Article

A Dynamic Programming Approach to the Collision Avoidance of Autonomous Ships

Raphael Zaccone

Dipartimento di Ingegneria Navale, Elettrica, Elettronica e delle Telecompunicazioni (DITEN), Scuola Politecnica, Università degli Studi di Genova, Via Montallegro 1, 16145 Genova, Italy; raphael.zaccone@unige.it

Abstract: The advancement of autonomous capabilities in maritime navigation has gained significant attention, with a trajectory moving from decision support systems to full autonomy. This push towards autonomy has led to extensive research focusing on collision avoidance, a critical aspect of safe navigation. Among the various possible approaches, dynamic programming is a promising tool for optimizing collision avoidance maneuvers. This paper presents a DP formulation for the collision avoidance of autonomous vessels. We set up the problem framework, formulate it as a multi-stage decision process, define cost functions and constraints focusing on the actual requirements a marine maneuver must comply with, and propose a solution algorithm leveraging parallel computing. Additionally, we present a greedy approximation to reduce algorithm complexity. We put the proposed algorithms to the test in realistic navigation scenarios and also develop an extensive test on a large set of randomly generated scenarios, comparing them with the RRT* algorithm using performance metrics proposed in the literature. The results show the potential benefits of an autonomous navigation or decision support framework.

Keywords: dynamic programming; autonomous ship; path planning; collision avoidance; optimization

MSC: 90C39; 65K05; 90C90

Citation: Zaccone, R. A Dynamic Programming Approach to the Collision Avoidance of Autonomous Ships. *Mathematics* **2024**, *12*, 1546. https://doi.org/10.3390/math12101546

Academic Editor: Abdullah N. Arslan

Received: 28 March 2024
Revised: 5 May 2024
Accepted: 14 May 2024
Published: 15 May 2024

Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The future of navigation points toward increasing the autonomous capabilities of ships [1,2], from decision support systems to fully autonomous navigation. The framework that will guide the future of the maritime world is hinged on the classification released by the International Maritime Organization (IMO) [3], in which four incremental MASS (Maritime Autonomous Surface Ship) autonomy levels are described for the classification of autonomous ships, ranging from MASS Level 1, featuring automated systems for decision support for the crew, to MASS Level 4, an uncrewed ship with full decisional autonomy. The push toward the development of autonomous technologies in the maritime field has created fertile ground for the research community, which has identified numerous scientific gaps in various areas typical of autonomous navigation and marine robotics, such as complex guidance and control algorithms [4–6], collaborative control [7–9], situational awareness [10], path planning, and collision avoidance.

The challenge for maritime collision avoidance is to develop algorithms capable of reacting to the presence of obstacles, fixed or moving in the surrounding environment, and generating trajectories or maneuvers capable of avoiding collision, maintaining an adequate safety distance, and, when applicable, complying with the COLREG [11,12], which sets the "rules of the road" a ship must follow when interacting with other ships. This requirement is essential in scenarios where autonomous systems interact with human-controlled systems [13]. Collision avoidance falls under the broader problem of path planning, i.e., the determination of an optimal path automatically based on information about the surrounding environment. In the robotics field, path planning is commonly divided into two levels [14,15], the off-line or global level, in which the path is determined based on a priori known information, such as information about fixed obstacles or weather forecasts (weather routing), and the local or reactive level, in which subparts of the path are re-planned online in reaction to changes in the environment detected by sensors, such as moving or unexpected obstacles. The scientific literature has proposed various approaches to the reactive collision avoidance of marine vessels, including A* [16], Dijkstra's algorithm [17,18], visibility graphs [19], rapidly exploring random trees [20–22], Artificial Potential Field methods [23,24], and various population-based heuristics [25–28].

Describing a ship's route or maneuver through a sequence of waypoints is a common approach in the literature, not only at the global planning level but also at the reactive planning level. Such an approach features some relevant advantages for the application to large human-crewed ships, both in a fully automatic collision avoidance system and within a MASS Level 1 decision support framework, such as the one Figure 1 illustrates. The representation of a maneuver by waypoints and legs is intelligible to seafarers, and the decision support system can propose it to the officer on the watch, who can understand it and decide whether to acknowledge it. Then, the new sequence of waypoints is taken over by a motion control system, for example, one based on Line of Sight [29,30], which determines, based on GNSS localization, the necessary propulsion and steering actions to track the course according to the ship's dynamics with a reasonably low track error.

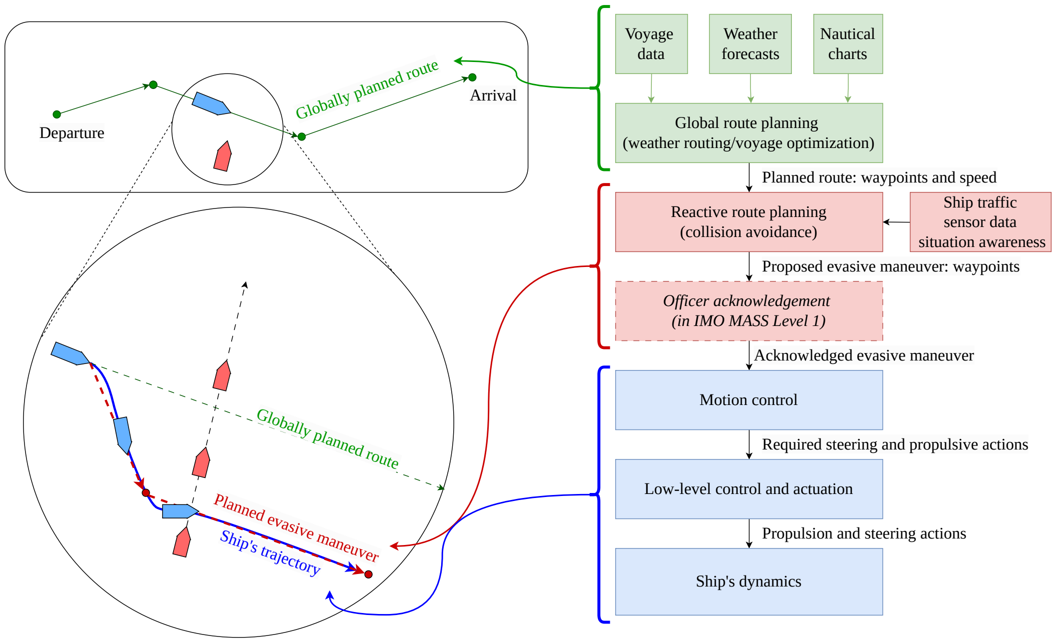

Departure
Arrival
Globally planned route
Globally planned route
Planned evasive maneuver
Ship's trajectory
Voyage data
Weather forecasts
Nautical charts
Global route planning (weather routing/voyage optimization)
Planned route: waypoints and speed
Reactive route planning (collision avoidance)
Ship traffic sensor data situation awareness
Proposed evasive maneuver: waypoints
Officer acknowledgement (in IMO MASS Level 1)
Acknowledged evasive maneuver
Motion control
Required steering and propulsive actions
Low-level control and actuation
Propulsion and steering actions
Ship's dynamics

Figure 1. The layered architecture of an autonomous navigation or navigation support system. The higher layer includes global route planning based on prior known environmental conditions and weather forecasts; the route is represented as a set of waypoints in which the reactive planning layer nests the collision avoidance maneuvers; the alternative maneuvers are actuated by a motion control system, potentially after operator acknowledgment.

Dynamic programming (DP) is an effective approach for solving multi-stage optimization problems. From its introduction by Bellman [31,32], DP has been successfully generalized and formulated to describe path planning problems [33]. The requirements of a collision avoidance system are the ability to effectively and efficiently represent obstacles and the surrounding environment and the ability to determine in real time, with low computational cost, evasive maneuvers that onboard control systems can actuate. These requirements perfectly match the potential of DP, which, in this framework, offers the possibility of a simple and effective problem description, implementation of constraints, and efficient solution schemes capable of exploiting parallelism. Despite the inherent recursive formulation of DP, very efficient solution strategies based on function memoization or tabulation [34] and leveraging parallel computation [35–37] have been proposed to optimize the computation efficiency. DP-based algorithms are ideal for scenarios where resources are limited and real-time computation is crucial, such as in autonomous vehicles, robotics, and embedded systems, and have found application in a broad range of industrial fields, including railway transportation [38], robotics [39,40], and maritime transport. The maritime literature features various applications of DP, primarily in typical maritime global planning problems such as weather routing [41–44], where the objective is to determine the optimal trajectory across a space–time domain to reach the destination by effectively navigating through forecasted dynamic weather conditions in compliance with various constraints including ship motions, seasickness, and minimization of fuel consumption or the ship's motion-related loss function.

In this article, we introduce a DP formulation for the problem presented by the collision avoidance of autonomous vessels which can be applied in a reactive collision avoidance context. In Section 2, we set up the framework to describe the ship collision avoidance problem. In Section 3, we formulate the optimal path planning problem as a multi-stage decision process with a recursive definition based on Bellman's equation. In Section 4, we describe the cost function and constraints based on the actual requirements of a collision avoidance maneuver for marine application, taking into account maneuvering limits and regulations to obtain smooth, collision-free, and COLREG-compliant maneuvers. In Section 5, we propose a bottom-up solution scheme for the problem, leveraging parallel computing, and we discuss some implementation-related strategies in Section 6. Next, in Section 7, we further reduce the algorithm time complexity by proposing a greedy approximation of the DP problem formulated previously. Eventually, in Section 8, we assess the performances of the proposed approach. First, we discuss the results of the proposed algorithms in some relevant navigation scenarios. Moreover, we analyze the performance of the proposed algorithms depending on the algorithm parameters and the required computation effort on a large set of randomly generated scenarios. Finally, we compare the proposed algorithms with the RRT* in randomly generated scenarios using performance metrics proposed in the literature.

2. Collision Avoidance Framework Definition

Despite the Earth's curvature being relevant when planning long routes [41], the horizon of a maneuver in a collision avoidance context is usually such that we can approximate the ship's state space X with a Euclidean plane. Figure 2 visually represents the definitions and notations we will provide in the following.

