Distributed Swarm Trajectory Planning for Autonomous Surface Vehicles in Complex Sea Environments

Anqing Wang 1, Longwei Li 1, Haoliang Wang 2, Bing Han 1,3 and Zhouhua Peng 1,*

1 College of Marine Electrical Engineering, Dalian Maritime University, Dalian 116026, China; anqingwang@dlmu.edu.cn (A.W.); lilongwei@dlmu.edu.cn (L.L.); han.bing@coscoshipping.com (B.H.)
2 College of Marine Engineering, Dalian Maritime University, Dalian 116026, China; haoliang.wang12@dlmu.edu.cn
3 Shanghai Ship and Shipping Research Institute Co., Ltd., Shanghai 200135, China
* Correspondence: zhpeng@dlmu.edu.cn

Abstract: In this paper, a swarm trajectory-planning method is proposed for multiple autonomous surface vehicles (ASVs) in an unknown and obstacle-rich environment. Specifically, based on the point cloud information of the surrounding environment obtained from local sensors, a kinodynamic path-searching method is used to generate a series of waypoints in the discretized control space at first. Next, after fitting B-spline curves to the obtained waypoints, a nonlinear optimization problem is formulated to optimize the B-spline curves based on gradient-based local planning. Finally, a numerical optimization method is used to solve the optimization problems in real time to obtain collision-free, smooth and dynamically feasible trajectories relying on a shared network. The simulation results demonstrate the effectiveness and efficiency of the proposed swarm trajectory-planning method for a network of ASVs.

Keywords: autonomous surface vehicles; kinodynamic path searching; uniform B-spline curves; nonlinear optimization

1. Introduction

The autonomous surface vehicle (ASV), as an intelligent, miniaturized, and versatile unmanned marine transport platform that operates through remote control or autonomous navigation, has a variety of applications in offshore ocean engineering [1–13]. Among the various applications of ASVs, the trajectory planning of ASVs is particularly crucial to provide a safe trajectory. In order to obtain the optimal trajectory, trajectory planning of the ASV is described as a constrained optimization problem by using local sensor and global map information.

The trajectory-planning problem of an ASV has been widely studied in the literature [14–21]. Specifically, in [14], a temporal-logic-based ASV path-planning method is employed, which enables the ASV to pass through heavy harbor traffic to an intended destination in a collision-free manner. In [15], an evolutionary-based path-planning approach is proposed for an ASV to accomplish environmental monitoring tasks. In [16], an extension of the hybrid-A* algorithm is proposed to plan optimal ASV paths under kinodynamic constraints in a leader-following scenario. Another hybrid-A*-based two-stage method is provided in [17] for energy-optimized ASV trajectory planning with experimental validation. In [18], a novel receding horizon multi-objective planner is developed for an ASV performing path planning in complex urban waterways. In [19], an essential visibility graph approach is proposed to generate optimal paths for an ASV with real-time collision avoidance. Furthermore, in [20], the particle swarm optimization algorithm together with the visibility graph is applied to the ASV path-planning problem among complex shorelines and spatiotemporal environmental forces. Recently, in [21], a hybrid artificial potential field method is proposed for an ASV cruising in a dynamic riverine environment. However,
it is worth mentioning that the works [14–21] are dedicated to the path planning of a single ASV.

Compared to a single ASV, multiple ASVs are more likely to complete difficult tasks such as exploration and development, territorial sea monitoring and route planning [4,22–32]. Therefore, the trajectory planning of multiple ASVs, as one of the key topics in the field of marine science, has attracted increasing interest in scientific research, and much success has been achieved. Existing path-planning methods for multiple ASVs include model predictive control [33–35], the RRT method [36], satellite maps [37], the priority target assignment method [38], the Voronoi-Visibility roadmap [39], and the multidimensional rapidly exploring random trees star algorithm [40], to list just a few. However, due to the uncertainties of obstacles, and limitations in sensor range and communication bandwidth, the trajectory planning of multiple ASVs in an unknown—an especially an obstacle-rich—marine environment still presents a great challenge. In fact, to the best of our knowledge, as for the autonomous navigation of multiple ASVs, little research is available that constructs a multi-objective optimization problem with optimal energy consumption, dynamic feasibility, and obstacle avoidance distance in unknown environments based on local sensor information.

In light of the above discussions, this paper focuses on the distributed swarm trajectory-planning problem for multiple ASVs in a complex and obstacle-rich marine environment. Firstly, by utilizing local sensors, the surrounding environment’s point cloud data are acquired, and a kinodynamic path-search technique is employed to generate a series of waypoints within a discretized control space. Subsequently, B-spline curves are fitted to these waypoints, and a nonlinear optimization problem is formulated. This problem is optimized using a gradient-based local planning approach, which allows the generation of collision-free, smooth, and dynamically feasible trajectories through a shared network. Compared with the existing trajectory-planning methods, the proposed trajectory-planning method has the following features:

- In contrast to the existing trajectory-planning methods dedicated to a single ASV [14–21] and multiple USVs [36–40], this work constructs a multi-objective optimization problem with optimal energy consumption, dynamic feasibility, curve smoothness, obstacle avoidance distance, and endpoint distance constraints based on local sensor information. By utilizing local onboard sensor information, ASVs can accomplish autonomous navigation requiring no external localization and computation or a pre-built map in unknown complex environments.

- By adopting the proposed planning method, the optimal trajectory generation problem for multiple ASVs is divided into initial trajectory generation and back-end trajectory optimization. A heuristic-based kinodynamic path search is employed to efficiently find a safe, feasible, and minimum-time initial path. The initial path is then optimized by a B-spline optimization incorporating gradient-based local planning. By this means, the generated trajectories are able to meet the dynamic feasibility with enhanced safety.

- By adopting the proposed planning method, ASVs can perceive the surrounding environment in real time, using point cloud information obtained from local sensors to generate real-time local optimal trajectories. Moreover, this work utilizes multiple local optimal trajectories to form a global trajectory, which better meets the requirements of real-time obstacle avoidance and generates shorter sail distance for short-range autonomous navigation.

