

Table 2: Drift distribution comparison

Method	Wasserstein Distance	
	Train	Test
BC-5histL	0.193	0.224
BC-0histL	0.238	0.276

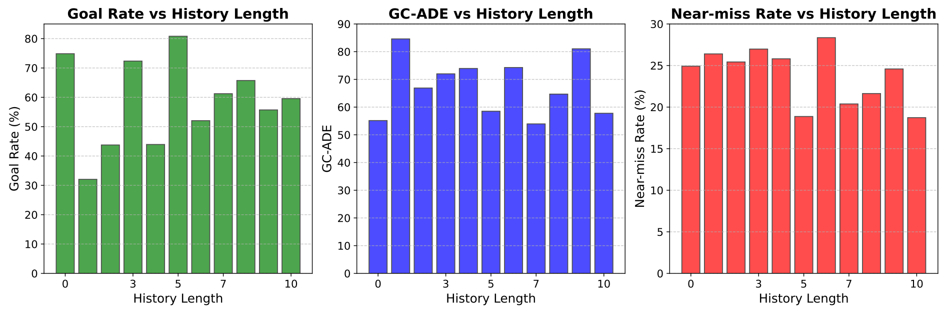

Figure 8: The impact of the observable past states (H)

movements caused by external factors. In Table 2, we also compare the distribution of drift (measured in radian) in historical data and the data generated by BC. For interpretation, the Wasserstein distance (optimal transport) of 0.19 radian (about 10.8 degrees, BC-5histL-Train) is a small value for a larger vessel. This indicates our BC algorithm can capture external forces.

5.2 Ablations on the nearby vessels and observable history features

Table 3: Impact of Nearby Ship Information

Method	Goal Rate ↑	Near Miss Rate ↓	ADE ↓
BC-0hist	**74.86**	**24.92 ± 29.50**	**55.14 ± 41.48**
BC-0hist-Drop	69.54	25.59 ± 30.77	79.94 ± 50.99
BC-5hist	**80.81**	**18.87 ± 26.47**	**58.51 ± 35.39**
BC-5hist-Drop	9.33	34.61 ± 33.21	84.67 ± 46.57

Impact of Nearby Ship Features. We train the Behavior Cloning (BC) policy with and without nearby vessel features to evaluate their impact. In Table 3, we present the results for *BC-0hist* and *BC-5hist*, which incorporate nearby ship information with history lengths of 0 and 5, respectively. We show results after removing nearby ship information, denoted as *BC-0hist-Drop* and *BC-5hist-Drop*. The results clearly indicate that removing nearby vessel information leads to a significant decline in performance. In the 0-history length setting, this results in a 5% reduction in goal rate, a 1-2% increase in near-miss rate, and an increase in displacement error of approximately 25 meters. In the 5-history length setting, the impact is even more pronounced, with a 70% reduction in goal rate, a 16% increase in near-miss rate, and an increase in ADE of 26 meters. In summary, we conclude that nearby ship information is crucial to learn an effective policy. Thus, our BC-based policies are able to effectively capture multi-vessel interactions.

Effect of Observing Past States of the Trajectory. To evaluate the impact of observing past states, we train our model with varying H values ranging from 0 to 10, representing the number of past states visible to the agent, as discussed in Section 3.3. The results, shown in Figure 8, include the Goal Rate, Near-miss Rate, and ADE metrics. We focus on presenting BC in this ablation, as it consistently demonstrates the best performance overall, as discussed in Section 5.1. The model with $H = 5$ achieves the highest goal rate of 81%, along with a significantly lower near-miss rate and ADE. In contrast, the model with $H = 1$ shows the lowest goal rate at approximately 32%. This indicates a clear effect of allowing the agent to observe past trajectory steps.