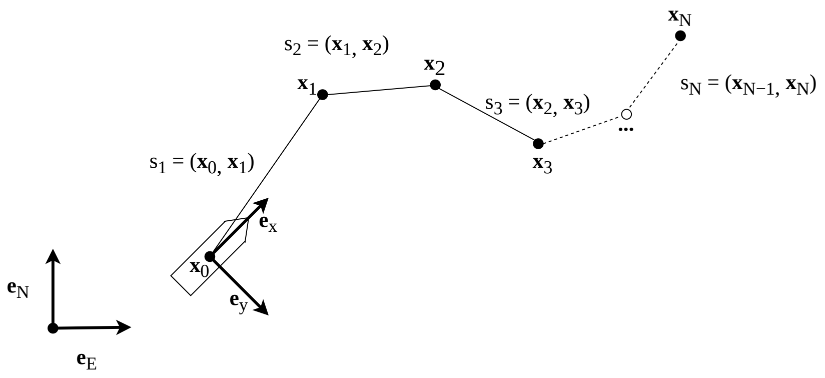

$\mathbf{x}_N$
$s_2 = (\mathbf{x}_1, \mathbf{x}_2)$
$\mathbf{x}_2$
$\mathbf{x}_1$
$s_3 = (\mathbf{x}_2, \mathbf{x}_3)$
$s_N = (\mathbf{x}_{N-1}, \mathbf{x}_N)$
...
$\mathbf{x}_3$
$s_1 = (\mathbf{x}_0, \mathbf{x}_1)$
$\mathbf{e}_x$
$\mathbf{e}_N$
$\mathbf{x}_0$
$\mathbf{e}_y$
$\mathbf{e}_E$

Figure 2. A representation of the ship's configuration space X, including a route R composed of waypoints $\mathbf{x}_i$ and route legs s_i.

At the beginning of the collision avoidance maneuver, the ship is located in $\mathbf{x}_{start}$. We introduce the orthogonal unit vectors $\mathbf{e}_x$ and $\mathbf{e}_y$ to represent the cardinal axes of X; the unit vector $\mathbf{e}_x$ is aligned with the ship's course at the beginning of the collision avoidance maneuver, while $\mathbf{e}_y$ points towards the ship's starboard (right) side so that $\mathbf{e}_z = \mathbf{e}_x \times \mathbf{e}_y$

points downwards, as per the standard convention for ship maneuverability. We will refer to any generic position $\mathbf{x} \in X$ as a waypoint. We can represent a generic maneuver or route R as a sequence of consecutive waypoints as follows:

$$R = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N) = (\mathbf{x}_i)_{i=0}^{N} \quad (1)$$

A sequence of two consecutive waypoints, $s = (\mathbf{x}, \mathbf{y})$, is called a route leg. If $s_i = (\mathbf{x}_{i-1}, \mathbf{x}_i)$, and we denote with "$\oplus$" the sequence concatenation operator such that $(\mathbf{x}, \mathbf{y}) \oplus (\mathbf{y}, \mathbf{z}) = (\mathbf{x}, \mathbf{y}, \mathbf{z})$, we can represent the route R as:

$$R = (\mathbf{x}_0, \mathbf{x}_1) \oplus (\mathbf{x}_1, \mathbf{x}_2) \oplus \ldots \oplus (\mathbf{x}_{N-1}, \mathbf{x}_N) = s_1 \oplus s_2 \oplus ... \oplus s_N \quad (2)$$

We introduce the notation $\vec{s} = \mathbf{y} - \mathbf{x}$ to represent the vector connecting the start point of the leg to the end point. We can obtain the course change between two consecutive legs $s_i \in S_i$, $s_j \in S_j$, $S_i, S_j \subseteq \{(\mathbf{x}, \mathbf{y}) | \mathbf{x}, \mathbf{y} \in X\}$ using the function θ: $S_i \times S_j \to [0, \pi]$ defined as follows:

$$\theta(s_i, s_j) = \arccos\left(\frac{\vec{s_i} \cdot \vec{s_j}}{|\vec{s_i}||\vec{s_j}|}\right) \quad (3)$$

Some considerations about the ship's kinematics are also needed. Firstly, we assume the ship is moving at a constant speed u. Such an assumption is reasonable since, in maritime practice, course alterations are preferred to speed reductions due to the long transients a speed alteration takes. In particular, it is common practice to avoid obstacles by altering the ship's course and trying to keep the speed or reducing the speed by a limited amount in the first part of the maneuver, keeping it constant and slowly regaining the cruise speed after the collision risk is mitigated. Leveraging the constant speed assumption, we can easily map R to a sequence of time instants $(t_i)_{i=0}^{N}$. The instant t_i at which it engages the waypoint $\mathbf{x}_i$ is given by:

$$\begin{cases} t_i = t_{i-1} + \dfrac{|\vec{s_i}|}{u} \\ t_0 = 0 \end{cases} \quad (4)$$

Therefore, the instantaneous vessel position $\mathbf{x}(t)$ at time $t = t_i + \Delta t$ is given by:

$$\mathbf{x}(t_i + \Delta t) = \mathbf{x}_i + u \frac{\vec{s_i}}{|\vec{s_i}|} \Delta t \quad (5)$$

Moreover, since collision avoidance is about the interaction with obstacles, we need to introduce a notation and some hypotheses to describe them. First, we assume there are M obstacles in the scenario with known kinematics. We denote with $\mathbf{a}_m(t) \in X$ the instantaneous position of the mth obstacle. For this study, we will approximate the motion of the obstacles as a straight line with constant speed motion as follows:

$$\mathbf{a}_m(t) = \mathbf{a}_m(t_0) + \mathbf{w}_m t \quad (6)$$

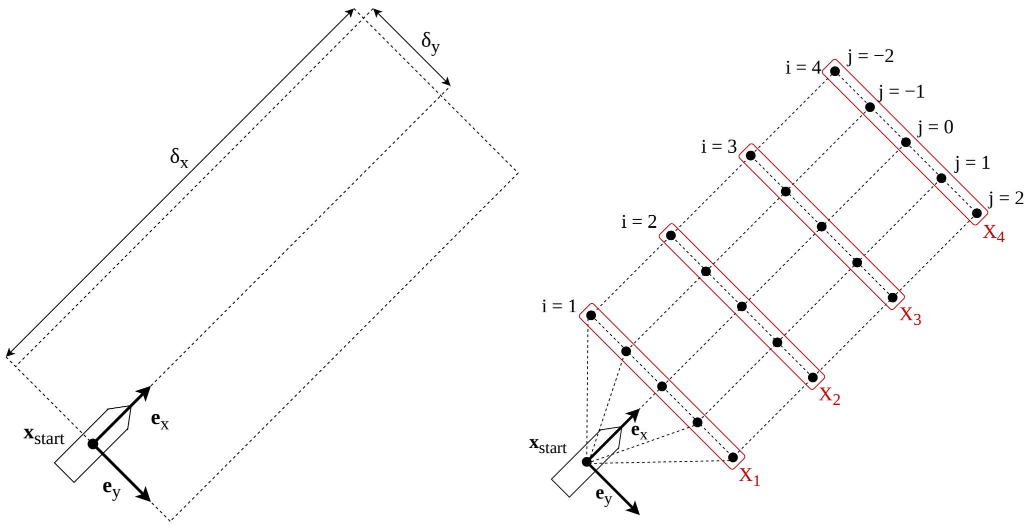
where $\mathbf{w}_m$ represents the speed vector of the mth obstacle. The proposed formulation could be generalized to any known motion law.

3. Dynamic Programming Formulation

Concerning Figure 3a, let $\mathbf{x}_{start} \in X$ be the initial position of the ship, the unit vector $\mathbf{e}_x$ be aligned with the ship's initial course, and $\mathbf{e}_y$ be a unit vector orthogonal to $\mathbf{e}_x$ such that $\mathbf{e}_z = \mathbf{e}_x \times \mathbf{e}_y$ points downwards. Let δ_x and δ_y be the dimensions of the region in which the ship can maneuver. We introduce a discretization of the domain by defining $X_0 = \{\mathbf{x}_{start}\}$ and $X_i \subset X, i \in \{1, ..., N\}$ as follows:

$$X_i = \left\{\mathbf{x} \in X | \mathbf{x} = \mathbf{x}_{start} + \frac{i}{N}\delta_x \mathbf{e}_x + \frac{j}{D}\delta_y \mathbf{e}_y,\ j \in \{-D, \ldots D\}\right\} \quad (7)$$

where $N, D \in \mathbb{N}$ control the fineness of the discretization along $\mathbf{e}_x$ and $\mathbf{e}_y$, respectively, and i, j are the indices of the discretization. Figure 3b exemplifies a graphical representation. A route $R = (x_i)_{i=0}^{N}$ is such that $x_i \in X_i$.



(**a**) Domain region (**b**) Discretized domain region

Figure 3. A representation of the domain of the collision avoidance problem (**a**) and an example discretization of the domain with $N = 4$ and $D = 2$ (**b**).

We can imagine a route $R = s_1 \oplus s_2 \oplus ... \oplus s_N$ as the result of a sequence of transitions between consecutive route legs $s_i = (\mathbf{x}_{i-1}, \mathbf{x}_i) \in S_i$, where $S_i = \{s = (\mathbf{x}, \mathbf{y}) | \mathbf{x} \in X_{i-1}, \mathbf{y} \in X_i\}$. In this framework, optimal route planning can be represented as finding an optimal policy in a multi-stage decision process in which the route legs represent the states. Let J: $\prod_{i=1}^{k} S_i \to \mathbb{R}$ be a function expressing the cost of a route up to state s_k, $k \in \{1, ..., N\}$; therefore, $J(s_1, s_2, ..., s_N)$ represents the cost of the whole route. Let us assume that J is additively separable, i.e., there exists a function c: $S_{i-1} \times S_i \to \mathbb{R}$ such that

$$J(s_1, s_2, ..., s_i) = J(s_1, s_2, ..., s_{i-1}) + c(s_{i-1}, s_i) \tag{8}$$

Let us denote $F(s_i)$: $S_i \to \mathbb{R}$, the route's cost up to the state s_i following an optimal policy, i.e., a policy such that $J(s_1, s_2, ..., s_i)$ is the minimum, as follows:

$$F(s_i) = \min_{s_1, s_2, ..., s_{i-1}} J(s_1, s_2, ..., s_i) \tag{9}$$

Since J is additively separable, we can leverage Bellman's equation to define F recursively [33]:

$$\begin{cases} F(s_i) = \min\limits_{z \in S_{i-1} | t(z, s_i)} c(z, s_i) + F(z) \\ F(s_1) = c(s_0, s_1) \end{cases} \tag{10}$$

where s_0 represents the state of the ship before the collision avoidance maneuver begins, and t: $S_{i-1} \times S_i \to \{True, False\}$ is a function such that $t(s_{i-1}, s_i) = True$ if the transition from s_{i-1} to s_i is feasible, $False$ otherwise.