The rest of this paper is organized as follows. Some preliminaries and the problem formulation are given in Section 2. Section 3 discusses the process of swarm trajectory planning of ASVs. Section 4 provides some simulation results to demonstrate the effectiveness of the proposed method. Section 5 concludes this paper with some concluding remarks and future works.
2. Preliminaries and Problem Formulation

2.1. Preliminaries

\( \mathbb{R}^n \) denotes the \( n \)-dimensional Euclidean space. The Euclidean norm is denoted by \( \| \cdot \| \). \( \lambda_{\min}(\cdot) \) and \( \lambda_{\max}(\cdot) \) represent the minimum and maximum eigenvalues of a square matrix \( (\cdot) \), respectively. \( \mathbb{N}^+ \) denotes a positive integer. \( \mathbb{R}_+ \) and \( \mathbb{R}_- \) represent the positive and negative real numbers, respectively.

2.2. Problem Formulation

In the trajectory planning, each ASV is governed by the kinematic model expressed as follows:

\[
\begin{align*}
\dot{x} &= u \cos \psi - v \sin \psi \\
\dot{y} &= u \sin \psi + v \cos \psi \\
\dot{\psi} &= r
\end{align*}
\]

(1)

where \([x, y]\) and \(\psi\) denote the position and yaw angle in an earth-fixed frame, respectively. \(u, v,\) and \(r\) are the surge velocity, sway velocity, and yaw rate in a body-fixed frame, respectively. For the trajectory-planning task, it is assumed that \(v = 0\) such that

\[
\begin{align*}
\dot{x} &= u \cos \psi \\
\dot{y} &= u \sin \psi \\
\dot{\psi} &= r
\end{align*}
\]

(2)

In this paper, a real-time trajectory-planning method is proposed for multiple ASVs to reach the designated target points in a complex and obstacle-rich environment, as shown in Figure 1.

Figure 1. An illustration of trajectory planning for multiple ASVs to reach the designated locations. The dashed lines represent trajectories that have not yet been optimized, the solid lines indicate the optimized trajectories, and the black obstacles represent the complex static environments.

3. Swarm Trajectory Planning

This section gives a general description of the proposed swarm trajectory-planning method, including the construction of occupancy grid maps, the kinodynamic path-searching method, the curve-fitting method based on cubic uniform B-spline, the construction of nonlinear optimization problems under multiple constraint conditions, and the numerical optimization algorithm for solving the formulated optimization problem. The proposed trajectory-planning framework for multiple ASVs is shown in Figure 2.
3.1. Construction of Occupancy Grid Map

The sensing radius of the sensor is set to $r_{id}$ and takes a circular area for point cloud acquisition. The position of the ASV is set to $p_{id} = [x_{id}, y_{id}, z_{id}] \in \mathbb{R}^3$. The point cloud position coordinates obtained by sampling the surrounding environment are set to $p_{tid} = [x_{tid}, y_{tid}, z_{tid}] \in \mathbb{R}^3$, and the quaternion of the ASV is set to $q_{id} = [w_{qid}, x_{qid}, y_{qid}, z_{qid}] \in \mathbb{R}^4$. In order to obtain the point cloud data within the desired range, the angle value corresponding to the maximum height of point cloud collection should be calculated. A grid is occupied and takes a circular area for point cloud acquisition. To represent the presence of obstacles, an occupied grid map is constructed by taking a circular area for point cloud acquisition. To represent the presence of obstacles, an occupied grid map is constructed by setting a grid as occupied if $m_{id} = 1$ and as free if $m_{id} = 0$.

For a grid $m$ in an occupancy grid map, the probability of the grid state being occupied is denoted as $p(m)$ and the probability of the grid state being free is denoted as $p(\overline{m})$. The $t$-th observation is set to $z_t$. To calculate the posterior probability of the grid state based on existing observations, $p(m_i | z_{1:t-1})$ and $p(\overline{m_i} | z_{1:t-1})$ should be calculated. A grid is occupied if $p(m_i | z_{1:t-1}) \geq \kappa$ with $\kappa$ being a preset threshold. The probability of the grid state being occupied can be specifically denoted as

$$p(m_i | z_{1:t-1}) = \frac{p(z_t | z_{1:t-1}, m_i)p(m_i | z_{1:t-1})}{p(z_t | z_{1:t-1})}$$

(4)

and the $t$-th observation is obtained as follows:

$$p(m_i | z_t) = \frac{p(z_t | m_i)p(m_i)}{p(z_t)}$$

(5)

According to the Markov assumption that the results of the first $t - 1$ observations are independent of the result of the $t$-th observation, one has that

$$p(z_t | z_{1:t-1}, m_i) = p(z_t | m_i)$$

(6)
Expanding (4) based on the Bayes formula, it follows that

\[
p(m_i | z_{1:t}) = \frac{p(m_i | z_t) p(m_i | z_{1:t-1})}{p(m_i) p(z_t | z_{1:t-1})}
\]  

(7)

and

\[
p(\overline{m}_i | z_{1:t}) = \frac{p(\overline{m}_i | z_t) p(\overline{m}_i | z_{1:t-1})}{p(\overline{m}_i) p(z_t | z_{1:t-1})}
\]  

(8)

Define \( l_t(m_i) \) as the \( t \)-th update posterior probability for a grid \( m_i \); then, one has that

\[
\begin{align*}
l_t(m_i) &= \log \frac{p(m_i | z_{1:t})}{p(\overline{m}_i | z_{1:t})} \\
l_{t-1}(m_i) &= \log \frac{p(m_i | z_{1:t-1})}{p(\overline{m}_i | z_{1:t-1})}
\end{align*}
\]

(9)

Further simplifying the above equation using (7), \( l_t(m_i) \) can be described as

\[
l_t(m_i) = \log \frac{p(m_i | z_t)}{p(\overline{m}_i | z_t)} - \log \frac{p(m_i)}{p(\overline{m}_i)} + l_{t-1}(m_i)
\]  

(10)

Substituting (5) into (10) yields

\[
l_t(m_i) = \log \frac{p(z_t | m_i)}{p(z_t | \overline{m}_i)} + l_{t-1}(m_i)
\]  

(11)

By assuming that the sensor model will not change during the environmental modeling process, the sensor model formulas \( p(z_t | m_i) \) and \( p(z_t | \overline{m}_i) \) are constant. Then, according to (9), one has that

\[
p(m_i | z_{1:t}) = \frac{a^{l_t(m_i)}}{1 + a^{l_t(m_i)}}
\]  

(12)

where \( a \) is the base of the \( \log \) function. Based on the posterior probability, the occupied grid map can be updated. The occupied grid map constructed by ASVs during autonomous navigation is depicted in Figure 3.