5.3 Comparison with trajectory prediction

Table 4: Results of comparing between BC and trajectory prediction method

Method	Data type	Goal Rate ↑	Near Miss Rate ↓	ADE ↓
Expert	Train	100.00	15.14	0.00
	Test	100.00	15.37	0.00
BC-5histL	Train	80.81	18.87	58.51
	Test	76.83	20.62	60.75
Autobot	Train	27.85	14.96	169.13
	Test	24.42	15.09	170.17

We want to note that trajectory prediction methods focus on short-term motion forecasting while our work deals with long-term motion planning (vessel trajectories can be more than 30 minutes long in our setting). They are two distinct problems with different purposes [11]. Although we can apply trajectory forecasting approach on the planning task by using predicted trajectory as historical features to re-plan after some environment steps, its performance is still worse than BC in autonomous driving scenario [17]. We also conduct experiments to show the similar phenomenon in our maritime setting where we use Autobot [15] method as the trajectory prediction baseline. As shown in the Table 4, Autobot has worse performance in terms of ADE and goal rate metrics but surprisingly has lower near-miss rate even than expert. This is due to the early truncation of Autobot's trajectories in which the episode is ended because the ego agent goes outside the map thus avoiding most of close-quarter situations.

5.4 Self-play results

In the previous sections, we train and evaluate agents in single-agent setting. However, maritime traffic management involves multiple agents. We also want more agents to be controlled by learned polices for generating diverse scenarios. Therefore, we evaluate our best agent BC-5histL on multi-agent setting in this section.

Table 5: Self Play Results (50% agents controlled by BC)

Method	Dataset	Goal Rate ↑	ADE ↓		
			Mean	Min	Max
BC-5histL	Train	63.4	68.92 ± 41.03	35.84 ± 37.85	106.07 ± 65.26
	Test	58.92	66.98 ± 38.84	34.32 ± 35.45	104.75 ± 63.11

Table 6: Self Play Results (ego agent perspective)

Method	Dataset	Goal Rate ↑		ADE ↓		Near Miss Rate ↓	
		Log-play	Self-play	Log-play	Self-play	Log-play	Self-play
BC-5histL	Train	80.81	72.32	58.51 ± 35.39	98.41 ± 67.50	18.87 ± 26.47	16.65 ± 27.05
	Test	76.83	69.64	60.75 ± 36.81	96.88 ± 66.16	20.62 ± 27.88	17.79 ± 27.65

For each ego agent scenario at the starting time step, we choose randomly 50% of agents inside its view to be controlled by the trained BC policy. If a self-play agent reaches the goal or moves out the planning region, it will be removed from the current view. The episode stops when there are no controlled agents in the planning area or the total number of simulation steps reach its limit. Table 5 show the results in multi-agent setting. For computing the goal rate, we adopt a stricter definition than the single ego agent setting. Only if *all* the BC-controlled agents reach their respective goals, we count it as goal achieved.

For each episode, we compute the min and max ade values over controlled agents. The column Min and Max in Table 5 show the average of these values across all episodes. We can see that the goal rates in both training and test datasets in case of multi agent are higher than 55%. The mean of ADE increases a little, about 10-15m, compared to single-agent results, which shows that vessels movements are still close to their historical trajectories. In the best case, the average of min ADE values is under 40 showing BC policy can still control some agents quite well.

We further investigate the effect of self-play scenarios on the performance of ego agent. Table 6 shows the goal rate, ade and near miss rate of ego agents for two settings:

- When surrounding agents of the ego agent are controlled by the log data replay ('Log-play')
- When 50% surrounding agents are controlled by the BC policy ('self-play')

We can clearly see that the performance in both log-play and self-play settings are roughly similar for various metrics. Even though self-play setting is more challenging setting, BC policy still manages to learn well from the historical data, and on the test dataset also, its performance is good, on-par with log-play.

5.5 Results on larger planning areas

We extend ShipNaviSim results to two larger planning regions: Region 1 (9.89 km long and 5.31 km wide) and Region 2 (12.52 km long and 9.01 km wide) as illustrated in Figure 9. The results in Table 7 show that BC still have stable performance on larger regions. The goal rate drops slightly but remains in a good range of 63-71%. The near-miss rate stays consistent between 19-21%, and the ADE remains relatively low at 59-72 meters. These results clearly demonstrate the effectiveness and flexibility of our approach when applying to other regions.

