Robust Offline Reinforcement Learning for Autonomous Vessel Navigation

Journal of Physics: Conference Series **3123** (2025) 012006 doi:10.1088/1742-6596/3123/1/012006

Robust Offline Reinforcement Learning for Autonomous Vessel Navigation

Bavo Lesy, Siegfried Mercelis and Ali Anwar

IDLab, Faculty of Applied Engineering, University of Antwerp - imec, Antwerp, Belgium

E-mail: bavo.lesy@uantwerpen.be

Abstract. Reinforcement Learning (RL) has shown promising advances in autonomous navigation, particularly in learning adaptive control policies through interaction with dynamic environments. However, training RL agents typically require direct interaction with the real environment, which can be dangerous and costly. To mitigate this, offline RL has emerged as a promising alternative, where agents learn from pre-collected datasets. In this work, we study the application of offline RL approaches in inland shipping, comparing the performance and robustness of Behavior Cloning (BC) and Conservative Q-Learning (CQL), two offline approaches, with Soft Actor-Critic (SAC), an online RL algorithm. We design experiments to test the robustness of these algorithms within the domain of unmanned surface vehicles (USVs), using an autonomous shipping simulator based on MOOS-IvP with varied environmental conditions. Our results demonstrate that both offline approaches can achieve performance comparable to the expert online policy in nominal conditions while maintaining similar robustness characteristics in out-of-distribution scenarios. Furthermore, we investigate the impact of dataset quality on offline RL performance, showing that with sufficient data, both BC and CQL can learn policies that match the dataset quality and achieve robustness similar to online RL algorithms.

1 Introduction

In recent years, there has been a growing interest in the development of autonomous shipping systems. This interest is driven by an increased demand for safer, more efficient and sustainable transportation methods. Inland Waterway Transport (IWT) is seen as a sustainable alternative to road and rail transport, with a study by the European Commission showing that IWT has a substantially lower carbon footprint than road and rail transport [3]. Beyond environmental benefits, IWT contributes to reducing road congestion and offers lower transport costs for large volumes of goods. However, in some European countries such as Belgium and The Netherlands, the size of IWT has been declining in recent years, mostly due to a shortage of qualified skippers and the harsh competition from road and rail transport. As these countries are home to the two largest ports in Europe and have a high density of inland waterways, they could benefit greatly from autonomous shipping and increased IWT capacity. The Central Commission for the Navigation of the Rhine (CCNR) has defined 6 levels of automation in inland waterway navigation [2]. Existing autonomy solutions predominantly reach level 1 or 2, where a navigation system partially automates the steering and propulsion. In these systems, a human operator performs all remaining aspects of dynamic navigation, namely route planning, obstacle detection and avoidance. To increase the efficiency of IWT, the industry is moving towards level 3 autonomy, where vessels can navigate autonomously under specific conditions, with a remote human operator remaining available to

intervene in case of overtake requests or system failures. With level 3 autonomy, a single remote operator can monitor and manage multiple vessels concurrently, thereby reducing the need for additional skippers and increasing the operational efficiency and economic viability of IWT.

Within the field of motion planning, much research is currently being done to develop level 3 autonomous navigation solutions. In the context of IWT, motion planning is the process of determining the most optimal path for a vessel to follow while avoiding obstacles and adhering to constraints such as fuel efficiency, safety, and regulatory compliance. Long-established algorithms such as Rapidly Exploring Random Tree (RRT*) [12] and Dynamic Window Approach (DWA) [6] have shown to be effective for the autonomous navigation of unmanned surface vehicles (USVs) [21, 13]. However, these algorithms assume only static obstacles. Furthermore, the need to manually tune the parameters of these algorithms for different scenarios and vessel types along with the computational complexity of these algorithms may render them unsuitable for real-world deployment. To address these challenges, Reinforcement Learning (RL) has recently emerged as a promising solution to the motion planning problem for USVs. RL offers several advantages over traditional approaches: it can automatically learn optimal policies without explicit parameter tuning, adapt to different scenarios through experience, and potentially discover more efficient solutions than hand-crafted algorithms. RL is a machine learning framework that enables agents to learn optimal decision-making strategies through trial and error interaction with their environment. The learning process is guided by a reward function that quantifies the desirability of different actions and states, allowing the agent to gradually improve its performance over time. However, training RL agents presents its own set of challenges. To learn these decision-making policies, RL agents typically require extensive interaction with their environment, which can be dangerous, time-consuming, and costly when dealing with physical vessels. To mitigate these risks, RL agents are often trained in simulated environments, allowing them to explore freely and learn from a vast number of experiences. However, this approach introduces the Sim-to-Real (S2R) gap, a discrepancy between simulated and real-world environments, leading to challenges in deploying RL solutions in physicals settings.

The field of offline RL [17] has been proposed as a solution to the S2R gap. Offline RL agents learn a policy from pre-collected datasets, which can be collected in simulation or real-world. This approach offers several advantages: it eliminates the need for real-time interaction with the environment during training and reduces the risk of unsafe exploration. However, offline RL agents face significant challenges in generalization, particularly when faced with Out-of-Distribution (OOD) scenarios. Given the high-dimensional nature of IWT, it is intractable to collect data for every potential scenario, making robustness a key concern. To develop economically viable autonomous shipping solutions, RL models must generalize across different vessel types, weather conditions, and environmental disturbances. This requires the development of algorithms that can effectively learn from limited data while maintaining robust performance in scenarios not present in the training data. To deal with the limited generalization of offline RL agents, we investigate the robustness of common online and offline RL approaches in IWT. Robust RL algorithms are designed to learn policies that are resilient to variations in the environment and more capable of handling unseen scenarios. The challenge lies in balancing the trade-off between robustness and performance, as overly conservative policies may fail to achieve optimal navigation while overly aggressive ones may lead to unsafe behaviors. In this paper, we compare the robustness of various offline RL algorithms for path planning in an autonomous shipping simulator. We design experiments to evaluate robustness of offline RL algorithms in IWT by analysing their performance under varying environmental conditions, comparing them with an online RL method. This research represents the first comprehensive investigation into how robust offline RL algorithms can be validated for path planning in autonomous shipping simulation environments. By investigating the robustness, our work contributes to the development of more reliable autonomous shipping solutions, narrowing the gap between simulation and real-world deployment.