Figure 3. An illustration of the occupied grid map in a complex and obstacle-rich environment. The colored point cloud part represents the environmental information perceived by the ASV, and the gray part represents the unknown obstacle environment.

3.2. Kinodynamic Path Searching

Inspired by the hybrid A* search proposed for autonomous vehicles in [41], the kinodynamic path searching is applied for ASV to obtain a safe and reliable trajectory in an occupancy grid map while minimizing time cost. The mechanism of the kinodynamic path searching is illustrated in Figure 4.
Figure 4. A schematic diagram of the kinodynamic path searching. The blue dashed line represents the motion primitives obtained by expanding the state point by Equation (14), and the black curve represents the optimal curve selected based on the heuristic function.

Similar to the standard A*, we use openlist and closelist to denote the open set and the closed set, respectively. A structure node is used to record the position of the expansion point as well as the \( g_c \) and \( f_c \) cost (Section 3.2.2). The nodes expand iteratively, and the one with the smallest \( f_c \) is saved in openlist. Then, CheckCollision(·) checks the safety and dynamic feasibility of the nodes. The searching process ends once any node reaches its goal successfully or the AnalyticExpand(·) succeeds. Details of the kinodynamic path searching are shown in Algorithm 1.

Algorithm 1 Kinodynamic Path Searching

\[
\begin{align*}
\textbf{while} & \quad \text{openlist.empty}() \quad \textbf{do} \\
& \quad \textbf{openlist pop} \; n_c \\
& \quad \textbf{closelist insert} \; n_c \\
& \quad \textbf{if} \; \text{Reachgoal}(n_c) \; \textbf{or} \; \text{AnalyticExpand}(n_c) \; \textbf{then} \\
& \quad \quad \text{return Path()} \\
& \quad \textbf{end if} \\
& \quad \text{expandnodes ← NodeExpand}(n_c) \\
& \quad \text{nodes ← NodePrune(expandnodes)} \\
\textbf{end while} \\
& \quad \textbf{for each} \; n_i \; \in \; \text{nodes do} \\
& \quad \quad \textbf{if} \; \text{closelist contain} \; n_i \; \textbf{and} \; \text{CheckCollision}(n_i) \; \textbf{then} \\
& \quad \quad \quad g_t \leftarrow n_c.g_c + \text{CurrentCost}(n_i) \\
& \quad \quad \quad \textbf{if} \; \text{openlist contain} \; n_i \; \textbf{then} \\
& \quad \quad \quad \quad \textbf{openlist add} \; n_i \\
& \quad \quad \quad \quad \textbf{if} \; g_t \geq n_i.g_c \; \textbf{then} \\
& \quad \quad \quad \quad \quad \text{continue} \\
& \quad \quad \quad \quad \textbf{end if} \\
& \quad \quad \quad \textbf{end if} \\
& \quad \quad \textbf{end if} \\
& \quad \textbf{end for} \\
& n_i.parent ← n_c, n_i.g_c ← g_t \\
& n_i.f_c ← n_i.g_c + h(n_i)
\end{align*}
\]
3.2.1. Primitives Generation

The motion primitives are generated by using NodeExpand(·) in Algorithm 1. Accordingly, we firstly define the discretization sampling of \( r_j, T_j \) and \( u_k, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n, k = 1, 2 \) with \( m, n \in \mathbb{N}^+ \) as follows:

\[
\begin{align*}
  r_i & \in \{ r_0, r_1, \ldots, r_m \} \\
  T_j & \in \{ T_0, T_1, \ldots, T_n \} \\
  u_k & \in \{ -u_{\text{max}}, u_{\text{max}} \}
\end{align*}
\]

where \( r_j, i = 1, 2, \cdots, m \) denotes a set of discretized yaw rates, \( T_j, j = 1, 2, \cdots, n, \) denotes a set of durations, and \( u_k, k = 1, 2 \) denotes the control input.

Let \( p_0 = [x_0, y_0, \psi_0] \in \mathbb{R}^3 \) be the node recording the current pose of the ASV, and let \( p_i = [x_i, y_i, \psi_i] \in \mathbb{R}^3 \) be the pose of the ASV after sampling. Recalling the ASV kinematic model (2) and applying the control variables \( u_k \) and \( r_i \) for duration \( T_j \), the pose of the ASV can be calculated by

\[
\begin{align*}
  x_i &= x_0 + \int_0^{T_j} u_k \cos(\psi_i) \, d\tau \\
  y_i &= y_0 + \int_0^{T_j} u_k \sin(\psi_i) \, d\tau \\
  \psi_i &= \psi_0 + \int_0^{T_j} r_i \, d\tau
\end{align*}
\]

3.2.2. Heuristic Cost

In this subsection, a heuristic function \( h(\cdot) \) is designed to speed up the searching in Algorithm 1 as used in A* algorithm. Heuristic cost refers to constructing a boundary optimal problem using the Pontryagin extremum principle, considering multiple target points generated by the A* path search method. Computing the cost values of all candidate target points, the one with the minimal cost is selected as the optimal target point. Then, based on the selected optimal target point, the A* algorithm is used to iteratively search toward the endpoint.