Similarly, we can define the optimal predecessor function p: $S_i \to S_{i-1}$ returning the optimal predecessor state in S_{i-1} for the state $s_i \in S_i$:

$$p(s_i) = \operatorname*{argmin}_{z \in S_{i-1} | t(z, s_i)} c(z, s_i) + F(z) \tag{11}$$

Let the recursive function B: $S_i \to \{(\mathbf{x}_k)_{k=0}^{i} | x_k \in S_k\}$ have the following form:

$$\begin{cases} B(s_i) = B(p(s_i)) \oplus s_i \\ B(s_1) = s_1 \end{cases} \tag{12}$$

Once we know $p(s_i)$ for all $i \in \{1, ..., N\}$, the optimal solution R can be obtained, backtracking the optimal decisions as follows:

$$R = B(s_N) \tag{13}$$

where

$$s_N = \underset{s \in S_N}{\operatorname{argmin}} F(s) \tag{14}$$

4. Constraints and Cost Function

The constraints are needed to formulate the function t: $S_{i-1} \times S_i \to \{True, False\}$ to identify whether a transition is feasible. For the transition to be feasible, the next leg must begin at the end of the previous. Moreover, each transition must be such that the final resulting sequence, i.e., the route, is compatible with the ship's maneuvering capabilities, respects the COLREG, and is collision safe. Therefore, the function t takes the following form:

$$t(s_{i-1}, s_i) = t_{link}(s_{i-1}, s_i) \wedge t_\theta(s_{i-1}, s_i) \wedge t_{COLREG}(s_i) \wedge t_{collision}(s_i) \tag{15}$$

In particular, t_{link} ensures that the two route legs can be connected. The course change constraint function t_θ ensures the course changes along the path are compatible with ship maneuvering capabilities and good seamanship, the COLREG constraint function t_{COLREG} implements a set of rules to ensure the COLREG compliance of the own ship's kinematics relative the other vessels', and the collision avoidance constraint function $t_{collision}$ aims to guarantee an appropriate distance from the obstacles to avoid collisions safely.

4.1. Route Leg Connection

The feasibility of the transition from $s_{i-1} \in S_{i-1}$ to $s_i \in S_i$ is expressed by the function t_{link}: $S_{i-1} \times S_i \to \{True, False\}$. The transition is possible if the end point of the first leg is the start point of the second:

$$t_{link}(s_{i-1} = (\cdot, \mathbf{x}), s_i = (\mathbf{y}, \cdot)) = (\mathbf{x} \overset{?}{=} \mathbf{y}) \tag{16}$$

where the symbol "$\cdot$" is a placeholder for any point in the proper domain, and the operator "$\overset{?}{=}$", returning values in $\{True, False\}$, is such that $A \overset{?}{=} B$ returns $True$ if $A = B$, $False$ otherwise.

4.2. Course Change Constraint

We need to define a minimum and maximum threshold value for the ship's course changes, denoted as $\theta_{\min}$ and $\theta_{\max}$, respectively. The introduction of $\theta_{\max}$ is motivated by the ship's maneuvering capabilities and the performance of the motion control system. During a course change, the ship goes off the predefined track by a certain distance which is related to the magnitude of the course change. A large course change will push the ship too much off track, increasing the risk of collision. The reason for $\theta_{\min}$ can be found in COLREG Rule 8(b), which requires that "Any alteration in course and/or speed to avoid collision must, if the circumstances of the case permit, be large enough to be readily apparent to another vessel observing visually or by radar; a succession of small alterations in course and/or speed must be avoided". We thus want to ensure that we either have course changes large enough to be apparent to an observer but not so large that the control and steering system cannot actuate them or no course change between legs. Thus, we can define t_θ: $S_{i-1} \times S_i \to \{True, False\}$ as:

$$t_\theta(s_{i-1}, s_i) = (\theta_{\min} \leq \theta(s_{i-1}, s_i) \leq \theta_{\max}) \vee \left(\theta(s_{i-1}, s_i) \overset{?}{=} 0\right) \tag{17}$$

4.3. COLREG Compliance Constraint

The COLREG is an international convention regulating several aspects of navigation, including visibility, navigation lights, and required behaviors in encounter situations to avoid collisions. This last part is a set of "rules of the road" that each ship has to follow when encountering other ships. COLREG-compliant behavior can be identified by analyzing the kinematics of the ships engaging in the scenario and relying on a set of if-then-else rules to assess which behavior each ship has to keep relative to the others. This action is usually called COLREG classification. Various approaches and algorithms for COLREG classification have been proposed in the literature [45,46].

For this study, we suppose that, to determine the COLREG-compliant behavior β_i, the own ship must keep relative to the ith dynamic obstacle in the scenario, and that $\beta_i \in CB$, where CB is the enumeration of all the possible behaviors:

$$CB = \{SO, GW, HO, AA\} \tag{18}$$

In particular:

- SO (Stand On): COLREG requires the target ship to maneuver to avoid a collision, so the own ship must keep its course and speed. In this case, we will neglect the presence of the target ship;
- GW (Give Way): The own ship must maneuver to avoid collision with the target ship, letting the latter pass ahead;
- HO (Head On): The own ship and the target ship are sailing on parallel and opposite routes, and the own ship must avoid the collision by turning to starboard (right);
- AA (Any Action): The own ship must take any appropriate action to avoid collision; this behavior is adopted in emergencies, such as when the target ship is expected to maneuver to avoid the collision but does not seem to initiate the evasive maneuver.

These behaviors are ensured in the solution by imposing constraints on the leg in which the own ship and the target ship cross each other's route.

Let us assume that the ship and the target $\mathbf{a}$ cross in leg $s_i = (\mathbf{x}_{i-1}, \mathbf{x}_i)$, i.e., for $t \in [t_{i-1}, t_i]$, and let us denote with $\mathbf{x}_c$ the intersection point. If $\beta = GW$, COLREG requires the target ship to engage $\mathbf{x}_c$ before the own ship; thus, we need to impose that, if the Own ship engages $\mathbf{x}_c$ during leg s_i, it does so after the target:

$$\begin{aligned} t_{GW}&(s_i = (\mathbf{x}_{i-1}, \mathbf{x}_i)) \\ &= (\mathbf{x}_c \in \{\mathbf{x}(t) \in X | t \in [t_{i-1}, t_i]\}) \wedge \left(\frac{|\mathbf{x}_c - \mathbf{x}_{i-1}|}{u} - \frac{|\mathbf{x}_c - \mathbf{a}(t_{i-1})|}{|\mathbf{w}|} > 0\right) \end{aligned} \tag{19}$$

If $\beta = HO$, the correct behavior is ensured by ensuring that the computed path keeps the target ship on the port side of the own ship. The latter is ensured by the constraint below:

$$t_{HO}(s_i) = \forall t \in [t_{i-1}, t_i] : (\mathbf{a}_m(t) - \mathbf{x}(t)) \times \vec{s_i} \cdot \mathbf{e}_z > 0 \tag{20}$$

Eventually, the COLREG constraint function takes the following form:

$$\begin{aligned} t_{COLREG}(s_i) = &\forall m \in \{1, \ldots, M\} : (\beta_m \overset{?}{=} GW \wedge t_{GW}(s_i)) \\ &\vee (\beta_m \overset{?}{=} HO \wedge t_{HO}(s_i)) \\ &\vee (\beta_m \in \{AA, SO\}) \end{aligned} \tag{21}$$

4.4. Collision Avoidance Constraint

The obstacle avoidance constraint aims to guarantee that the evasive maneuver is collision safe. To this end, let us introduce the concepts of safety distance d_{safety} and the Closest Point of Approach CPA. The d_{safety} is the distance that the own ship must maintain from all obstacles during an evasive maneuver. It provides a safety margin that accounts for ship dimensions and uncertainties related to the environment, control, and measurement systems, ensuring the ship's safety during the avoidance maneuver. The CPA is the minimum distance between the center points of the own ship $\mathbf{x}(t)$ and an obstacle, whose position over time is denoted by $\mathbf{a}_m(t)$, for $t \in [t_0, t_N]$. In particular, we define the $CPA_m(s_i)$: $S_i \to \mathbb{R}$ as follows:

$$CPA_m(s_i) = \min_{t\in[t_i-1,t_i]} |\mathbf{a}_m(t) - \mathbf{x}(t)| \tag{22}$$

The ship and obstacles may have different sizes, speeds, and relative positions. In addition, previous results have shown that some encounter scenarios are more critical than others in their dynamic evolution [13]. For the sake of generality, it may thus be appropriate to determine a specific safety distance $d_{safety,m}$ for each obstacle belonging to the scenario. Thus, if there are $M \in \mathbb{N}$ obstacles, the function $t_{collision}$: $S_i \to \{True, False\}$ checks whether a transition is collision safe:

$$t_{collision}(s_i) = \forall m \in \{1, \ldots, M\} \mid \beta_m \neq SO: \; CPA_m(s_i) \geq d_{safety,m} \tag{23}$$

4.5. Cost Function

The definition of the cost function plays an essential role since it directly influences the characteristics of the optimal path. Various approaches can be used, depending on the application. At the global route planning level, cost functions related to travel time or distance are often used to determine the shortest or fastest route. More elaborate approaches feature models of ship response to the environment, including ship motions, to determine a safer or more comfortable route, and ship propulsion, to determine the most fuel-efficient route [41,42,47,48]. Such approaches require more or less accurate modeling of the ship's response to weather conditions and the availability of weather forecast data. Conversely, local planning aims to alter short route segments in reaction to an encounter situation with other ships to ensure a safe distance and avoid collisions. Fuel consumption and comfort relative to ship motions have a minor relevance in executing an evasive maneuver where the time horizon is short and the priority is collision avoidance. While the constraints described in the previous sections are crucial for this purpose, the role of the cost function is to describe the characteristics of a "desirable" route so that the algorithm can choose it. Although the same approaches described for high-level planning are possible, such calculations come at the price of a potentially increased computational effort due to the complexity of the ship response model.

Within the proposed framework, a maneuver, i.e., a series of waypoints, should be simple for an operator or autopilot to execute, with limited course alterations to limit overshoots and prevent the ship from going off track. To this end, we propose a cost function inspired by the concept of minimum control energy. In the context of this paper, control energy is related to the amplitude of the maneuvers needed to follow a path. In other words, a path with high control energy requires the ship to perform large course changes over time. According to this principle, we can define the transition cost c as follows:

$$c(s_{i-1}, s_i) = \theta^2(s_{i-1}, s_i) \tag{24}$$

The choice of a minimum path length cost function would lead to equal ease of evaluation with potentially larger angle variations and, in general, a smaller margin on safety distance, but the guarantee of a shorter and more direct solution.

5. Solution Scheme

This section presents a bottom-up solution scheme for the problem posed in Section 3 based on a tabulation approach. We can divide the algorithm into two phases: the tabulation phase and the backtracking phase.

Algorithm 1 illustrates the tabulation phase. We sequentially generate the feasible states at each stage by leveraging Equation (7), while keeping track of the partial optimum F values and the backward link to the optimal predecessor p, into proper data structures.

Concerning Algorithm 1, we can estimate the number of stages in each state as $(2D+1)^2$, i.e., the total number of transitions to be evaluated at each stage is $(2D+1)^4$. Since the only transitions to be evaluated are those whose initial state ends in the starting point of the final state, a proper implementation allows them to be accessed directly in constant time; thus, the number of evaluated transitions can be reduced to $(2D+1)^3$ in constant time, leading to a time complexity of $O(ND^3)$ for the overall process if the cost and constraints have constant time complexity. It is worth noting that the inner for-loop at line 10 of Algorithm 1 can be run in parallel since the loop does not mutate shared data. Once the tabulation phase is completed, the backtracking phase, described in Algorithm 2, reconstructs the optimal solution based on the previously tabulated back-links returned by the best predecessor function p.