This work is structured as follows: In Section 2, we review the related work in the field of autonomous shipping, offline RL and robust RL. In Section 3, we describe RL and the main methodology used in this work. In Section 4, we describe the shipping simulator used for our experiments. In Section 5, we present the results of the experiments. Finally, in Section 6, we draw conclusions from the results and outline future work.

2 Related Works

The field of autonomous shipping has seen significant advancements in recent years, with various approaches being developed for vessel navigation and control. Traditional approaches to autonomous shipping navigation have primarily relied on classical control methods and search algorithms. A recent work by [21] combined DWA with RRT* to create a robust navigation system for autonomous surface vehicles. This hybrid approach addresses the challenges of both local collision avoidance and global

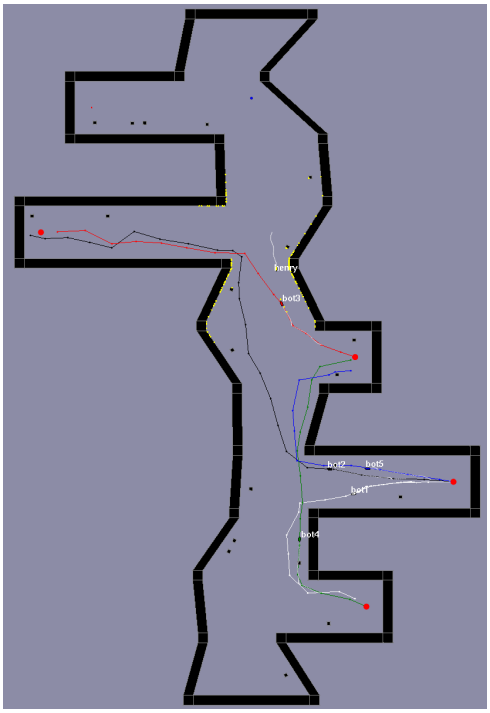

Figure 1: A screenshot from the MOOS-IvP simulator. The RRT* algorithm is used to plan the path of the ego vessel (henry). The other vessels (bots) and their paths are also shown. The yellow dots indicate the obstacles detected by the ego vessel's ranging sensor.

path planning. Similarly, Kim et al. [13] developed a collision avoidance algorithm specifically designed for USVs that adheres to the International Regulations for Preventing Collisions at Sea (COLREGs). Their approach demonstrates how traditional control methods can be enhanced with rule-based systems to ensure safe navigation in compliance with maritime regulations. More advanced approaches, such as Model-Predictive-Control (MPC), have been proposed to solve autonomous navigation for USVs [31]. [28] expands upon this work by combining MPC with Artificial Potential Field (APF) to create a simultaneous planning and control method. For a more comprehensive review of path planning methods for USVs, we refer the reader to Öztürk et al. [18]. While traditional methods have established a foundation for autonomous shipping, they face significant limitations in complex, dynamic environments, requiring extensive manual tuning and struggling with real-time adaptation to changing conditions. This has led to an increased interest in learning-based approaches and RL in particular, which can develop adaptive control policies through environmental interaction. Recent works have shown promising results applying online RL to autonomous shipping motion planning [27, 26, 9]. However, a critical gap remains: most research evaluates algorithms in single environments without testing their robustness across varying environmental conditions. In a previous work [15], we demonstrated that the online Soft Actor-Critic (SAC) algorithm exhibits considerable robustness to changes in environment dynamics, maintaining decent performance even when faced with significant environmental variations. This robustness is essential for autonomous shipping systems, as they must navigate diverse environments with different vessel types and conditions. Developing robust algorithms that can be trained on a single vessel type and deployed across various scenarios reduces development costs and accelerates deployment, enabling autonomous shipping systems to generalize effectively to diverse operational scenarios. Given the potential to utilize real-world collected datasets, offline RL algorithms have seen a recent surge in interest for autonomous shipping applications. Zhou et al. [32] proposed an offline RL algorithm for path following of USVs that utilizes real-world data to train the policy. Furthermore, Sun et al. [22] develop USVs for cooperative defense of marine infrastructure using a Behavioral Cloning (BC) approach. However, their work did not investigate the robustness of the algorithm. Robustness is particularly critical for offline RL algorithms, which often struggle with generalization when faced with OOD scenarios that were not represented in their training datasets. In this work, we investigate the robustness of offline RL algorithms in the context

of autonomous shipping, addressing an understudied area in the literature. We analyze how these algorithms perform when faced with environmental variations and vessel parameter changes absent from their training data. Challenging the common hypothesis that offline RL algorithms are inherently less robust than their online counterparts, we demonstrate that offline methods can achieve comparable robustness when properly designed. This aligns with recent findings [5], which suggest that the performance gap between offline and online reinforcement learning is smaller than previously thought. Our work compares the performance and robustness of offline RL algorithms to the simpler BC approach, while examining their resilience to OOD data relative to online reinforcement learning algorithms. Additionally, we investigate how dataset quality affects the performance of offline reinforcement learning algorithms, providing insights into the practical deployment considerations for autonomous shipping systems.