In order to find a trajectory that is optimal in time, the Pontryagins minimum principle is applied to design a heuristic cost function \( J(T) \) as follows:

\[
\begin{align*}
  J(T) &= T + \sum_{\mu \in \{ s, f \}} \left( \frac{1}{2} \alpha_{\mu}^2 T^3 + \alpha_{\mu} \beta_{\mu} T^2 + \beta_{\mu}^2 T \right) \\
  \begin{bmatrix} \alpha_{\mu} \\ \beta_{\mu} \end{bmatrix} &= \begin{bmatrix} -\frac{12}{T^3} & -\frac{6}{T^2} \\ \frac{6}{T} & \frac{3}{T} \end{bmatrix} \begin{bmatrix} p_{\mu f} - p_{\mu 0} - v_{\mu 0} T \\ v_{\mu f} - v_{\mu 0} \end{bmatrix}
\end{align*}
\]

where \( p_{\mu 0}, p_{\mu f} \) are the current and goal position, respectively, and \( v_{\mu 0}, v_{\mu f} \) are the current and goal velocity, respectively.

In order to obtain the minimum heuristic cost, \( \alpha_{\mu} \) and \( \beta_{\mu} \) are substituted into \( J(T) \) to obtain the solutions of \( \partial J(T) / \partial T = 0 \). The solution which minimizes \( J(T) \) is denoted as \( T_{\text{min}} \), and \( J(T_{\text{min}}) \) represents the heuristic cost \( h_c \) for the current node. Moreover, \( g_c \) is used to represent the actual cost of the trajectory from the start position \( (x_s, y_s) \) to the current state \( (x_c, y_c) \), and it is calculated as \( g_c = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2} \). Thus, the final cost of the current state is obtained by \( f_c = g_c + h_c \).

3.2.3. Collision Check

Collisions are checked based on the occupied grid map constructed by point clouds from local sensors. We assume that the entire ASV is included in a maximum rectangular box; then, a series of rectangular boxes is constructed sequentially on the trajectory points along the generated trajectory. All rectangular boxes are detected at a frequency of 20 Hz to determine whether they overlap with the surrounding point clouds or not. If there is
overlap, it is determined that a collision has occurred, and multiple ASVs’ trajectories need to be replanned. Otherwise, it is determined that the trajectory is safe.

To be specific, approximating an ASV by a rectangle, a set of footprints $S$ representing the area occupied by the ASV is obtained by $x_i, y_i$ and $\psi_i$ along the footprints, as illustrated in Figure 5. The union of footprints $S$ is called the swath along trajectory, which needs to be checked for collisions, as can be found in CheckCollision($\cdot$) in Algorithm 1. With the obtained swath, the occupancy grids of the swath are calculated to see if they overlap with obstacles on the occupancy grid map, as shown in Figure 6.

![Figure 5. An illustration of the swath along trajectory. The red rectangle represents the area occupied by the ASV. The black area represents complex static obstacles.](image1)

![Figure 6. Schematic diagram of collision checking during autonomous navigation of an ASV. The red cube area represents the area occupied by the ASV, while the gray area represents complex static obstacles.](image2)
3.2.4. Analytic Expansion

Generally speaking, it is difficult for discretized input to reach the endpoint accurately. To this end, an analytic expansion scheme AnalyticExpand() in Algorithm 1 is induced to speed up the trajectory searching. When the current node $p_{i0}$ is close to the endpoint $p_{f}$, the same approach used in Section 3.2.2 is directly applied to compute a trajectory from $p_{i0}$ to $p_{f}$ without generating primitives. If the trajectory can pass the safety and dynamic feasibility check, the path searching is terminated in advance. This strategy proves to be beneficial in enhancing efficiency, particularly in a sparse environment, as it greatly improves the success rate of the algorithm and terminates the searching earlier.

3.3. Trajectory Smoothing

Theoretically, the safety and dynamic feasibility of the path generated by kinodynamic path searching cannot be strictly guaranteed, as kinodynamic path searching ignores the distance information of obstacles and does not consider curve smoothness. Therefore, a B-spline curve is used in this section to fit the path curve.

3.3.1. Cubic Uniform B-spline

Let $q = \{q_0, q_1, \ldots, q_n\}$ be the control points obtained from kinodynamic path searching and $\Theta = \{\theta_0, \theta_1, \ldots, \theta_m\}$ be the knot vector with $q_k \in \mathbb{R}^2$, $\theta_i \in \mathbb{R}$ and $m = n + p + 1$. A B-spline is a piecewise polynomial determined by its degree $p$ and $n+1$ control points $q$ and $\Theta$. A cubic B-spline trajectory, used to fit the above control points, is parameterized by $\theta$. For a uniform cubic B-spline trajectory, it is noted that each knot span satisfies $\Delta \theta_k = \theta_{k+1} - \theta_k = \Delta \theta$.

The convex hull property of B-spline curves is illustrated in Figure 7. For $\theta \in [\theta_i, \theta_{i+1})$, $i = 0, 1, \ldots, m-1$, the four control points of a cubic uniform B-spline trajectory set within the knot vector $\theta$ are $q_{k-3}, q_{k-2}, q_{k-1}, q_k$, where $3 \leq k \leq n$. To construct B-spline curves, the B-spline basis function firstly needs to be calculated, of which the degree is set to $p$. Denoting the $k$-th B-spline basis function of degree $p$ as $N_{k,p}$, the 0-order and the $p$-order basis function are given as follows:

$$
\begin{align*}
N_{k,0} &= \begin{cases} 0 & \text{if } \theta_i \leq \theta < \theta_{i+1} \\ 1 & \text{otherwise} \end{cases} \\
N_{k,p} &= \frac{\theta - \theta_i}{\theta_{i+p} - \theta_i} N_{k,p-1}(\theta) + \frac{\theta_{i+p+1} - \theta}{\theta_{i+p+1} - \theta_{i+1}} N_{k+1,p-1}(\theta)
\end{align*}
$$

(16)

For ease of implementation, $p$ is set to $p = 3$. Correspondingly, the four basis functions are set to $N_{k-3,3}, N_{k-2,3}, N_{k-1,3}$, and $N_{k,3}$.