Algorithm 1: Bottom-up solution scheme.

```
1  $S_1 = \emptyset$
2  **for** $j \in [-D, ..., D]$ **:**
3      $s_j = (\mathbf{x}_{start}, \mathbf{x}_{start} + \frac{1}{N}\delta_x \mathbf{e}_x + \frac{j}{D}\delta_y \mathbf{e}_y)$
4      $S_1 = S_1 \cup \{s_j\}$
5      $F(s_j) = c((\mathbf{x}_{start} - \mathbf{e}_x, \mathbf{x}_{start}), s_j)$
6      $p(s_j) = None$
7
8  **for** $i \in [2, \dots, N]$ **:**
9      $S_i = \emptyset$
10     **parallel for** $j \in [-D, \dots, D]$ **:**
11         $\mathbf{x}_j = \mathbf{x}_{start} + \frac{i}{N}\delta_x \mathbf{e}_x + \frac{j}{D}\delta_y \mathbf{e}_y$
12         **for** $k \in [-D, \dots, D]$ **:**
13             $\mathbf{x}_k = \mathbf{x}_{start} + \frac{i-1}{N}\delta_x \mathbf{e}_x + \frac{k}{D}\delta_y \mathbf{e}_y$
14             $s_j = (\mathbf{x}_k, \mathbf{x}_j)$
15             $T = \{z \in S_{i-1} | t(z, s_j)\}$
16             **if** $T \neq \emptyset$ **:**
17                 $S_i = S_i \cup \{s_j\}$
18                 $F(s_j) = \min_{z \in T} c(z, s_j) + F(z)$
19                 $p(s_j) = \mathrm{argmin}_{z \in T} c(z, s_j) + F(z)$
```

Algorithm 2: Backtracking of the optimal solution.

```
1  $R = (\,)$
2  $s = \mathrm{argmin}_{z \in S_N} F(z)$
3  **while** $s \neq None$ **:**
4      $R = s \oplus R$
5      $s = p(s)$
6  **return** $R$
```

6. Implementation

By leveraging a graph structure, we can implement the approach described in Algorithms 1 and 2. We can think of each node of the graph as a data structure $n_{i,j}$, which stores the following for each stage i and for each state j:

- The current state $s_j \in S_i$, which we will refer to with the shorthand notation $s_{i,j}$;
- The optimal cost of the current state $F(s_{i,j})$;
- A back-link to the optimal predecessor $p(s_{i,j})$, implemented as a pointer to the predecessor node.

Algorithm 1 allows us to create all the nodes of the graph starting from the initial stage (lines 2–6) and generating the nodes at stage $i+1$ from stage i by initializing each node's state (lines 11, 13, 14) and then assessing the node's optimal cost and predecessor by solving a simple optimization problem to choose the best predecessor node (lines 15 to 19). Figure 4 shows a section of an example graph structure generated based on Algorithm 1. Each node contains the state $s_j \in S_i$, denoted in shorthand as $s_{i,j}$, the optimal cost $F(s_{i,j})$, and the optimal predecessor $p(s_{i,j})$, i.e., a back-link to the optimal predecessor node represented with an arrow.

When the graph has been completed, we can select the node with the minimum cost (i.e., minimum value of F) at the last stage and then backtrack the optimal solution by following the chain of back-links, concatenating each state $s_{i,j}$ to the front of a sequence according to Algorithm 2 until we reach the starting point.

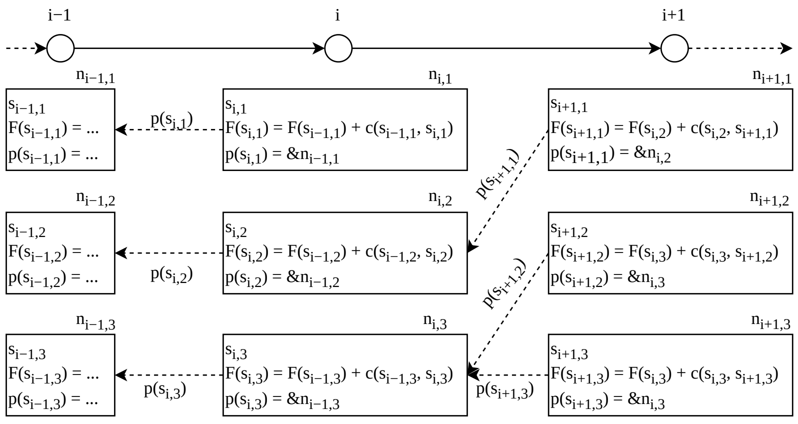


Figure 4. Tabular representation of part of the graph structure, including stages $i-1$, i, and $i+1$. For each stage i and for each state j, each node $n_{i,j}$ stores the state $s_j \in S_i$, denoted with the shorthand notation $s_{i,j}$, the optimal cost F, and a back-link p to its optimal predecessor, represented with an arrow.

7. Greedy Approximation

From Equation (15), we can note that only part of the conditions for the feasibility of the transition from stage $i-1$ to stage i depends on the state s_{i-1}, as the transition cost c defined in Equation (24) does. In this section, we propose a greedy approximate dynamic programming (GADP) scheme that makes greedy decisions to reduce the number of evaluated transitions. The proposed scheme loses the capacity to find the globally optimal policy, yet it allows for the time complexity of the algorithm to be reduced. The basic idea is to reformulate the route $R = (\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N)$ as a sequence of transitions between consecutive states $\mathbf{x}_i \in X_i$, $i \in \{0, 1, ..., N\}$. In other words, we use the waypoints to represent the ship's state rather than representing it with a leg connecting two consecutive waypoints. The greedy minimum cost at stage i is expressed by the function F_g: $X_i \to \mathbb{R}$, while the greedy optimal predecessor is represented by the function p_g: $X_i \to X_{i-1}$:

$$F_g(\mathbf{x}_i) = \min_{\mathbf{z} \in X_{i-1} | t_g(\mathbf{z}, \mathbf{x}_i)} c_g(\mathbf{z}, \mathbf{x}_i) + F_g(\mathbf{z}) \tag{25}$$

$$p_g(\mathbf{x}_i) = \underset{\mathbf{z} \in X_{i-1} | t_g(\mathbf{z}, \mathbf{x}_i)}{\operatorname{argmin}} \quad c_g(\mathbf{z}, \mathbf{x}_i) + F_g(\mathbf{z}) \tag{26}$$

where t_g: $X_{i-i} \times X_i \to \{True, False\}$ and c_g: $X_{i-i} \times X_i \to \mathbb{R}$ greedily compute the result based on p_g: $X_i \to X_{i-1}$:

$$t_g(\mathbf{x}, \mathbf{y}) = t((p_g(\mathbf{x}), \mathbf{x}), (\mathbf{x}, \mathbf{y})) \tag{27}$$

$$c_g(\mathbf{x}, \mathbf{y}) = c((p_g(\mathbf{x}), \mathbf{x}), (\mathbf{x}, \mathbf{y})) \tag{28}$$

Algorithm 3 shows the proposed approach. Since a whole cycle over $[-D, ...D]$ disappears in the solution scheme, we now evaluate only the $(2D+1)^2$ transition at each stage, and the time complexity required to run the algorithm drops from $O(ND^3)$ to $O(ND^2)$.

Algorithm 3: Approximate dynamic programming bottom-up solution scheme.

```
1  $S_0 = \{\mathbf{x}_{start}\}$
2  $F(\mathbf{x}_{start}) = 0$
3  $p(\mathbf{x}_{start}) = None$
4  for $i \in [1, \ldots, N]$ :
5      $S_i = \emptyset$
6      parallel for $j \in [-D, \ldots, D]$ :
7          $\mathbf{x}_j = \mathbf{x}_{start} + \frac{i}{N}\delta_x \mathbf{e}_x + \frac{j}{D}\delta_y \mathbf{e}_y$
8          $T = \{\mathbf{z} \in S_{i-1} | t_g(\mathbf{z}, \mathbf{x}_j)\}$
9          if $T \neq \emptyset$ :
10             $S_i = S_i \cup \{\mathbf{x}_j\}$
11             $F_g(\mathbf{x}_i) = \min_{\mathbf{z} \in T} c_g(\mathbf{z}, \mathbf{x}_i) + F_g(\mathbf{z})$
12             $p_g(\mathbf{x}_i) = \operatorname{argmin}_{\mathbf{z} \in T} c_g(\mathbf{z}, \mathbf{x}_i) + F_g(\mathbf{z})$
```

The backtracking phase works similarly to in the previous case.

8. Results

This section aims to evaluate the performance and capabilities of the proposed algorithms to understand their potential contribution if applied within an autonomous navigation and decision support context. In particular, this section is divided into three parts: In Section 8.1, we present results for three example scenarios which include moving and fixed obstacles. This analysis aims to show, qualitatively and quantitatively, the differences between the proposed algorithms in simple scenarios inspired by practical navigation conditions. The second and third subsections involve extensive tests of the presented algorithms against randomly generated scenarios, including those with fixed and moving obstacles with randomly assigned positions, direction, and speed. We solve one thousand randomly generated scenarios to statistically evaluate the algorithms' performance. In particular, Section 8.2 analyzes the influence of domain discretization on the optimality of the solution, the computation time, and the algorithm failure rate, while Section 8.3 compares DP and GADP with the RRT* algorithm according to performance metrics from the literature. All scenarios take place in a square domain of $D = [0, 10] \times [0, 10]$ nautical miles, in which the own ship starts from point $(0, 0)$ with heading $0°$ to reach the opposite side of the domain, i.e., any point $(10, y) \in D$. An implementation of the two proposed algorithms has been developed for testing and comparison purposes. The algorithms are implemented in Rust language, relying on the Rayon library for parallelization. We perform the computations on an AMD Ryzen 9 5900HS CPU (8×3.3 Hz $-$ 4.6 Hz boost), 32 GB DDR.

8.1. Example Navigation Scenarios

In this subsection, we apply the proposed algorithms to three navigation scenarios. The first features two sets of fixed obstacles extended transversely to the navigation direction, mimicking two docks; the second is a COLREG encounter scenario with sailboats; the third is a head-on scenario within a channel. More details on the analyzed scenarios are given below.

- Scenario 1, shown in Figure 5a, has two barriers placed at $x = 5$ and $x = 9$ nautical miles, respectively, ranging in $[-2.5, 5]$ and $[-5, 2.5]$. These obstacles force the own ship to perform a zig-zag maneuver between the barriers;
- Scenario 2, shown in Figure 5b, features two sailboat vessels, the first starting in $(8.0, 4.5)$ with a speed of 5.0 knots and a heading of 270°, the second positioned at $(4, -2)$ with a speed of 6 knots and a heading of 90°. The COLREG imposes on the own ship a "give-way" behavior against both the target vessels;
- Scenario 3, shown in Figure 5c, features a double "head-on" with two target vessels starting from $(9, 1)$ and $(10, 0)$ with a heading of 180° and speeds of 9 and 8 knots, respectively. In addition, two fixed side barriers form a channel parallel to the x-axis that is 8 nautical miles wide.