3 Preliminaries

3.1 Reinforcement Learning

RL is a branch of machine learning focused on solving sequential decision-making problems [24]. These problems are formally represented as Markov Decision Processs (MDPs). An MDP is defined by a tuple (S, A, P, R, γ), where S is the set of states, A is the set of actions, $P(s'|s,a)$ is the transition probability from state s to state s' given action a, $R(s,a)$ is the reward for taking action a in state s, and γ is the discount factor which balances the trade-off between near-term and long-term rewards.

In an MDP, an agent interacts with an environment over a sequence of timesteps, producing a trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_H)$. At each timestep t, the agent observes the current state $s_t \in S$, selects an action $a_t \in A$ according to its policy π, and receives a reward $r_t = R(s_t, a_t)$ as feedback. The environment then transitions to a new state s_{t+1} according to the probability distribution $P(s_{t+1}|s_t, a_t)$. This interaction continues until the episode terminates, either after a fixed number of timesteps (horizon H) or upon reaching a terminal state. In this work, the terminal state is defined as the vessel reaching the goal waypoint or colliding with an obstacle. The objective of RL is to learn an optimal policy π^* (Equation 1) which maximizes the expected cumulative reward over trajectories, which in our experiments leads to an agent that successfully navigates the vessel to the goal waypoint. This objective is formalized as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{H} \gamma^t \, r(s_t, a_t) \right], \tag{1}$$

where $p_\pi(\tau) = P(s_0) \prod_{t=0}^{H-1} \pi(a_t \mid s_t) \, P(s_{t+1} \mid s_t, a_t)$ is the probability of a trajectory under policy π and environment dynamics P. Many RL algorithms, utilize Q-learning [29] to learn an action-value function (Q-function) to determine the optimal policy. The Q-function $Q(s,a)$ quantifies the value of taking action a in state s. Specifically, the Q-function for a given policy π, denoted $Q^\pi(s,a)$, is the expected cumulative discounted reward for taking action a in state s and subsequently following policy π. The optimal Q-function, $Q^*(s,a)$, represents the maximum expected cumulative discounted reward achievable by taking action a in state s and then following the optimal policy π^*. Q-learning is based on temporal difference (TD) learning, where agents learn value functions by bootstrapping from estimated future values rather than waiting for complete episode returns. The algorithm updates Q-values using the immediate reward plus the discounted maximum future Q-value. The optimal Q-function satisfies the Bellman optimality equation and can be computed recursively:

$$Q^*(s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a) + \gamma \max_{a'} Q^*(s', a') \right] \tag{2}$$

Once all the Q-values are known, the optimal policy π^* is obtained by selecting the action a that maximizes $Q^*(s,a)$ for each state s:

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s,a) \tag{3}$$

3.2 Offline Reinforcement Learning

Traditional online RL is challenging for real-world problems such as autonomous shipping due to its reliance on direct environmental interaction for training. This interaction poses substantial risks (e.g., costly collisions, safety incidents) and is often time-consuming. Offline RL addresses these limitations while maintaining the same objective of finding an optimal policy (Equation 1). Instead of interacting with the environment, the agent learns from a dataset of transitions, collected using a policy π_D (typically gathered from expert demonstrations or human operators). The dataset D (Equation 4) is represented as a collection of transitions with size T, each of which is a tuple containing the current state s_t, the

action taken a_t, the reward received r_t, the next state s_{t+1}, and a boolean flag indicating whether the transition resulted in a terminal state:

$$D = (s_t, a_t, r_t, s_{t+1}, \text{terminal}_t)_{t=0}^{T} \tag{4}$$

Since the agent cannot explore beyond the states and actions represented in the dataset, the quality and coverage of the dataset are critical factors in determining the success of the learned policy. For the experiments performed in this work, we collected multiple datasets from the MOOS-IvP (uSimMarine) shipping simulator [1]. Following the methodology established in the D4RL benchmark [7], we collected datasets under different policies with varying levels of quality to evaluate the robustness of offline RL algorithms under different data distributions. We specifically generated three types of datasets. π_{expert} is the policy of a fully trained SAC agent, π_{medium} is a policy of a partially trained SAC agent, and π_{random} is a policy of uniform random actions. All datasets consist of 500,000 transition samples. These diverse datasets allow us to evaluate how different offline RL algorithms perform when learning from datasets with varying quality and coverage of the state-action space.

In this work, we focus on two offline algorithms: BC and Conservative Q-Learning (CQL). BC (or imitation learning) employs a pure supervised learning approach, training the agent to imitate the behavior policy (π_D) present in the dataset. The objective $\mathcal{J}_{BC}$ is to minimize the expected difference between the learned policy π_θ and the behavior policy π_D present in the dataset. Formally, this can be expressed as:

$$\mathcal{J}_{BC}(\theta) = \min_\theta \mathbb{E}_{(s,a)\sim D}\left[\|a - \pi_\theta(s)\|^2\right], \tag{5}$$

where θ represents the parameters of the learned policy and (s, a) pairs are sampled from the dataset D. For each observation in the dataset, the agent learns to predict the corresponding action taken by π_D. Since BC does not utilize the reward signal during this process, its effectiveness is highly dependent on the quality of the dataset. If the dataset contains suboptimal actions, the BC agent will learn to imitate these actions, leading to poor performance. In contrast to BC, CQL [14] aims not to imitate π_D, but to learn an optimal policy by estimating the Q-values from the dataset and selecting actions that maximize these values. The core objective of CQL is to learn a conservative Q-function that avoids overestimation of out-of-distribution (OOD) actions while still learning from the dataset. To address the critical issue of Q-value overestimation for OOD actions in offline RL [30], the CQL objective $\mathcal{J}_{CQL}$ consists of two main components: a standard Q-learning term and a conservative regularization term:

$$\mathcal{J}_{CQL}(\theta) = \min_\theta \mathcal{L}_{TD}(\theta) + \alpha\mathcal{L}_C(\theta), \tag{6}$$

where $\mathcal{L}_{TD}(\theta)$ is the standard TD loss:

$$\mathcal{L}_{TD}(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[(Q_\theta(s,a) - (r + \gamma \max_{a'} Q_\theta(s', a')))^2\right], \tag{7}$$

and $\mathcal{L}_C(\theta)$ is the conservative regularization term that penalizes high Q-values for actions not present in the dataset:

$$\mathcal{L}_C(\theta) = \mathbb{E}_{s\sim D}\left[\log \sum_a \exp(Q_\theta(s,a)) - \mathbb{E}_{a\sim\pi_D(\cdot|s)}[Q_\theta(s,a)]\right] \tag{8}$$