Defining a normalized variable $s(\theta) = (\theta - \theta_i)/\Delta \theta$, and substituting $s(\theta)$ into (16), one has that

$$
\begin{align*}
N_{k-3,3} &= \frac{1}{6} (1 - s(\theta))^3 \\
N_{k-2,3} &= \frac{1}{2} s(\theta)^3 - s(\theta)^2 + \frac{2}{3} \\
N_{k-1,3} &= -\frac{1}{2} s(\theta)^3 + \frac{1}{2} s(\theta)^2 + \frac{1}{2} s(\theta) + \frac{1}{6} \\
N_{k,3} &= \frac{1}{6} s(\theta)^3
\end{align*}
$$

(17)

Therefore, the following matrix can be calculated to represent the coefficients of cubic uniform B-spline trajectories:

$$
\begin{align*}
p(s(\theta)) &= n_k Q_k \\
n_k &= \begin{bmatrix} N_{k-3,3} & N_{k-2,3} & N_{k-1,3} & N_{k,3} \end{bmatrix} \\
Q_k &= \begin{bmatrix} q_{k-3} & q_{k-2} & q_{k-1} & q_k \end{bmatrix}^T
\end{align*}
$$

(18)
3.3.2. Convex Hull Property

In order to ensure the dynamic feasibility, it is necessary to construct a nonlinear constrained optimization problem for the first-order and second-order derivatives of B-spline trajectory. For this purpose, it is necessary to prove that the derivative of a B-spline is also a B-spline. With control points \( \{ q_0, q_1, \ldots, q_n \} \) and basis functions \( N_{k,p} \) defined above, a \( p \)-degree B-Spline \( C(\theta) \) can be obtained as follows:

\[
C(\theta) = \sum_{k=0}^{n} N_{k,p}(\theta)q_k
\]  

(19)

with its first-order and second-order derivatives being

\[
\begin{align*}
\dot{C}(\theta) &= \sum_{k=0}^{n-1} N_{k+1,p-1}(\theta)\dot{q}_k \\
\ddot{C}(\theta) &= \sum_{k=0}^{n-2} N_{k+2,p-2}(\theta)a_k
\end{align*}
\]  

(20)

Moreover, the first derivative of the basis function is given as follows:

\[
\frac{dN_{k,p}(\theta)}{d\theta} = \frac{pN_{k,p-1}(\theta)}{\theta_{i+p} - \theta_i} + \frac{-pN_{k+1,p-1}(\theta)}{\theta_{i+p+1} - \theta_{i+1}}
\]  

(21)
Substituting (21) into (20) yields
\[
\mathcal{C}(\theta) = \sum_{k=0}^{n-1} N_{k+1,p-1}(\theta)(\frac{pq_{k+1}}{\theta_{i+p} - \theta_{i+1}} + \frac{-pq_k}{\theta_{i+p+1} - \theta_{i+1}}) \tag{22}
\]
Thus, the control points of the B-spline \(C(\theta)\)'s first derivative \(v_k\) can be computed by
\[
v_k = \frac{pq_{k+1}}{\theta_{i+p} - \theta_{i+1}} + \frac{-pq_k}{\theta_{i+p+1} - \theta_{i+1}} \tag{23}
\]
Since the B-spline trajectory used in Section 3.3.1 is uniform, Equation (23) is further simplified as follows:
\[
v_k = \frac{q_{k+1} - q_k}{\Delta \theta} \tag{24}
\]
Similarly, the second-order and third-order derivatives of the B-spline trajectory can be further derived as follows:
\[
\begin{cases}
  a_k = \frac{v_{k+1} - v_k}{\Delta \theta} = \frac{1}{\Delta \theta^2} (q_{k+2} - 2q_{k+1} + q_k) \\
  h_k = \frac{a_{k+1} - a_k}{\Delta \theta} = \frac{1}{\Delta \theta^3} (q_{k+3} - 3q_{k+2} + 3q_{k+1} - q_k)
\end{cases} \tag{25}
\]
3.4. Nonlinear Optimization

For the B-spline trajectory defined by a set of control points \(\{q_0, q_1, \ldots, q_n\}\) in Section 3.3, a nonlinear optimization method is constructed in this subsection to further ensure the safety of the curve. The overall optimization function is defined as follows:
\[
f_{\text{total}} = \lambda_1 f_{\text{sm}} + \lambda_2 f_{c} + \lambda_3 (f_{s} + f_{s}) + \lambda_4 f_{\text{sw}} + \lambda_5 f_{t}, \tag{26}
\]
where \(f_{\text{sm}}\) and \(f_{c}\) are optimization terms for trajectory smoothness and collision distance, respectively. \(f_{s}\) and \(f_{s}\) are constrained optimization terms of velocity and acceleration, respectively. \(f_{\text{sw}}\) and \(f_{t}\) are optimization terms for the collision distance between ASVs and endpoint arrival, respectively. \(\lambda_1, \lambda_2, \lambda_3, \lambda_4\) and \(\lambda_5\) are the designed weight coefficients.

3.4.1. Smoothness Penalty

The smoothness cost \(f_{\text{sm}}\) is defined by a function that uses the integral of the squared jerk. According to (25), the jerk (i.e., the 3rd-order derivative of the position) of the trajectory is minimized to obtain a smooth trajectory. \(f_{\text{sm}}\) is defined as follows:
\[
f_{\text{sm}} = \sum_{k=2}^{n-3} ||h_k||^2 = \sum_{k=2}^{n-3} ||q_{k+2} - 3q_{k+1} + 3q_k - q_{k-1}||^2 \tag{27}
\]
3.4.2. Collision Distance Penalty

Initially, a naive B-spline trajectory is given regardless of whether the control points collide with an obstacle or not, as depicted by the black solid line passing through the obstacle in Figure 8. Therefore, the naive trajectory needs to be optimized by the A* algorithm to obtain a collision-free trajectory \(\Gamma\).