To comply with the COLREG, the own ship must make only visible heading alterations, with a minimum angle of 15°, while a maximum of 60° turn is accepted. The algorithms perform the path optimization on a discrete computation grid defined as per Equation (7), where $N = 10$ and $D = 20$.

Figure 6 shows the solutions found by the two algorithms in the three proposed test scenarios. In scenarios 1 and 2 (Figure 6a,b), DP and GADP propose different maneuvers; in particular, the GADP solution features more delayed direction changes. In scenario number 3 (Figure 6c), on the contrary, the solution computed by the two approaches is the same, i.e., the greedy optimum computed by GADP corresponds to the global optimum of the DP. Eventually, Figure 7a,b present the value of the cost function and the computation time required to determine the solution, respectively. We can note that, at the price of a higher cost function value, the solutions determined by the GADP algorithm require less computation time.

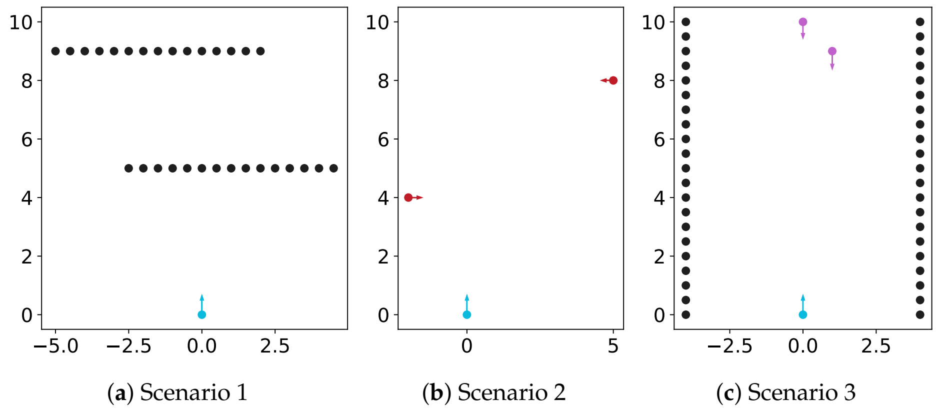
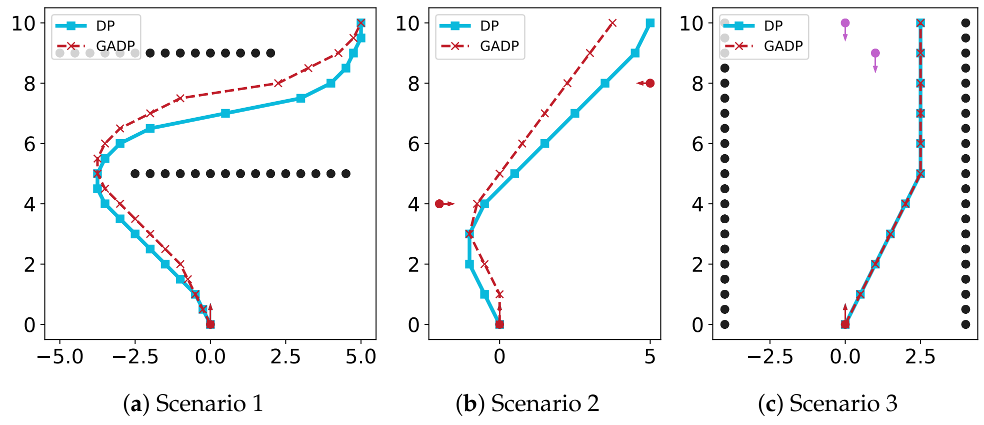
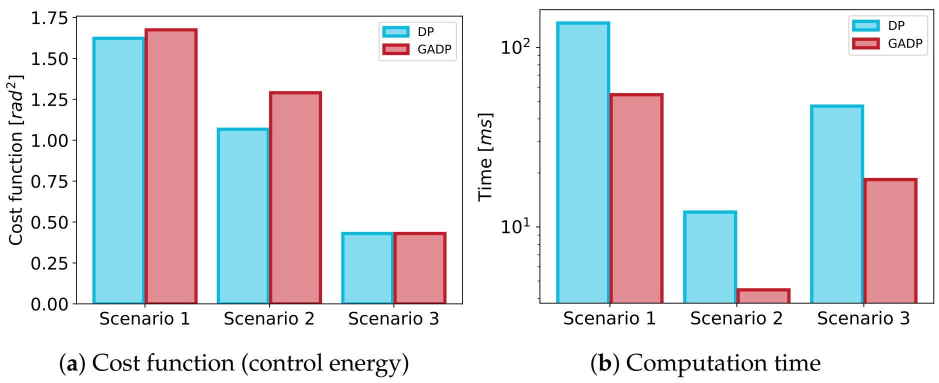


Figure 5. Application case scenarios. Scenario 1 (**a**) includes fixed obstacles only; scenario 2 (**b**) features two sailboats (in red), both requiring "give-way" behavior from the own ship (blue); scenario 3 (**c**) features two head-on ships (magenta) in a narrow channel.



(**a**) Scenario 1 (**b**) Scenario 2 (**c**) Scenario 3

Figure 6. Application case scenarios solved using DP (blue) and GADP (orange).



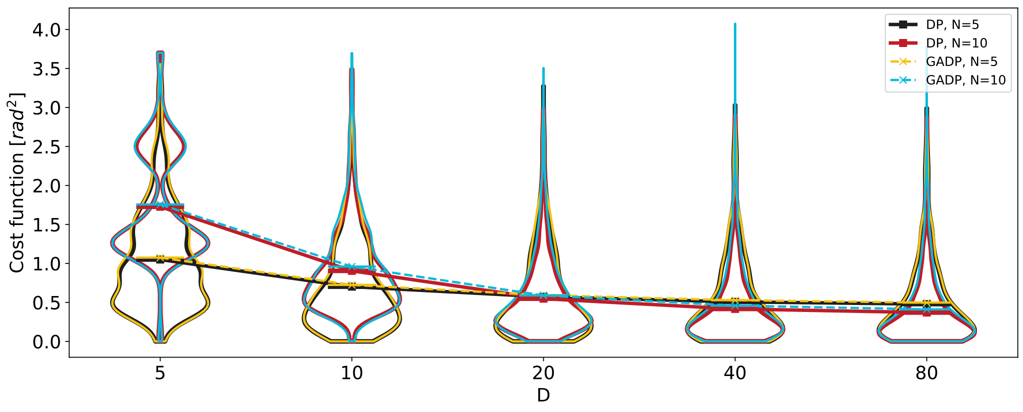
(**a**) Cost function (control energy) (**b**) Computation time

Figure 7. Cost function value (**a**) and computation time (**b**) comparison of the two proposed algorithms relative to the solutions presented in Figure 6.

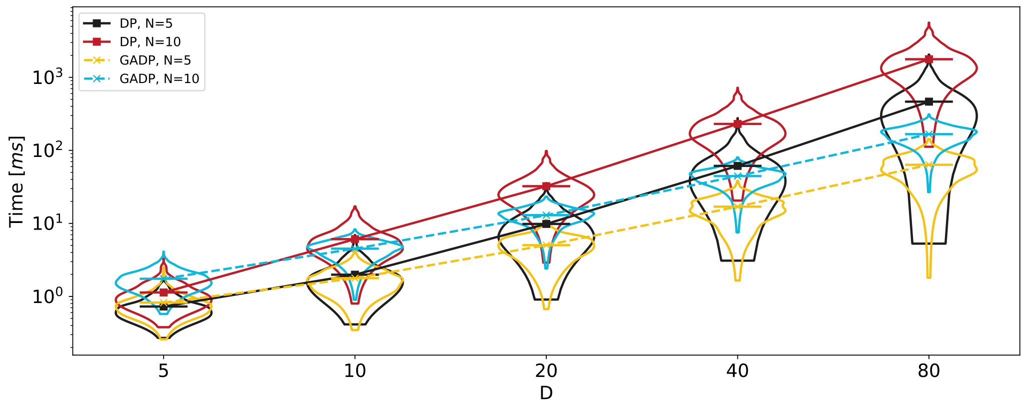
8.2. Effect of the Grid Fineness Parameters

The proposed algorithms determine the best maneuver by selecting a sequence of waypoints on a discrete grid controlled by the parameters N and D. Section 5 discusses how the computational effort depends strictly on these two parameters; however, a too-sparse grid could compromise the optimum quality. To estimate how much the fineness of the grid affects the minimum cost function and computational time, we perform a series of experiments based on the random generation of 1000 scenarios. Each scenario includes up to 10 fixed and 10 moving obstacles, each with a randomly assigned position, direction, and speed. The scenarios thus generated are solved with the DP and GADP algorithms with $N = 5$ and $N = 10$ and D doubling from 5 to 80.

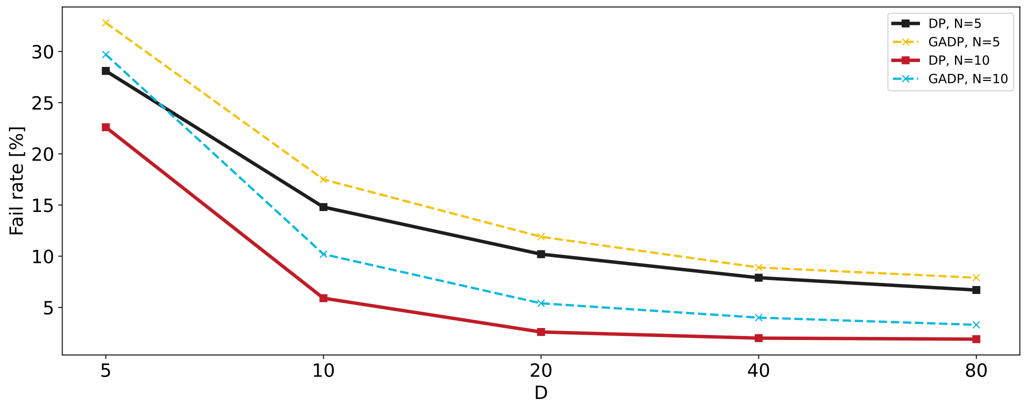
The results are presented in Figure 8 using violin plots [49]. We can observe how, while the grid fineness increases, the cost function reduces to stabilize when we reach a sufficiently fine grid (Figure 8a). Conversely, Figure 8b shows that the computation time increases as expected. We can also note how GADP requires less time than DP and how the greedy optimum tends to approach the exact optimum as the grid becomes dense. Eventually, Figure 8c shows the trend in the percentage of failed random scenarios as the fineness of the grid changes. With a sufficiently high D, the percentage stabilizes to a fraction of the "unsolvable" scenarios, which decreases as N increases. We can also note that the greedy approximated algorithm can solve fewer scenarios. In other words, although a solution exists, the GADP cannot find it due to the consequences of the greedy assumptions made in Section 7 on the transition feasibility.