The hyperparameter α controls the strength of the conservative penalty. The first term acts as a soft maximum over Q-values for all possible actions. By itself, this term would push up all Q-values. However, the second term counteracts this by encouraging high Q-values specifically for actions that appear in the dataset. When minimizing the difference between these terms, the Q-values for OOD actions are reduced while the high values for in-distribution actions are maintained. The use of Q-values gives CQL the potential to improve upon π_D, particularly when the dataset is of suboptimal quality [10]. By learning to select actions that maximize Q-values, CQL can disregard bad actions that might be present in the dataset, a significant advantage over BC, which purely imitates the data. However, for this learning to be effective the dataset must adequately cover the space of high-reward state-action pairs [17]. Furthermore, a key challenge with CQL is its sensitivity to the conservative penalty term, which, if improperly tuned, can lead to an overly conservative policy. In Section 5.2, we test the hypothesis that CQL can learn a better policy than BC from suboptimal datasets, and potentially even outperform π_D, by comparing their performance when using datasets of varying quality. For our experiments, we used the OfflineRL-Kit [23] implementation of BC and CQL.

4 Simulation Environment

4.1 MOOS-IvP Simulator

Our experiments were performed in the MOOS-IvP (uSimMarine) simulator [1] (see Figure 1). Previous work [9] has adapted this simulator for RL research by extending it to include procedurally generated port environments. This extension addresses the distributional shift problem, where agents trained in a single environment fail to generalize to new scenarios. Through domain randomization [25], we train the agent on diverse scenarios by randomizing the port environment every 4 episodes, along with the goal locations and routes of other vessels. For more detailed information on these procedurally generated environments, readers are referred to [9]. It's important to note that our agent operates without direct knowledge of other vessels' locations, instead relying on a ranging ray-casting sensor to detect both static and dynamic obstacles. For vessel navigation, we implement a two-part motion planning approach. A global motion planner determines the overall path using a set of waypoints, whilst a local motion planner handles navigation between these waypoints. Specifically, we employ the RRT* algorithm [12] for global path planning, and a RL agent for local navigation and collision avoidance.

4.2 Vessel Model

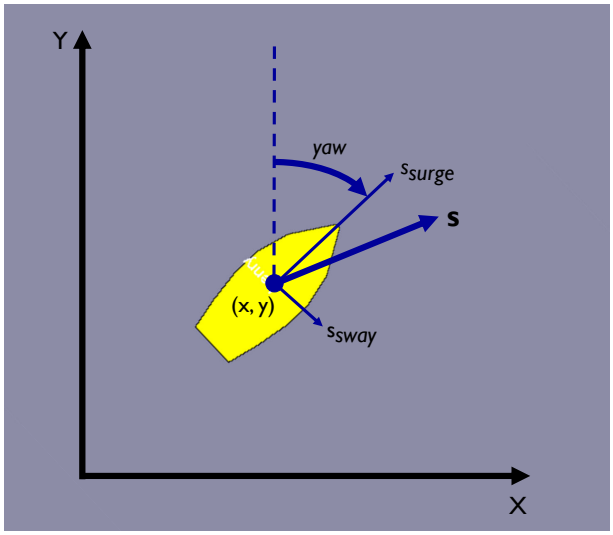

Figure 2: Visualization of the vessel model used in the simulation. The vessel has three degrees of freedom: surge, sway, and yaw (ϕ).

In this work, we focus on IWT, where the hydrodynamic forces are less complex compared to open marine settings. In inland waterways, factors such as roll, pitch and heave play a less significant role due to calmer water conditions and more constrained environments. This is why we employ a Three Degrees of Freedom (3DoF) vessel model that accounts only for surge, sway, and yaw movements. The pose vector at time step t is defined as $\mathbf{p_t} = [x_t, y_t, \phi_t]^T$ where x_t and y_t are the coordinates of the vessel, and ϕ_t is the heading angle of the vessel. The velocity vector at time step t is defined as $\mathbf{v_t} = [s_t, \omega_t]^T$ where s_t is the speed of the vessel and ω_t is the angular velocity of the vessel. In this simulation, the vessel dynamics are updated at each time step (dt = 100 ms) according to Equations 9 and 10. The vessel model is visualized in Figure 2.

$$\mathbf{v_{t+1}} = \mathbf{v_t} + \begin{bmatrix} a \cdot dt \\ u_{rudder} \cdot T \cdot dt \end{bmatrix} \tag{9}$$

$$\mathbf{p_{t+1}} = \mathbf{p_t} + \begin{bmatrix} \sin\phi(s \cdot dt) \\ \cos\phi(s \cdot dt) \\ \omega \cdot dt \end{bmatrix} \tag{10}$$

The acceleration a is computed from the net force acting on the vessel as shown in Equation 11, where m is the mass of the vessel. The control inputs to the vessel are the thrust force F_{thrust} and the rudder angle u_{rudder}. The parameter T represents the turn rate constant that determines the rate at which the

vessel turns. Drag is calculated using the drag equation (Equation 12) (ρ is the density of water, v is the velocity of the vessel, C_D is the drag coefficient and A is the cross-sectional area of the vessel). In the literature, a default drag coefficient between 0.5 and 1 is usually used when the actual drag coefficient is unknown [11]. To conservatively estimate the drag force, we set the drag coefficient to 1.