Specifically, for control points \(q_k\) on the colliding segment, points on the obstacle boundary denoted by \(p_k\) are assigned with corresponding repulsive direction vectors \(v_k\). The distance between \(q_k\) and \(p_k\) is denoted by \(d_k = (q_k - p_k)^T v_k\), as shown in Figure 8. In order to avoid generating \(p_k, v_k\) repeatedly, the B-spline trajectory can only be optimized when \(d_k > 0\). By ensuring that \(d_k\) is less than the safe distance \(s_f\), the control points can be kept away from the obstacles. In order to optimize the collision distance of the B-spline trajectory, a twice continuously differentiable function \(F_{c}\) is applied as follows:
The collision penalty function is denoted as $f_c$ and its derivative can be obtained as follows:

\[
\begin{align*}
\mathbf{c}_k &= s_f - d_k \\
F_c &= \begin{cases} 
0 & c_k \leq 0 \\
c_k^3 & 0 < c_k \leq s_f \\
3s_f c_k^2 - 3s_f^2 c_k + s_f^3 & s_f < c_k
\end{cases} \\
fc &= \sum_{k=2}^{N-3} F_c(q_k) \\
f_c &= \frac{\delta fc}{\delta q_k} = \begin{cases} 
-3c_k v_k & c_k \leq s_f \\
-6s_f c_k + 3s_f^2 & s_f < c_k
\end{cases}
\end{align*}
\]

Figure 8. The black solid line passing though the obstacle represents the naive B-spline trajectory to be optimized. The red solid line represents the edge of the obstacle obtained by A* search.

3.4.3. Dynamic Feasibility Penalty

The dynamic feasibility can be ensured by constraining the high-order derivatives of B-spline trajectories at discrete control points $\{q_0, q_1, \ldots, q_n\}$. Specifically, due to the convex hull property, constraining derivatives of the control points is sufficient for constraining the whole B-spline. Therefore, the constrained optimization terms of velocity and acceleration are given, respectively, as follows:
where \(a_1, b_1, c_1, a_2, b_2, \) and \(c_2\) are design parameters to ensure the second-order continuous differentiability of \(F_v\) and \(F_a\). \(v_{\text{max}}\) and \(a_{\text{max}}\) are the derivative limits, respectively. \(v_j\) and \(a_j\) are the splitting points of quadratic and cubic curves, respectively. According to (24) and (25), the first-order derivatives corresponding to \(f_v\) and \(f_a\) are given, respectively, as follows:

\[
\frac{\partial f_v}{\partial q_k} = \frac{1}{\Delta \theta} \left\{ \sum_{k=3}^{N-3} 3 \frac{\partial v_k}{\partial q_k} (v_k - v_{\text{max}}) \left( v_{\text{max}} \leq v_k \leq v_j \right) \right. \\
\left. \sum_{k=3}^{N-3} 2 \frac{\partial v_k}{\partial q_k} a_2 v_k + \frac{\partial v_k}{\partial q_m} b_2 \left( v_k \geq v_j \right) \right\}
\]

\[
\frac{\partial f_a}{\partial q_k} = \frac{1}{\Delta \theta^2} \left\{ \sum_{k=3}^{N-3} 3 \frac{\partial a_k}{\partial q_k} (a_k - a_{\text{max}}) \left( a_{\text{max}} \leq a_k \leq a_j \right) \right. \\
\left. \sum_{k=3}^{N-3} 2 \frac{\partial a_k}{\partial q_k} a_2 a_k + \frac{\partial a_k}{\partial q_m} b_2 \left( a_k \geq a_j \right) \right\}
\]

### 3.4.4. Swarm Distance Penalty

An illustration of swarm distance penalty is depicted in Figure 9. Similar to the collision distance penalty and dynamic feasibility penalty, the swarm distance penalty function is formulated as follows:

\[
\begin{align*}
    d_{sw} &= (q_{i,k} - q_{j,k})^2 \\
    s_{\mu} &= s_{sw} - d_{sw} \\
    f_{sw} &= \sum_{k=0}^{1/2N} F_a(s_{\mu}) \\
    F_{sw}(s_{\mu}) &= \begin{cases} 
        s_{\mu}^2 & (s_{\mu} \geq 0) \\
        0 & (s_{\mu} < 0) 
    \end{cases}
\end{align*}
\]

where \(d_{sw}\) and \(s_{sw}\) represent the actual distance between swarm trajectories and the preset safety distance, respectively; \(q_{i,k}\) and \(q_{j,k}\) represent the control point of the \(i\)-th and \(j\)-th trajectories at time \(k\), respectively. The derivative of the swarm distance penalty function can be obtained as follows:
\[ J_{sw} = \delta f_{sw} = \begin{cases} \frac{1}{2N} \sum_{k=0}^{1/2N} -2(q_{i,k} - q_{j,k}) & (s_\mu \geq 0) \\ 0 & (s_\mu < 0) \end{cases} \] (33)

**Figure 9.** An illustration of swarm distance penalty. The red line represents the \( i \)-th trajectory, while the green line represents the \( j \)-th trajectory. \( q_{i,k} \) and \( q_{j,k} \) represent the control points corresponding to the \( i \)-th and \( j \)-th trajectories, respectively, at time \( k \).