Table 7: Results on other planning regions

Method	Data type	Goal Rate ↑	Near Miss Rate ↓	ADE ↓
		Region 1		
Expert	Train	100	14.15	0.00
	Test	100	12.09	0.00
BC-5histL	Train	71.86	21.04	59.75
	Test	65.78	21.11	65.75
		Region 2		
Expert	Train	100	11.43	0.00
	Test	100	11.04	0.00
BC-5histL	Train	64.89	19.10	67.71
	Test	63.29	19.68	72.72

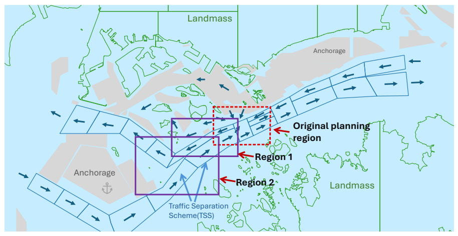

Figure 9: Larger planning regions (purple rectangles)

6 CONCLUSION

We developed ShipNaviSim, a high fidelity simulator based on real world maritime traffic data for simulating vessel traffic in busy port areas. Given the importance of maritime traffic to the world trade, and high environmental and human risks of accidents, developing such accurate simulators is crucial to test and validate maritime traffic control strategies. We integrated several imitation learning based methods with our simulator, and showed that the learned policy aligns well with the historical data. We also developed several maritime traffic related metrics to validate the behavior of learned policies. Our results also highlighted the importance of considering multi-vessel interactions to learn accurate movement patterns.

ACKNOWLEDGMENTS

This research/project is supported by the National Research Foundation, Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2- RP-2020-017).

REFERENCES

[1] 2024. Curvature. https://en.wikipedia.org/w/index.php?title=Curvature&oldid=1251369919 Page Version ID: 1251369919.

[2] 2024. Oil tankers collision off Singapore Strait spotlight perils of dark fleet ships. https://www.scmp.com/news/asia/southeast-asia/article/3271072/blazing-oil-tankers-singapore-strait-spotlight-perils-dark-fleet-ships

[3] Lucas Agussurja, Akshat Kumar, and Hoong Chuin Lau. 2018. Resource-constrained scheduling for maritime traffic management. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 214–223.

[5] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. 2018. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079* (2018).

[6] Chaithanya Basrur, Arambam James Singh, Arunesh Sinha, and Akshat Kumar. 2021. Ship-GAN: Generative Modeling Based Maritime Traffic Simulator. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (Virtual Event, United Kingdom) *(AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1755–1757.

[7] Chaithanya Basrur, Arambam James Singh, Arunesh Sinha, Akshat Kumar, and T. K. Satish Kumar. 2022. Trajectory Optimization for Safe Navigation in Maritime Traffic Using Historical Data. In *28th International Conference on Principles and Practice of Constraint Programming (CP 2022) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 235)*, Christine Solnon (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 5:1–5:17. https://doi.org/10.4230/LIPIcs.CP.2022.5

[8] Saumya Bhatnagar, Akshat Kumar, and Hoong Chuin Lau. 2019. Decision Making for Improving Maritime Traffic Safety Using Constraint Programming. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 5794–5800. https://doi.org/10.24963/ijcai.2019/803

[9] Raunak P Bhattacharyya, Derek J Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J Kochenderfer. 2018. Multi-agent imitation learning for driving simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1534–1539.

[10] Eli Bronstein, Mark Palatucci, Dominik Notz, Brandyn White, Alex Kuefler, Yiren Lu, Supratik Paul, Payam Nikdel, Paul Mougin, Hongge Chen, et al. 2022. Hierarchical model-based imitation learning for planning in autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8652–8659.

[11] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. 2021. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810* (2021).

[12] Konstantinos Christodoulou, Herodotos Herodotou, and Michalis P Michaelides. 2022. Estimation of sea surface current velocities using AIS data. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 407–412.

[13] Daphne Cornelisse and Eugene Vinitsky. 2024. Human-compatible driving partners through data-regularized self-play reinforcement learning. *arXiv preprint arXiv:2403.19648* (2024).

[14] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. 2021. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems* 34 (2021), 4028–4039.

[15] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. 2022. Latent Variable Sequential Set Transformers for Joint Multi-Agent Motion Prediction. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Dup_dDqkZC5

[16] Adam Gleave, Mohammad Taufeeque, Juan Rocamonde, Erik Jenner, Steven H Wang, Sam Toyer, Maximilian Ernestus, Nora Belrose, Scott Emmons, and Stuart Russell. 2022. imitation: Clean imitation learning implementations. *arXiv preprint arXiv:2211.11972* (2022).