$$a = \frac{F_{\text{thrust}} - F_{\text{drag}}}{m} \tag{11}$$

$$F_{\text{drag}} = \frac{1}{2}\rho v^2 C_D A \tag{12}$$

4.3 Reinforcement Learning in MOOS-IvP

We utilize the same state representation as used previously in [9]. The observation space consists of the vessel's odometry, Line-of-Sight (LoS) path-following errors [4], distance sensor readings to detected obstacles, coordinates of the intermediate waypoints and the previous control action. The action space is a 2-dimensional vector consisting of continuous control signals for thrust force $[0, 1]$ and rudder angle $[-1, 1]$. The reward function consists of four components, each designed to encourage specific behaviors:

$$\begin{aligned} R_{path} &= \exp((-|e_x| + |e_y| + |e_\phi|)) - 1 \\ R_{distance} &= \frac{d_{t-1} - d_t}{v} \text{ if } v \neq 0 \text{ else } 0 \\ R_{speed} &= -0.1 + speed_t \\ R_{end} &= \begin{cases} 1000 & \text{if goal reached} \\ -1000 & \text{if collision} \end{cases} \end{aligned} \tag{13}$$

R_{path} penalizes deviation from the planned path using the line-of-sight path following errors [4]. Specifically, it incorporates cross-track error (e_x), along-track error (e_y), and heading error (e_ϕ) between the vessel's current position and the desired path. The $R_{distance}$ component rewards progress toward the goal waypoint. It calculates the difference between the previous distance to the goal (d_{t-1}) and the current distance (d_t). When the vessel moves toward the goal, this component yields a positive reward; conversely, when moving away, it produces a negative reward. Additionally, R_{speed} encourages the vessel to move as fast as safely possible, where $speed_t$ is the speed of the vessel at time step t normalized to the maximum speed of the vessel. We noticed that without this component, the agent would sometimes learn to stand still indefinitely to avoid collisions. Finally, R_{end} provides a strong terminal reward or penalty based on episode outcome. In this case, successfully reaching the goal results in a large positive reward, while collisions incur a severe penalty. The final reward is calculated as a weighted (with G_1, G_2, G_3) sum of these components:

$$R(s_t, a_t) = G_1 \cdot R_{path} + G_2 \cdot R_{distance} + G_3 \cdot R_{speed} + R_{end} \tag{14}$$

5 Results

5.1 Robustness

For our first experiment, we tested the robustness of the BC and CQL algorithms for varying mass and turn rate values. We trained the agents utilizing the dataset $D_{\pi_{expert}}$, which was collected from a fully trained SAC agent. The dataset was collected in the nominal environment, with the ego vessel having a mass of 175,000 kg and a turn rate of 70. To test the robustness of our algorithms, we altered these vessel parameters during evaluation. We compared the performance of the BC and CQL algorithms against the SAC policy across these varying conditions. An episode is considered successful if the vessel reaches the goal waypoint in less than 1000 time steps without colliding with any obstacles. The success rate is calculated as the number of successful episodes divided by the total number of episodes. As shown in Figure 3, in the nominal environment, both BC and CQL learn policies that closely match the success rate of the expert (SAC) policy. Notably, we observe that the offline approaches demonstrate robustness comparable to the online approach. They are able to maintain performance consistent with the expert policy across varying mass and turn rate values.

For BC, which minimizes the Mean Squared Error (MSE) between predicted and expert actions given observations, our trained policy achieved a MSE of 0.0005. Considering our action values range between -1 and 1, this indicates that the BC policy has effectively become functionally equivalent to the expert policy, explaining its robust performance across varying vessel parameters. For CQL, the interpretation

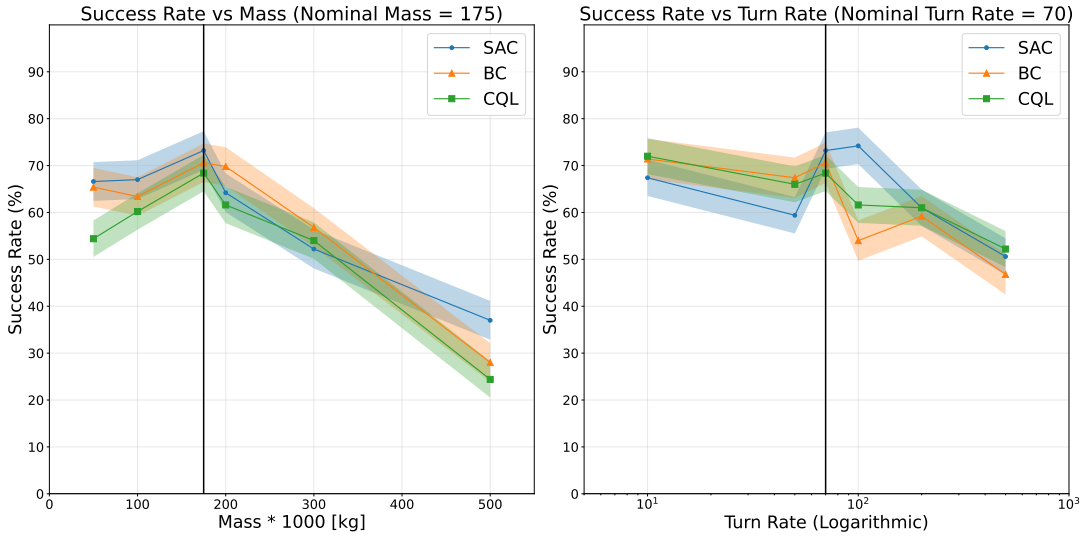

Figure 3: Success rate of the SAC, BC and CQL algorithms for varying mass and turn rate values. Only one parameter was varied at a time. Evaluation was done over 500 episodes. The nominal value is indicated by the bold line. The shaded areas represent the 95% confidence interval.