### 3.4.5. Endpoint Arrival Penalty

To ensure that each ASV can reach the endpoint, the last three control points of the B-spline trajectory are set to \( q_{k-2}, q_{k-1}, q_k \), and \( k \in [2, n] \), respectively. Let \( f_t \) be the penalty function for reaching the endpoint; then, one has that

\[ f_t = \left( \frac{1}{6}(q_{k-2} + 4q_{k-1} + q_k) - G \right)^2, \] (34)

where \( G \in \mathbb{R}^2 \) represents the endpoint. The first derivative of \( f_t \) is obtained as

\[ J_t = \frac{\delta f_t}{\delta q_k} = \frac{1}{3}(q_{k-2} + 4q_{k-1} + q_k) - 2G \] (35)

### 3.5. Numerical Optimization

The nonlinear optimization problem has the following two characteristics. Firstly, the total penalty function \( f_{total} \) will be updated in real time based on changes in the environment. Secondly, the quadratic optimization term about dynamic feasibility and obstacle avoidance distance will make \( f_{total} \) closer to the quadratic form, which means that the utilization of Hessian information can significantly improve the speed of solution. However, in the process of trajectory planning for ASVs, solving the inverse Hessian information is prohibitive in real time. Therefore, the limited memory BFGS (L-BFGS) method is adopted to achieve accurate values through a series of iterations. The details of the optimization process are summarized in Algorithm 2.
Algorithm 2 Numerical Optimization

Initialize $x^0, g^0 \leftarrow \nabla f(x^0)$, $B_0 \leftarrow I$, $k \leftarrow 0$

while $||g^k|| > \delta$ do
  
  $d \leftarrow -B^k g^k$
  
  $t \leftarrow$ Lewis Overton line search
  
  $x^{k+1} \leftarrow x^k + t d$\n  
  $g^{k+1} \leftarrow \nabla f(x^{k+1})$
  
  $B^{k+1} \leftarrow \text{LBFGS}(g^{k+1} - g^k, x^{k+1} - x^k)$
  
  $k \leftarrow k + 1$

end while

For an unconstrained optimization problem $\min f(x)$, the iterative updating method for $x$ is similar to Newton’s method. Specifically, the update of $x$ is given as follows:

$$x^{k+1} = x^k - t B^k g^k,$$  \hspace{1cm} (36)

where $B^k$ is updated at every iteration of the LBFGS method as summarized in Algorithm 3, $g^k$ represents the gradient at $x^k$, and $t$ is the step length obtained through the Lewis–Overton line search method, as summarized in Algorithm 4.

Algorithm 3 The L-BFGS algorithm

Initialize $s^k = x^{k+1} - x^k$, $y^k = g^{k+1} - g^k$, $\rho^k = 1 / (y^k s^T d)$, $d \leftarrow g^k$

for $i = k - 1, k - 2, ..., k - m$ do
  
  $\alpha_i \leftarrow \rho^i s^T d$
  
  $d \leftarrow d - \rho^i y^i$

end for

$\gamma \leftarrow \rho^{k-1} y^{k-1} s^T y^{k-1}$

$d \leftarrow d / \gamma$

for $i = k - m, k - m + 1, ..., k - 1$ do
  
  $\beta_i \leftarrow \rho^i y^i s^T d$
  
  $d \leftarrow d + s^i (\alpha_i - \beta_i)$

end for

Algorithm 4 Lewis–Overton line search

Initialize $l \leftarrow 0$, $u \leftarrow +\infty$, $a \leftarrow a_0$

if $S(\alpha)$ fails then
  
  $u \leftarrow \alpha$

else if $C(\alpha)$ fails then
  
  $l \leftarrow \alpha$

else
  
  return $\alpha$

end if

if $u < +\infty$ then
  
  $\alpha \leftarrow (l + u) / 2$

else
  
  $\alpha \leftarrow 2 l$

end if

It is noted that $S(\alpha)$ and $C(\alpha)$ in Algorithm 4 represent strong Wolfe conditions and weak Wolfe conditions, respectively, which are given as follows:

$$\text{strong wolfe conditions : } \begin{cases} f(x^k) - f(x^k + td) \geq -c_1 \cdot t d^T \nabla f(x^k) \\ |d^T \nabla f(x^k + td)| \leq |c_2 d^T \nabla f(x^k)| \end{cases}$$  \hspace{1cm} (37)
weak wolf e conditions : 
\[ \left\{ \begin{array}{l} \left( f(x^k) - f(x^k + td) \right) \geq -c_1 * td T \nabla f(x^k) \\ d T \nabla f(x^k + td) \geq c_2 d T \nabla f(x^k) \end{array} \right. \] 
(38)

where \( c_1 = 10^{-4}, c_2 = 0.9 \).

3.6. Broadcast Network

Once an ASV generates a new trajectory in the complex environment, it will publish the trajectory to other ASVs through a broadcast network. Other ASVs will save the trajectory and generate their own safety trajectory when necessary based on the saved trajectory. Meanwhile, each ASV checks the collision condition when the neighbor’s trajectory is received from the network. If the received trajectory collides with the trajectories of other ASVs, this strategy can solve the problem. In addition, considering the increasing number of ASVs, each ASV should compare its current position with the trajectories received from neighboring ASVs before conducting trajectory planning.

Remark 1. The current algorithm proposed in the article aims at obstacle avoidance and collision avoidance of multiple ASVs in complex static obstacle environments. When dynamic obstacles exist and enter the perception range of the ASV, it will stop and continuously replan until a passable path appears. Therefore, it would still be feasible for online application while meeting unknown moving obstacles. Nevertheless, in the case of unknown moving obstacles, it is necessary for the ASV to have strong braking ability when sailing at a fast speed. In our future work, for the case of unknown moving obstacles, we will try to add dynamic point cloud filtering to complete the reconstruction of the surrounding obstacle environment. Moreover, point cloud recognition and/or clustering algorithms will be added to determine the position and speed of unknown dynamic obstacles. With relevant constraint conditions constructed, the safety of autonomous navigation for multiple ASVs can be guaranteed.

4. Simulation Results

In this section, simulations are provided to illustrate the performance of the proposed distributed trajectory-planning method for multiple ASVs in an unknown environment with lots of static obstacles. Two cases are considered as follows:

Case 1: Swarm trajectory planning for four ASVs. In the simulation, swarm trajectories were planned for four ASVs by various planners in the same scenario. In particular, the proposed method is compared with the enhanced conflict-based search (ECBS) method and the A* + B spline method in terms of sail distance \( (d_{sail}) \), sail time \( (t_{sail}) \), and collision times per ASV.