(**a**) Cost function (control energy)



(**b**) Computation time



(**c**) Failure rate

Figure 8. Cost function value (**a**), computation time (**b**), and failure rate (**c**) as a function of the grid size, estimated in solving a thousand randomly generated scenarios. The distribution of the cost and time values in the dataset is represented using violin plots.

*8.3. Performance Assessment and Comparison with RRT**

This subsection compares the proposed algorithms against a set of randomly generated scenarios according to a set of performance metrics to better highlight the advantages and disadvantages of the proposed algorithms, which we partially discuss in Section 8.1. The scenarios used to perform the test are randomly generated using the same technique as described in Section 8.2, i.e., by arranging in the domain a random number from 1 to 10 of fixed obstacles and moving obstacles with randomly assigned positions, direction, and speed. For completeness, the comparison includes a state-of-the-art algorithm with comparable features and applicability, the RRT* algorithm. The RRT* is a random sampling algorithm designed to explore domains using tree structures and determine collision-free maneuvers quickly. RRTs [50] iteratively generate tree-like structures, initiating from a root node and terminating when a node is close enough to the desired goal. The "*" (star) variant, or Optimal RRT [51], includes local optimizations of the tree topology within the neighborhood of each newly generated node, leveraging a cost function to generate heuristically optimal trajectories. The implementation of the RRT* for comparison includes assumptions similar to those described for the dynamic programming algorithms described in this paper. In addition, our implementation can account for fixed and moving obstacles with constant speed and direction, similar to the DP-based schemes proposed here [21].

The RRT* produces heuristically better solutions the more iterations it runs. However, a large number of iterations leads to a high computation time since the complexity of the RRT* algorithm, if the cost function runs in constant time, is $O(K\log(K))$, where K is the number of iterations [51]. As a stopping criterion, we assume that the algorithm ends when it is possible to reach the target side of the domain from a node closer than the distance used to select the near nodes in the tree optimization. If more than one maneuver is possible, the one with the lowest cost is chosen. Setting a minimum number of nodes in the tree ensures enough iterations to obtain sufficiently optimized paths. Since the computation time depends on the number of iterations, i.e., roughly on the number of newly created nodes, this parameter significantly affects the trade-off between computation time and optimality of the solution. On the other hand, to avoid infinite iteration in dead-end scenarios, a maximum number of nodes after which the algorithm stops is usually set. To diversify the comparison, we consider two values for the minimum number of nodes of the RRT*: 500 nodes (RRT*-500n) and 2000 nodes (RRT*-2000n). In both cases, the algorithm stops if the number of nodes is ten times the minimum.

The four algorithms, DP, GADP, and RRT* with two different minimum tree nodes, are tested against one thousand randomly generated scenarios, three of which are shown in Figure 9 for example purposes. The comparison includes the cost function defined in Equation (24), the computation time, and the percentage of failed scenarios. Moreover, for each scenario, some further algorithm evaluation metrics proposed in the literature have been included [15]. In summary, we consider the following metrics:

- The cost of the solution, which we can generally recall as:

$$J(R) = \sum_{i=1}^{N-1} c((\mathbf{x}_{i-1}, \mathbf{x}_i), (\mathbf{x}_i, \mathbf{x}_{i+1})) \tag{29}$$

- The computation time;
- The path smoothness σ: $\prod_{i=1}^{N} S_i \to \mathbb{R}$:

$$\sigma(R) = \frac{1}{N-2}\left(\sum_{i=1}^{N-1} \theta^2(s_i, s_{i+1})\right)^{\frac{1}{2}} \tag{30}$$

- The minimum CPA during the maneuver CPA_{min}: $\prod_{i=1}^{N} S_i \to \mathbb{R}$:

$$CPA_{min}(R) = \min_{i\in\{1,N\}}\left(\min_{m\in\{1,\dots,M\}} CPA_m(s_i)\right) \tag{31}$$

- The path elongation L: $\prod_{i=1}^{N} S_i \to \mathbb{R}$:

$$L(R) = \sum_{i=1}^{N} |\vec{s}_i| \tag{32}$$

- The percentage of failed scenarios.

Each of the considered metrics is analyzed both in absolute value and in normalized form. If we denote with $\mu(k, a)$ a generic metric measuring the performance of algorithm a over the testing scenario k, the normalized metric $\mu_n(k)$ over a set A of algorithms is defined as follows:

$$\mu_n(k) = \frac{\mu(a,k) - \min_{a \in A} \mu(k,a)}{\max_{a \in A} \mu(k,a) - \min_{a \in A} \mu(k,a)} \tag{33}$$

While absolute metrics allow us to appreciate the absolute values of certain quantities of interest, which, in some cases, are significant (e.g., computation time), normalized metrics allow us to understand how often one algorithm is better than another on a per-scenario basis. We present the results in graphical format using violin plots in Figures 10–14.

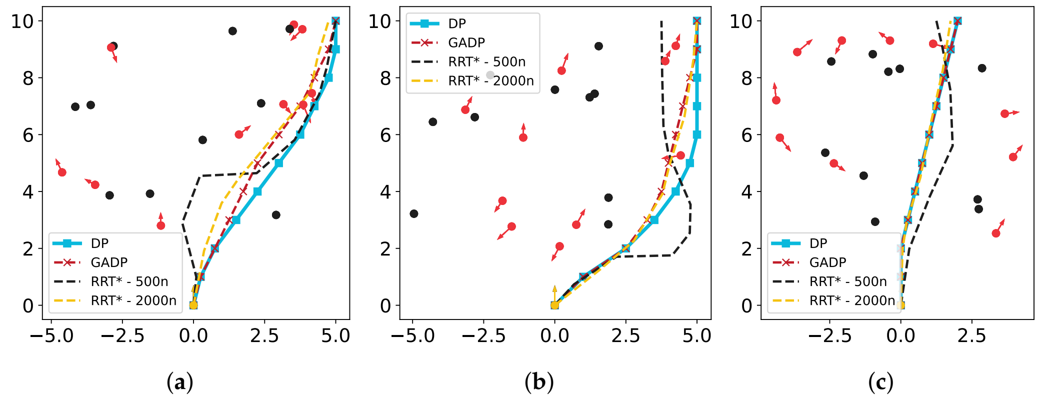
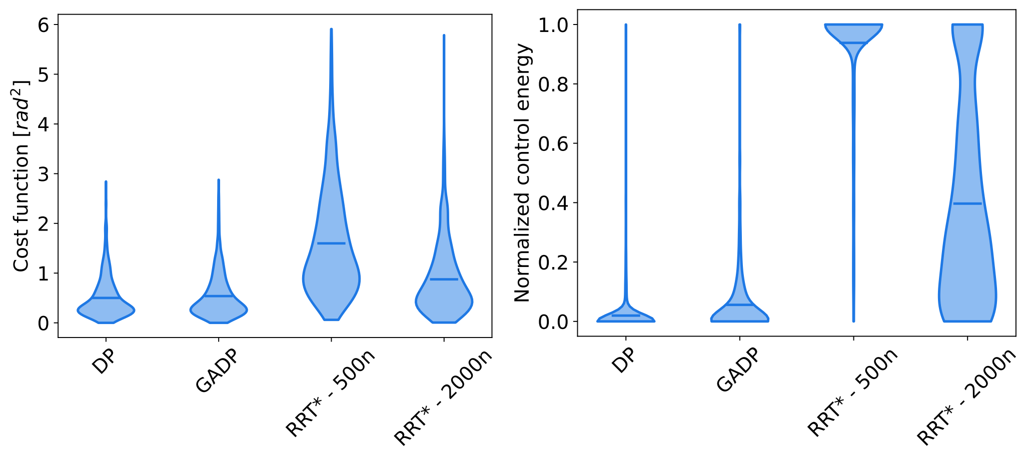


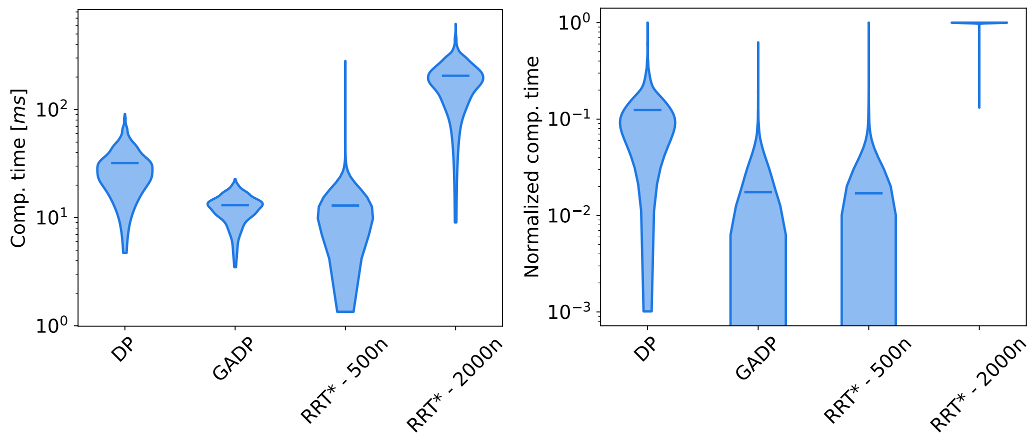
Figure 9. Subfigures (**a**–**c**) show three examples of the thousand randomly generated scenarios solved by the four algorithms. The black dots indicate randomly positioned obstacles, while the red arrows indicate randomly moving obstacles with randomly assigned positions, headings, and speeds, represented by the arrow direction and magnitude.

Figure 10 shows the distribution of control energy, i.e., the cost function. DP and GADP have lower values than RRT*. We also note that the optimality of the RRT* result increases if we impose a higher minimum number of nodes, as expected. The normalized values highlight that DP and GADP score the best results in most cases. In particular, the average value obtained by DP is the lowest of the four, followed by that of the GADP. Figure 11 shows the other side of the trade-off, which is computation time. In this case, we can see that GADP and RRT* with 500 nodes score the best performance, the former showing less variance than the latter. RRT* with 2000 nodes averages an order of magnitude above the others in absolute terms and is almost always the slowest on a per-scenario basis. Cross-comparison with Figure 10 allows us to appreciate the good trade-off of the proposed algorithms in the analyzed scenarios. Figure 12 shows the smoothness of the trajectories. Since DP and GADP provide a fixed number of waypoints, evaluating control energy and smoothness is roughly equivalent. This is not the case for RRT*, which provides solutions with varying numbers of waypoints. In comparison with Figure 10, we can observe that this difference favors the DP-based algorithms. Figure 13 presents the minimum CPAs obtained by the four algorithms. We can observe that they all meet the required safety distance constraint of 1 nmi with comparable performance. Such behavior is reasonable since the distance to obstacles does not intervene in ranking the solutions but only as a constraint. Figure 14 shows the results in terms of path length. Considering the absolute values, we might note that the four algorithms provide similar results; however, the normalized values highlight that the DP-based algorithms generally provide shorter paths than the RRT*-based algorithms on a per-scenario basis. Eventually, Figure 15 shows the percentage of failed scenarios, i.e., scenarios in which the algorithm fails to find the solution. We can again see how the greedy approximation has a greater tendency to fail scenarios even where a solution exists since DP can compute it. We note that RRT* algorithms have a lower failure rate due to the greater flexibility that a non-grid-based algorithm allows.