of loss values is more complex due to its sophisticated objective function (see Section 3.2) that includes a conservative regularization term. This term acts as a safety mechanism that prevents the algorithm from making overly optimistic predictions about actions it hasn't seen in the dataset. Effectively, this makes the policy more cautious about trying actions it hasn't seen before. However, our results demonstrate that with a sufficiently comprehensive dataset, CQL also approximates the expert policy. The slightly lower base performance of CQL compared to BC can be attributed to this inherent conservative nature or potential hyperparameter sensitivity. On Figure 3, we notice the biggest performance decrease when increasing the mass of the vessel. When the mass is increased, the vessel accelerates and decelerates more slowly. This reduced responsiveness makes it harder for the vessel to decelerate or stop in time when encountering obstacles, resulting in a notable increase in static collisions, as shown in Figure 4. Meanwhile, dynamic collisions remain relatively consistent across higher mass values because the dynamic obstacles now move faster than the ego vessel. Furthermore, the overall slower movement can extend episode durations, sometimes causing the vessel to fail to reach the goal within the predefined horizon. When the mass is decreased, the vessel accelerates and decelerates faster. As a result, the vessel may accelerate too quickly, making it difficult for the agent to maintain control and leading to an increase in both static and dynamic collisions. Figure 5 shows the impact of varying the vessel's turn rate on collisions. Decreasing the turn rate below nominal values maintains stable collision rates across all algorithms, indicating robustness to slower turning capabilities. In contrast, increasing the turn rate increases both static and dynamic collisions. While higher turn rates improve vessel agility, they also make control more difficult for agents trained on lower turn rates. This increased sensitivity can lead to oversteering and oscillation, resulting in more frequent collisions.

5.2 Impact of Dataset Quality

Both BC and CQL achieve success rates comparable to the expert policy, even under varied vessel dynamics. This strong performance of BC, a simpler algorithm, naturally leads to the question of why one would opt for the more complex CQL. As highlighted in Section 3.2, BC's objective is to imitate the behavior policy π_D from the dataset, irrespective of π_D's optimality. A possible benefit of CQL is that it offers the potential to learn a policy superior to π_D when it is suboptimal. To test this hypothesis, we trained the BC and CQL algorithms on an additional randomized dataset and a medium-quality dataset. The policies learned from these datasets were evaluated in the nominal environment, with the results presented in Table 1. When trained on the medium-quality dataset $D_{\pi_{medium}}$, CQL achieves a success rate of 31.5%, precisely matching the dataset performance. Similarly, when trained on the expert dataset $D_{\pi_{expert}}$, both BC and CQL achieve success rates comparable to or slightly better than the original expert policy (70.6% for BC and 68.4% for CQL versus 66.3% for the expert policy). However, neither

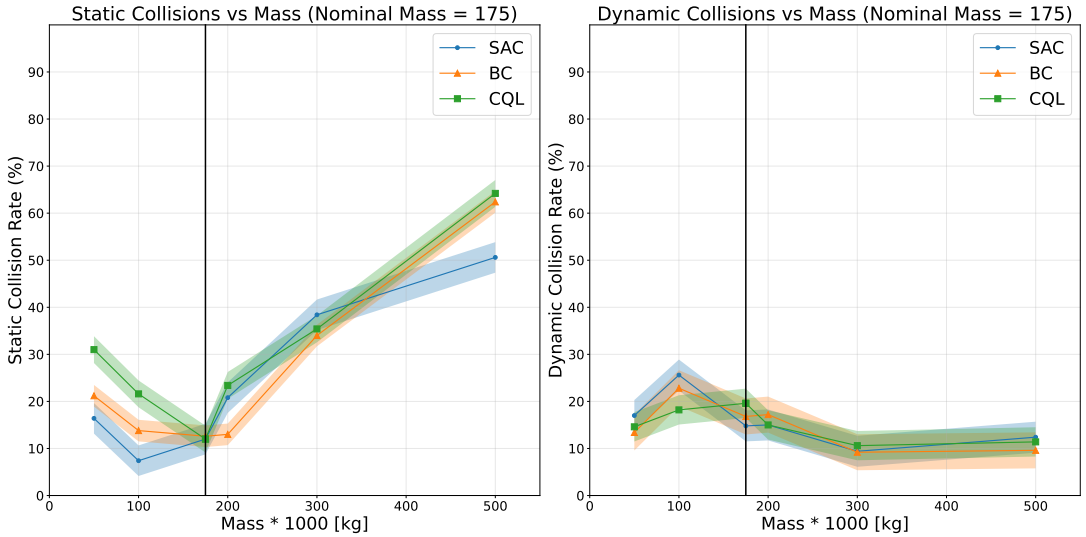

Figure 4: Static and dynamic collision rates of the BC, CQL, and SAC algorithms when varying vessel mass. Evaluations were performed over 500 episodes. The nominal mass value is indicated by a bold line. Shaded areas represent 95% confidence intervals.

algorithm demonstrates significant improvement over the dataset policy they were trained on. This suggests that in our autonomous vessel navigation task, the theoretical advantage of CQL in learning a policy superior to the behavior policy is not realized. The random dataset $D_{\pi_{random}}$ results further emphasize the limitations of offline methods when the dataset lacks sufficient coverage of high-reward state-action pairs, with both algorithms achieving near-zero success rates. The similar performance of BC and CQL may be attributed to the large dataset size. With 500,000 samples in our dataset, most situations encountered during evaluation are well-represented. This extensive coverage provides sufficient examples for BC to effectively clone the expert policy without suffering from the lack of generalization that typically affects imitation learning [19].