Figure 10 shows trajectories planned by various planners in the same scenario with the same initial and final positions. It is observed from Table 1 and Figure 10 that all planners can generate collision-free trajectories, while the proposed one needs shorter sail distance and sail time compared to the other two planners. In particular, compared with the A* + B spline method, the proposed method solves the multi-objective optimization problem and achieves optimization subject to obstacle avoidance distance, curve smoothness, and collision avoidance distance between ASVs, allowing multiple ASVs to reach the specified target points in a shorter time and with lower energy loss. Meanwhile, compared with the ECBS method using global planning, the proposed one utilizes multiple local optimal trajectories to form a global trajectory, which better meets the requirements of real-time obstacle avoidance and generates shorter sail distance for short-range navigation. However, local planning using our method and the A* + B spline method generates a global path using multi-segment local trajectories, which is prone to getting stuck in local optima. This is also what we want to solve in the future.
Table 1. Comparisons between different planners.

<table>
<thead>
<tr>
<th>Planner</th>
<th>$d_{sail}$ (m)</th>
<th>$t_{sail}$ (s)</th>
<th>Collision</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECBS</td>
<td>12.6</td>
<td>11.6</td>
<td>0</td>
</tr>
<tr>
<td>A* + B spline</td>
<td>13.6</td>
<td>9.2</td>
<td>0</td>
</tr>
<tr>
<td>Proposed</td>
<td>10.6</td>
<td>8.3</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 10. Trajectories planned by various planners in the same scenario.

Case 2: Swarm trajectory planning for seven ASVs. In this case, we conducted simulations on seven ASVs with a random map generated by the Berlin algorithm, where the generated trajectories are shown in Figure 11. The generated trajectories of seven ASVs in the xy-plane are shown in Figure 12. The evolution of the velocity and acceleration optimization terms, the navigation speed and acceleration of ASVs do not exceed 2.5 m/s and 3 m/s$^2$, respectively. The distances of each ASV to the goals, the distances of each ASV to the closest ASV, and the distances of each ASV to the closest obstacle are shown in Figures 15–17, respectively. It is noted that the safety threshold of distance between each ASV and the closest ASV is set to 0.4 m, and the safety threshold of distance between each ASV and the closest obstacle is set to 0.5 m. At the same time, due to the existence of boundary constraints, ASVs can reduce their speed and acceleration to 0 when they reach the goals. Moreover, when the target arrival constraint exists, the ASVs can reach the goals accurately.

Figure 11. Seven ASVs conduct autonomous navigation in a simulated environment. (a–c) represent the top view, left-side view, and right-side view, respectively.

To further verify the effectiveness of the proposed method, we conducted multiple simulations based on different numbers of ASVs under different obstacle coverage ranges and different initial conditions. Setting the obstacle coverage rate to 50% and 75%, setting obstacle shape roughness to 10% and 15%, and setting the number of ASVs to 5, 7, and 9, respectively, different simulations were conducted, as shown in Figure 18. As a result, one can see that the proposed method performs well under different obstacle coverage ranges and different initial conditions.
Figure 12. The trajectories of seven ASVs in the xy-plane.

Figure 13. The speed of seven ASVs with each type of line corresponding to one ASV.

Figure 14. The acceleration of seven ASVs with each type of line corresponding to one ASV.
Figure 15. The distances of each ASV to the goals with each type of line corresponding to one ASV.

Figure 16. The distances of each ASV to the closest ASV with each type of line corresponding to one ASV.

Figure 17. The distances of each ASV to the closest obstacle with each type of line corresponding to one ASV.
Figure 18. Autonomous navigation of different numbers of ASVs under different obstacle coverage ranges and different initial conditions (Left: Top View, Right: Left-side View).

5. Conclusions

In this paper, a swarm trajectory-planning method is proposed for multiple ASVs using distributed asynchronous communication. The issues of curve smoothness, dynamic feasibility, collision avoidance between ASVs, and obstacle avoidance are transformed into a non-constrained nonlinear optimization problem. Efficient solutions are proposed for generating smooth and collision-free trajectories that ASVs can track. Since real-time local planning and collision-detection strategies have been adopted, it is effective to reduce the total navigation time and avoid obstacles in a marine environment. In the future, we will further address the issue of formation of ASVs subject to multiple constraints and static obstacles.

Author Contributions: Conceptualization, investigation, methodology, validation, formal analysis, A.W.; resources, data curation, writing—original draft preparation, L.L.; writing—review and editing, H.W.; visualization, supervision, B.H.; project administration, funding acquisition, Z.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Key R&D Program of China under Grant 2022ZD0119902, in part by the Key Basic Research of Dalian under Grant 2023JJ11CG008, in part by the National Natural Science Foundation of China under Grant 51979020, in part by the Top-notch Young Talents Program of China under Grant 36261402, in part by the Doctoral Scientific Research Foundation of Liaoning Province under Grant 2023-BS-155, in part by the Basic Scientific Research Project of Higher Education Department of Liaoning Province under Grant LJKZ0044, in part by the National Natural Science Foundation of China under Grant 62203081, in part by the Postdoctoral
Research Foundation of China under Grant 2022M720628, and in part by Doctoral Science Research Foundation of Liaoning under Grant 2022-BS-097.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Bing Han was employed by the company Shanghai Ship and Shipping Research Institute Co., Ltd.; The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References


24. Zhang, G.; Huang, C.; Li, J.; Zhang, X. Constrained coordinated path-following control for underactuated surface vessels with the disturbance rejection mechanism. *Ocean. Eng.* 2020, **196**, 106725. [CrossRef]

25. Liu, L.; Wang, D.; Peng, Z. Coordinated path following of multiple underactuated marine surface vehicles along one curve. *ISA Trans.* 2016, **64**, 258–268. [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.