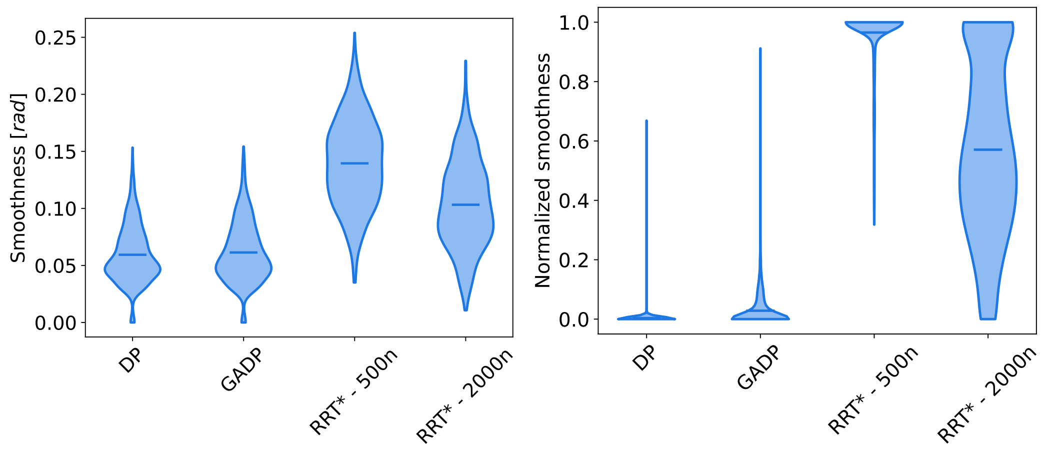
(**a**) Control energy (**b**) Normalized control energy

Figure 10. Per-algorithm distribution of the cost function (control energy) values found solving 1000 randomly generated scenarios.



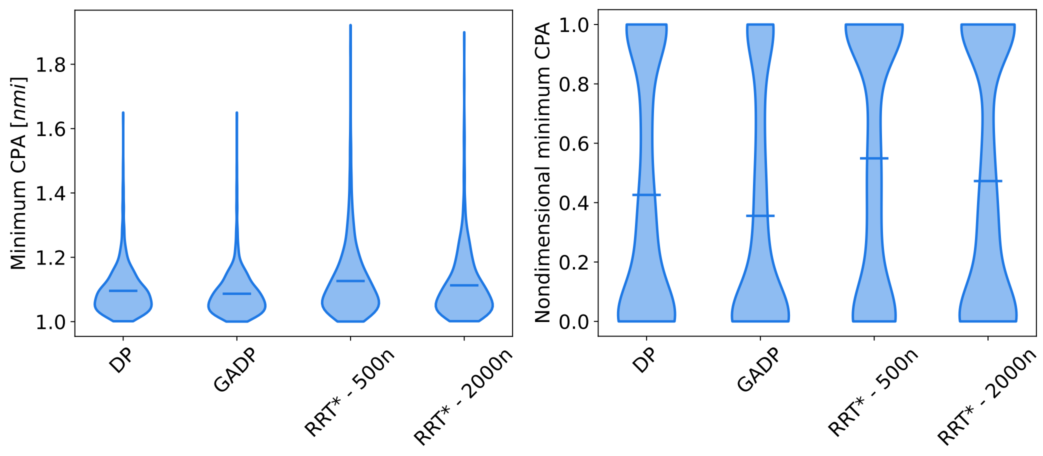
(**a**) Computation time (**b**) Normalized computation time

Figure 11. Per-algorithm distribution of the computation time required to solve 1000 randomly generated scenarios.



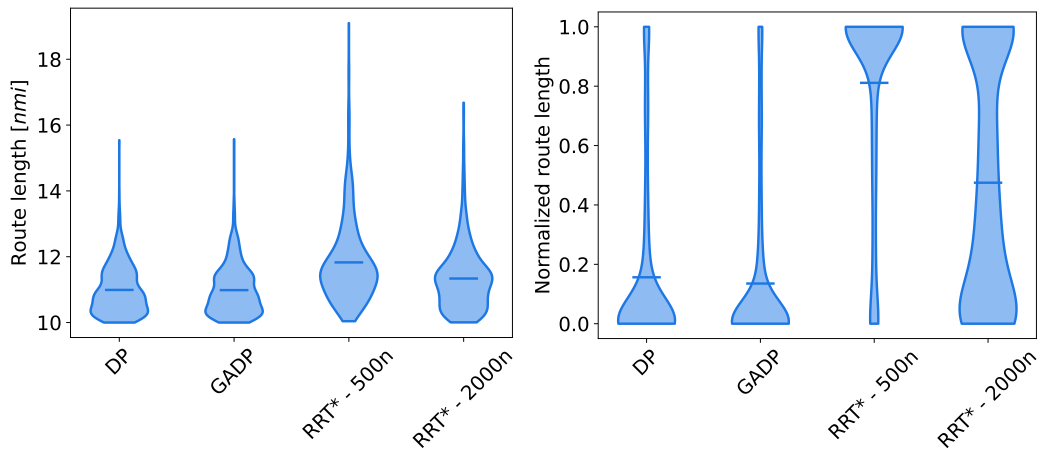
(**a**) Smoothness (**b**) Normalized smoothness

Figure 12. Per-algorithm smoothness distribution of the solutions of the 1000 randomly generated scenarios.



(**a**) Minimum CPA (**b**) Normalized minimum CPA

Figure 13. Per-algorithm distribution of the minimum CPA over 1000 randomly generated scenarios.



(**a**) Path length (**b**) Normalized path length

Figure 14. Per-algorithm path length distribution of the solutions found solving 1000 randomly generated scenarios.

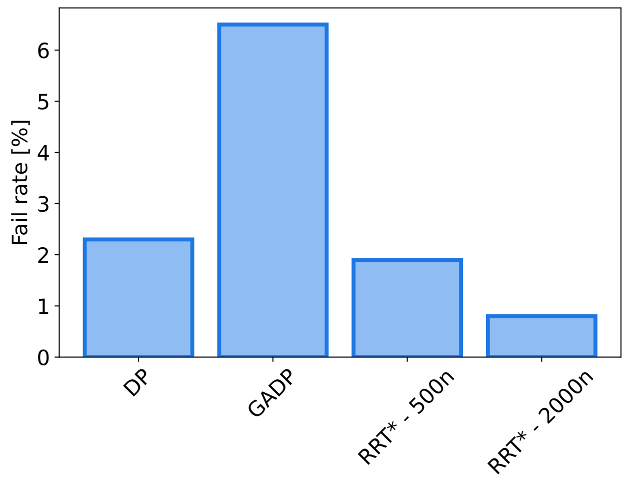


Figure 15. Per-algorithm scenario failure rate over 1000 randomly generated scenarios.

9. Conclusions

In this paper, we proposed a dynamic programming approach for the collision avoidance of autonomous ships. We showed how the collision-free route calculation for a ship can be formulated as a multi-stage optimal decision problem by leveraging Bellman's equation, and we described appropriate constraints for obtaining a collision-safe route with features compatible with the maneuvering capabilities of a ship and compliant with collision regulations.

We proposed and discussed a tabulation-based solution scheme to compute the solution. Moreover, we proposed a greedy approximate dynamic programming (GADP) scheme that allows, using a greedy approach, the number of transitions to be evaluated for each step to be reduced, consequently reducing the algorithm's time complexity.

We evaluated the performance of the proposed algorithms. Firstly, we tested the algorithms in some navigation scenarios. Secondly, we tested the proposed algorithms against randomly generated scenarios to evaluate the influence of grid parameters on solution optimality, computation time, and failure rate, highlighting the trade-off between DP and GADP. Finally, we compared the proposed algorithms with a known planning algorithm, the RRT*, using performance metrics proposed in the literature. Dynamic programming confirmed its role as a powerful and flexible tool for solving practical problems efficiently with low computational burden.

This work has laid a solid theoretical foundation, paving the way for integrated applications. The result showed that the proposed algorithms could be used in real-time applications, such as in an automatic navigation or decision support system, to play a first-class role in the transition towards autonomous shipping. As a future development of this research, we foresee the possibility of testing the developed algorithms in an autonomous navigation architecture both in simulation and model-scale experiments. Moreover, we envision human-in-the-loop decision support tests using the simulator from the perspective of full-scale testing.

Funding: This research was partially funded by the European Union's Horizon Europe under the call HORIZON-CL5-2022-D6-01 (Safe, Resilient Transport and Smart Mobility services for passengers and goods), grant number 101077026, project name SafeNav. However, the views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DP	Dynamic programming
COLREG	Convention on the International Regulations for Preventing Collisions at Sea
CPA	Closest Point of Approach
GADP	Greedy approximate dynamic programming
IMO	International Maritime Organization
MASS	Maritime Autonomous Surface Ship
RRT	Rapidly exploring random tree
RRT*	Optimal rapidly exploring random tree

References

1. Martelli, M.; Virdis, A.; Gotta, A.; Cassarà, P.; Di Summa, M. An outlook on the future marine traffic management system for autonomous ships. *IEEE Access* **2021**, *9*, 157316–157328. [CrossRef]
2. Tran, H.A.; Johansen, T.A.; Negenborn, R.R. Collision avoidance of autonomous ships in inland waterways—A survey and open research problems. In *Journal of Physics: Conference Series, Proceedings of the International Conference on Maritime Autonomous Surface Ships (ICMASS 2023), Rotterdam, The Netherlands, 8–9 November 2023*; IOP Publishing: Bristol, UK, 2023; Volume 2618, p. 012004.