Method	**Success Rate (%)**		
	Random	**Medium**	**Expert**
Dataset	0.48	31.5	66.3
BC	0.4	32.9	70.6
CQL	0.1	31.5	68.4

Table 1: Success rates of different methods across datasets collected under three different policies, π_{random}, π_{medium} and π_{expert}. The dataset column shows the performance of the original data, while BC and CQL show the performance of the learned policies. Evaluation was done over 500 episodes.

6 Discussion

6.1 Conclusion

This paper investigated the robustness of offline RL algorithms for autonomous vessel navigation, comparing the performance of BC and CQL against an online SAC baseline. Our key findings challenge common assumptions about offline RL and provide valuable insights for the development of autonomous shipping systems. As seen on Figure 3, 4 and 5, both BC and CQL can achieve performance comparable to the expert (SAC) policy. However, as Table 1 highlights, the performance of these offline methods is significantly lower when trained on suboptimal datasets. This emphasizes the importance of dataset quality and coverage in offline RL. Contrary to common belief in the literature [17], we found that these offline approaches exhibit robustness similar to the online SAC baseline when tested under varying vessel dynamics, specifically changes in mass and turn rate. This suggests offline RL methods are viable alternatives for autonomous vessel navigation, particularly when high-quality demonstration data is available.

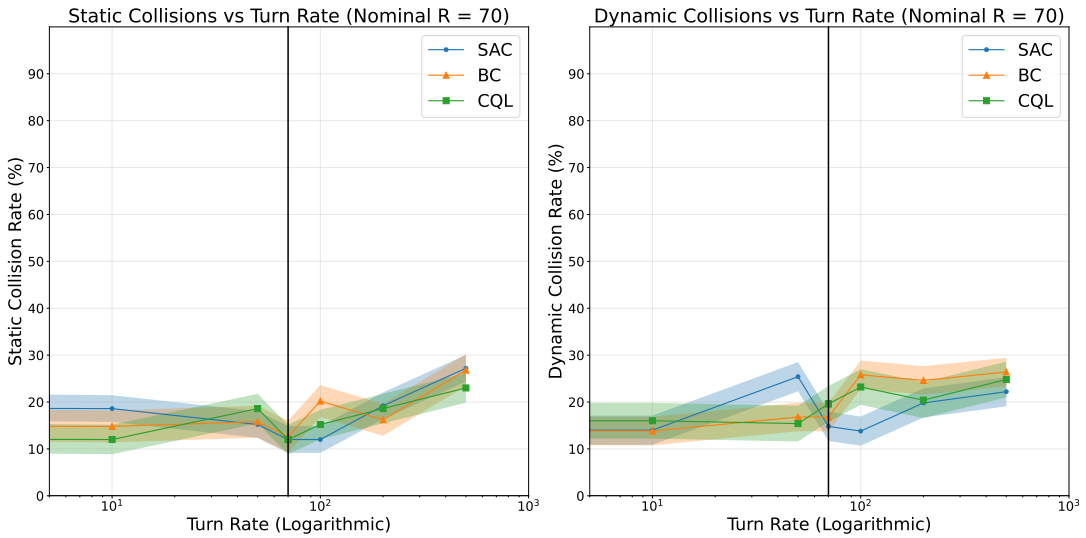

Figure 5: Static and dynamic collision rates of the BC, CQL, and SAC algorithms when varying vessel turn rate. Evaluations were performed over 500 episodes. The nominal turn rate value is indicated by a bold line. Shaded areas represent 95% confidence intervals.

The more complex CQL did not demonstrate significant advantages over BC in our experiments. This held true even when training on suboptimal datasets, where both algorithms achieved similar performance levels. For well-defined navigation tasks with enough training data, simpler imitation learning approaches may suffice. These findings have important implications for the development of autonomous shipping systems. They suggest that offline approaches such as BC can be effective for vessel navigation tasks when provided with high-quality demonstration data. This may simplify the development and deployment of autonomous shipping solutions, reducing the need for complex online training procedures while maintaining robust performance.

6.2 Future Work

Building on our findings, we have identified several promising directions for future research. First, we plan to investigate how dataset size and coverage affect algorithm performance, particularly focusing on whether CQL's theoretical advantages emerge more clearly with smaller datasets where BC typically struggles with generalization. This investigation would help establish minimum dataset requirements for effective offline learning of autonomous navigation policies. Second, we aim to expand our robustness analysis by including additional vessel parameters such as drag coefficient and cross-sectional area. This extension would enable simulation of vessels with varying hull shapes and sizes, providing insights into the algorithms' adaptability across different vessel types. Furthermore, comparing the robustness of our current approaches with robust RL algorithms such as Robust Adversarial Model-based Offline Reinforcement Learning (RAMBO) [20] and Robust Model-Based Policy Optimization (RMBPO) [8] could be a valuable direction for future research. These algorithms are specifically designed to handle environmental variations and uncertainties, and such a comparison might reveal whether specialized robust approaches significantly outperform standard methods for vessel navigation. Another important aspect of robustness is the ability to handle increased environmental complexity and traffic density. Future work should investigate whether policies trained in sparse environments can maintain performance when faced with more complex scenarios containing a higher number of dynamic obstacles and vessels. This would provide valuable insights into the generalization capabilities of these algorithms across varying levels of navigational difficulty. Additionally, these experiments may be replicated in ASVSim (AirSim for Surface Vehicles) [16], an open-source vessel simulator we have recently developed. Lastly, our research group has collected data using a real-world USV. By training and validating our algorithms with this real-world data, we can assess their practical applicability and robustness under actual operating conditions.

7 Acknowledgements
This work is conducted within the DEFRA AHOI project, funded by the Belgian Royal Higher Institute for Defence, under contract number 23DEFRA002.