[17] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. 2024. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems* 36 (2024).

[18] Siyu Guo, Xiuguo Zhang, Yisong Zheng, and Yiquan Du. 2020. An autonomous path planning model for unmanned ships based on deep reinforcement learning. *Sensors* 20, 2 (2020), 426.

[19] S Hochreiter. 1997. Long Short-term Memory. *Neural Computation MIT-Press* (1997).

[20] AN Ince and E Topuz. 2004. Modelling and simulation for safe and efficient navigation in narrow waterways. *The Journal of Navigation* 57, 1 (2004), 53–71.

[21] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. 2019. Learning to drive in a day. In *2019 international conference on robotics and automation (ICRA)*. IEEE, 8248–8254.

[22] Quanyi Li, Zhenghao Mark Peng, Lan Feng, Zhizheng Liu, Chenda Duan, Wenjie Mo, and Bolei Zhou. 2024. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. *Advances in neural information processing systems* 36 (2024).

[23] Jiajing Ling, Arambam James Singh, Nguyen Duc Thien, and Akshat Kumar. 2022. Constrained multiagent reinforcement learning for large agent population. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 183–199.

[24] Esther Loi. 2023. Singapore port hits all-time high of 3 billion gross tons in vessel arrivals in 2023. *The Straits Times* (12 2023). https://www.straitstimes.com/singapore/transport/singapore-port-nets-all-time-high-of-3b-gross-tons-in-vessel-arrivals-in-2023

[25] Liyuan Mao, Haoran Xu, Weinan Zhang, and Xianyuan Zhan. 2024. ODICE: Revealing the Mystery of Distribution Correction Estimation via Orthogonal-gradient Update. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=L8UNn7Llt4

[26] Eivind Meyer, Haakon Robinson, Adil Rasheed, and Omer San. 2020. Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning. *IEEE Access* 8 (2020), 41466–41481.

[27] Faris Mokhtar. 2017. Busy shipping lane's narrow passageway hard for vessels to navigate. https://www.todayonline.com/singapore/busy-shipping-lanes-narrow-passageway-hard-vessels-navigate

[28] MPA. 2021. Vessel Traffic Information System. https://www.mpa.gov.sg/web/portal/home/port-of-singapore/operations/vessel-traffic-information-system-vtis.

[29] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. 2023. Wayformer: Motion forecasting via simple & efficient attention networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2980–2987.

[30] Shinkyu Park, Michal Cap, Javier Alonso-Mora, Carlo Ratti, and Daniela Rus. 2020. Social trajectory planning for urban autonomous surface vessels. *IEEE Transactions on Robotics* 37, 2 (2020), 452–465.

[31] Dean A Pomerleau. 1988. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems* 1 (1988).

[32] Arambam James Singh, Akshat Kumar, and Hoong Chuin Lau. 2020. Hierarchical Multiagent Reinforcement Learning for Maritime Traffic Management. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems* (Auckland, New Zealand) *(AAMAS '20)*. International Foundation for Autonomous Agents and Multiagent Systems, 1278–1286.

[33] Arambam James Singh, Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. 2019. Multiagent decision making for maritime traffic management. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6171–6178.

[34] Teck-Hou Teng, Hoong Chuin Lau, and Akshat Kumar. 2017. Coordinating Vessel Traffic to Improve Safety and Efficiency. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS '17)*. International Foundation for Autonomous Agents and Multiagent Systems, 141–149.

[35] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral Cloning from Observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden, 4950–4957. https://doi.org/10.24963/ijcai.2018/687

[36] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical review E* 62, 2 (2000), 1805.

[37] UNCTAD. 2023. Review of maritime transport | UNCTAD. Retrieved October 12, 2024 from https://unctad.org/topic/transport-and-trade-logistics/review-of-maritime-transport

[38] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. 2022. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 7814–7821.

[39] Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. 2024. *Minari*. https://doi.org/10.5281/zenodo.13767625

[40] Yang Zhou, Winnie Daamen, Tiedo Vellinga, and Serge Hoogendoorn. 2019. Review of maritime traffic models from vessel behavior modeling perspective. *Transportation Research Part C: Emerging Technologies* 105 (2019), 323–345.

[41] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. 2023. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 17863–17873.