3. IMO; MSC. *Outcome of the Regulatory Scoping Exercise for the Use of Maritime Autonomous Surface Ships (MASS)*; IMO: London, UK, 2021.
4. Alessandri, A.; Donnarumma, S.; Luria, G.; Martelli, M.; Vignolo, S.; Chiti, R.; Sebastiani, L. Dynamic positioning system of a vessel with conventional propulsion configuration: Modeling and simulation. In Proceedings of the 2nd International Conference in Maritime Technology and Engineering (MARTECH 2014), Lisbon, Portugal, 15–17 October 2014; pp. 725–733.
5. Alessandri, A.; Donnarumma, S.; Vignolo, S.; Figari, M.; Martelli, M.; Chiti, R.; Sebastiani, L. System control design of autopilot and speed pilot for a patrol vessel by using LMIs. In Proceedings of the 16th International Congress of the International Maritime Association of the Mediterranean, IMAM 2015, Pola, Croatia, 21–24 September 2015; pp. 577–583.
6. Singh, Y.; Bibuli, M.; Zereik, E.; Sharma, S.; Khan, A.; Sutton, R. A novel double layered hybrid multi-robot framework for guidance and navigation of unmanned surface vehicles in a practical maritime environment. *J. Mar. Sci. Eng.* **2020**, *8*, 624. [CrossRef]
7. Bruzzone, G.; Bibuli, M.; Caccia, M.; Zereik, E. Cooperative robotic maneuvers for emergency ship towing operations. In Proceedings of the 2013 MTS/IEEE OCEANS-Bergen, Bergen, Norway, 13–14 June 2013; pp. 1–7.
8. Chen, L.; Huang, Y.; Zheng, H.; Hopman, H.; Negenborn, R. Cooperative multi-vessel systems in urban waterway networks. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 3294–3307. [CrossRef]
9. Chen, L.; Haseltalab, A.; Garofano, V.; Negenborn, R.R. Eco-VTF: Fuel-efficient vessel train formations for all-electric autonomous ships. In Proceedings of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 2543–2550.
10. Faggioni, N.; Ponzini, F.; Martelli, M. Multi-obstacle detection and tracking algorithms for the marine environment based on unsupervised learning. *Ocean Eng.* **2022**, *266*, 113034. [CrossRef]
11. International Maritime Organization (IMO). *Convention on the International Regulations for Preventing Collisions at Sea*; IMO: London, UK, 1972.
12. Burmeister, H.C.; Constapel, M. Autonomous collision avoidance at sea: A survey. *Front. Robot. AI* **2021**, *8*, 739013. [CrossRef]
13. Zaccone, R.; Martelli, M. Interaction between COLREG-compliant collision avoidance systems in a multiple MASS scenario. In *Journal of Physics: Conference Series, Proceedings of the International Conference on Maritime Autonomous Surface Ships (ICMASS 2023), Rotterdam, The Netherlands, 8–9 November 2023*; IOP Publishing: Bristol, UK, 2023; Volume 2618, p. 012006.
14. Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.A.; Burgard, W. *Principles of Robot Motion: Theory, Algorithms, and Implementations*; MIT Press: Cambridge, MA, USA, 2005.
15. Filotheou, A.; Tsardoulias, E.; Dimitriou, A.; Symeonidis, A.; Petrou, L. Quantitative and qualitative evaluation of ROS-enabled local and global planners in 2D static environments. *J. Intell. Robot. Syst.* **2020**, *98*, 567–601. [CrossRef]
16. Seo, C.; Noh, Y.; Abebe, M.; Kang, Y.J.; Park, S.; Kwon, C. Ship collision avoidance route planning using CRI-based A* algorithm. *Int. J. Nav. Archit. Ocean Eng.* **2023**, *15*, 100551. [CrossRef]
17. Singh, Y.; Sharma, S.; Sutton, R.; Hatton, D. Optimal path planning of an unmanned surface vehicle in a real-time marine environment using a dijkstra algorithm. In *Marine Navigation*; CRC Press: Boca Raton, FL, USA, 2017; pp. 399–402.
18. Singh, Y.; Sharma, S.; Sutton, R.; Hatton, D.; Khan, A. Feasibility study of a constrained Dijkstra approach for optimal path planning of an unmanned surface vehicle in a dynamic maritime environment. In Proceedings of the 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Torres Vedras, Portugal, 25–27 April 2018; pp. 117–122.
19. D'Amato, E.; Nardi, V.A.; Notaro, I.; Scordamaglia, V. A Visibility Graph approach for path planning and real-time collision avoidance on maritime unmanned systems. In Proceedings of the 2021 International Workshop on Metrology for the Sea, Learning to Measure Sea Health Parameters (MetroSea), Reggio Calabria, Italy, 4–6 October 2021; pp. 400–405.
20. Chiang, H.T.L.; Tapia, L. COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2024–2031. [CrossRef]
21. Zaccone, R.; Martelli, M. A collision avoidance algorithm for ship guidance applications. *J. Mar. Eng. Technol.* **2020**, *19*, 62–75. [CrossRef]
22. Enevoldsen, T.T.; Reinartz, C.; Galeazzi, R. COLREGs-Informed RRT* for collision avoidance of marine crafts. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 8083–8089.
23. Zhu, Z.; Lyu, H.; Zhang, J.; Yin, Y. An efficient ship automatic collision avoidance method based on modified artificial potential field. *J. Mar. Sci. Eng.* **2022**, *10*, 3. [CrossRef]
24. Li, L.; Wu, D.; Huang, Y.; Yuan, Z.M. A path planning strategy unified with a COLREGS collision avoidance function based on deep reinforcement learning and artificial potential field. *Appl. Ocean Res.* **2021**, *113*, 102759. [CrossRef]
25. Ito, M.; Zhnng, F.; Yoshida, N. Collision avoidance control of ship with genetic algorithm. In Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No. 99CH36328), Kohala Coast, HI, USA, 22–27 August 1999; Volume 2, pp. 1791–1796.
26. Kang, Y.T.; Chen, W.J.; Zhu, D.Q.; Wang, J.H.; Xie, Q.M. Collision avoidance path planning for ships by particle swarm optimization. *J. Mar. Sci. Technol.* **2018**, *26*, 3.
27. Ning, J.; Chen, H.; Li, T.; Li, W.; Li, C. COLREGs-Compliant unmanned surface vehicles collision avoidance based on multi-objective genetic algorithm. *IEEE Access* **2020**, *8*, 190367–190377. [CrossRef]
28. Gao, P.; Zhou, L.; Zhao, X.; Shao, B. Research on ship collision avoidance path planning based on modified potential field ant colony algorithm. *Ocean Coast. Manag.* **2023**, *235*, 106482. [CrossRef]

29. Alessandri, A.; Donnarumma, S.; Martelli, M.; Vignolo, S. Motion control for autonomous navigation in blue and narrow waters using switched controllers. *J. Mar. Sci. Eng.* **2019**, *7*, 196. [CrossRef]
30. Piaggio, B.; Garofano, V.; Donnarumma, S.; Alessandri, A.; Negenborn, R.; Martelli, M. Follow-the-Leader Guidance, Navigation, and Control of Surface Vessels: Design and Experiments. *IEEE J. Ocean. Eng.* **2023**, *48*, 997–1008. [CrossRef]
31. Bellman, R. The theory of dynamic programming. *Bull. Am. Math. Soc.* **1954**, *60*, 503–515. [CrossRef]
32. Bellman, R. Dynamic programming. *Science* **1966**, *153*, 34–37. [CrossRef] [PubMed]
33. Jones, M.; Peet, M.M. A generalization of Bellman's equation with application to path planning, obstacle avoidance and invariant set estimation. *Automatica* **2021**, *127*, 109510. [CrossRef]
34. Ayyappan, B.; Gopalan, S. A Performance and Power Characterization study of Memoization and Tabulation methods in Graph Neural Networks by assessing Dynamic Programming Workloads. In Proceedings of the 2022 6th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, Indonesia, 13–14 December 2022; pp. 1–6.
35. Rytter, W. On efficient parallel computations for some dynamic programming problems. *Theor. Comput. Sci.* **1988**, *59*, 297–307. [CrossRef]
36. González, D.; Almeida, F.; Roda, J.; Rodriguez, C. From the theory to the tools: Parallel dynamic programming. *Concurr. Pract. Exp.* **2000**, *12*, 21–34. [CrossRef]
37. Guo, Q.; Li, Z.; Song, W.; Fu, W. Parallel computing based dynamic programming algorithm of track-before-detect. *Symmetry* **2018**, *11*, 29. [CrossRef]
38. Mazzarello, M.; Ottaviani, E. A traffic management system for real-time traffic optimisation in railways. *Transp. Res. Part B Methodol.* **2007**, *41*, 246–274. [CrossRef]
39. Shin, K.; McKay, N. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Trans. Autom. Control* **1986**, *31*, 491–500. [CrossRef]
40. Li, X.; Wang, L.; An, Y.; Huang, Q.L.; Cui, Y.H.; Hu, H.S. Dynamic path planning of mobile robots using adaptive dynamic programming. *Expert Syst. Appl.* **2024**, *235*, 121112. [CrossRef]
41. Zaccone, R.; Ottaviani, E.; Figari, M.; Altosole, M. Ship voyage optimization for safe and energy-efficient navigation: A dynamic programming approach. *Ocean Eng.* **2018**, *153*, 215–224. [CrossRef]
42. Zaccone, R.; Figari, M.; Martelli, M. An optimization tool for ship route planning in real weather scenarios. In Proceedings of the ISOPE International Ocean and Polar Engineering Conference, ISOPE, Sapporo, Japan, 10–15 June 2018; pp. 738–744.
43. Tang, Z.; Ji, C.; Wang, X. Design of ship route through waterway based on dynamic programming. In *Journal of Physics: Conference Series, Proceedings of the 4th International Conference on Advanced Materials, Intelligent Manufacturing and Automation (AMIMA 2021), Hangzhou, China, 2–4 April 2021*; IOP Publishing: Bristol, UK, 2021; Volume 1906, p. 012001.
44. Choi, G.H.; Lee, W.; Kim, T.W. Voyage optimization using dynamic programming with initial quadtree based route. *J. Comput. Des. Eng.* **2023**, *10*, qwad055. [CrossRef]
45. Zaccone, R.; Martelli, M.; Figari, M. A colreg-compliant ship collision avoidance algorithm. In Proceedings of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 2530–2535.
46. Martelli, M.; Žuškin, S.; Zaccone, R.; Rudan, I. A COLREGs-compliant decision support tool to prevent collisions at sea. *TransNav: Int. J. Mar. Navig. Saf. Sea Transp.* **2023**, *17*, 347–353. [CrossRef]
47. Vettor, R.; Soares, C.G. Development of a ship weather routing system. *Ocean Eng.* **2016**, *123*, 1–14. [CrossRef]
48. Gaggero, T.; Bucciarelli, F.; Besio, G.; Mazzino, A.; Villa, D. A method to assess safety and comfort for different ships types in a region of interest. *Ocean Eng.* **2022**, *250*, 110995. [CrossRef]
49. Hintze, J.L.; Nelson, R.D. Violin plots: A box plot-density trace synergism. *Am. Stat.* **1998**, *52*, 181–184. [CrossRef]
50. LaValle, S. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report; Department of Computer Science, Iowa State University: Ames, IA, USA, 1998.
51. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.