References
[1] Michael R. Benjamin, Henrik Schmidt, Paul M. Newman, and John J. Leonard. Nested autonomy for unmanned marine vehicles with moos-ivp. *Journal of Field Robotics*, 27(6):834–875, 2010.

[2] CCNR. International definition of levels of automation in inland navigation. `https://www.ccr-zkr.org/files/documents/AutomatisationNav/DefinitionAutomatisation_en.pdf` (accessed 21 March 2025), 2022.

[3] European Commission. *Assessment of potential of maritime and inland ports and inland waterways and of related policy measures, including industrial policy measures – Final report.* Publications Office, 2020.

[4] Thor I. Fossen, Morten Breivik, and Roger Skjetne. Line-of-sight path following of underactuated marine craft. *IFAC Proceedings Volumes*, 36(21):211–216, September 2003.

[5] Dylan J Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. *arXiv preprint arXiv:2407.15007*, 2024.

[6] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[7] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[8] Siemen Herremans, Ali Anwar, and Siegfried Mercelis. Robust Model-Based Reinforcement Learning with an Adversarial Auxiliary Model, July 2024.

[9] Siemen Herremans, Ali Anwar, Arne Troch, Ian Ravijts, Maarten Vangeneugden, Siegfried Mercelis, and Peter Hellinckx. Autonomous port navigation with ranging sensors using model-based reinforcement learning. In *Volume 5: Ocean Engineering*, OMAE2023. American Society of Mechanical Engineers, June 2023.

[10] Zhang-Wei Hong, Aviral Kumar, Sathwik Karnik, Abhishek Bhandwaldar, Akash Srivastava, Joni Pajarinen, Romain Laroche, Abhishek Gupta, and Pulkit Agrawal. Beyond uniform sampling: Offline reinforcement learning with imbalanced datasets. *Advances in Neural Information Processing Systems*, 36:4985–5009, 2023.

[11] International Towing Tank Conference (ITTC). Resistance and propulsion test and performance prediction with skin frictional drag reduction techniques. `https://www.ittc.info/media/8009/75-02-02-03.pdf`, 2017. Accessed: 2025-04-30.

[12] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[13] Hyo-Gon Kim, Sung-Jo Yun, Young-Ho Choi, Jae-Kwan Ryu, and Jin-Ho Suh. Collision avoidance algorithm based on colregs for unmanned surface vehicle. *Journal of Marine Science and Engineering*, 9(8):863, 2021.

[14] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

[15] Bavo Lesy, Ali Anwar, and Siegfried Mercelis. Evaluating robustness of reinforcement learning algorithms for autonomous shipping. In *2024 4th International Conference on Robotics, Automation and Artificial Intelligence (RAAI)*, pages 370–374, 2024.

[16] Bavo Lesy, Siemen Herremans, Robin Kerstens, Jan Steckel, Walter Daems, Siegfried Mercelis, and Ali Anwar. ASVSim (AirSim for Surface Vehicles): A High-Fidelity Simulation Framework for Autonomous Surface Vehicle Research. *arXiv preprint arXiv:2506.22174*, 2025.

[17] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[18] Ülkü Öztürk, Melih Akdağ, and Tarık Ayabakan. A review of path planning algorithms in maritime autonomous surface ships: Navigation safety perspective. *Ocean Engineering*, 251:111010, May 2022.

[19] Allen Ren, Sushant Veer, and Anirudha Majumdar. Generalization Guarantees for Imitation Learning. In *Proceedings of the 2020 Conference on Robot Learning*, pages 1426–1442. PMLR, October 2021.

[20] Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.

[21] Einvald Serigstad, Bjørn-Olav H. Eriksen, and Morten Breivik. Hybrid collision avoidance for autonomous surface vehicles. *IFAC-PapersOnLine*, 51(29):1–7, 2018. 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2018.

[22] Siqing Sun, Tianbo Li, Xiao Chen, Huachao Dong, and Xinjing Wang. Cooperative defense of autonomous surface vessels with quantity disadvantage using behavior cloning and deep reinforcement learning. 164:111968.

[23] Yihao Sun. Offlinerl-kit: An elegant pytorch offline reinforcement learning library. `https://github.com/yihaosun1124/OfflineRL-Kit`, 2023.

[24] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018.

[25] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[26] Astrid Vanneste, Simon Vanneste, Olivier Vasseur, Robin Janssens, Mattias Billast, Ali Anwar, Kevin Mets, Tom De Schepper, Siegfried Mercelis, and Peter Hellinckx. Safety aware autonomous path planning using model predictive reinforcement learning for inland waterways. In *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, 2022.

[27] Martin Waltz, Niklas Paulig, and Ostap Okhrin. 2-level reinforcement learning for ships on inland waterways: Path planning and following. *Expert Systems with Applications*, 274:126933, 2025.

[28] Xinwei Wang, Jie Liu, Haijun Peng, Xiwang Qie, Xudong Zhao, and Chen Lu. A Simultaneous Planning and Control Method Integrating APF and MPC to Solve Autonomous Navigation for USVs in Unknown Environments. *Journal of Intelligent & Robotic Systems*, 105(2):36, June 2022.

[29] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[30] Jing Zhang, Linjiajie Fang, Kexin Shi, Wenjia Wang, and Bingyi Jing. Q-Distribution guided Q-learning for offline reinforcement learning: Uncertainty penalized Q-value via consistency model. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

[31] Xinmin Zhou, Yawei Wu, and Jinxin Huang. Mpc-based path tracking control method for usv. In *2020 Chinese Automation Congress (CAC)*, pages 1669–1673, 2020.

[32] Zexing Zhou, Tao Bao, Jun Ding, Yihong Chen, Zhengyi Jiang, and Bo Zhang. An Offline Reinforcement Learning Approach for Path Following of an Unmanned Surface Vehicle. *Journal of Marine Science and Engineering*, 12(12):2173, December 2024.