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

Anqing Wang , Longwei Li , Haoliang Wang , Bing Han , Zhouhua Peng *

Posted Date: 26 December 2023

doi: 10.20944/preprints202312.1824.v1

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

Article

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; anqingwang@dlmu.edu.cn;2495328028@qq.com;zhpeng@dlmu.edu.cn
[2] College of Marine Engineering, Dalian Maritime University; haoliang.wang12@dlmu.edu.cn
[3] Shanghai Ship and Shipping Research Institute Co., Ltd; han.bing@coscoshipping.com
* 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. 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

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 [1–9]. Among various applications of ASVs, 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 literature [10–17]. Specifically, in [10], a temporal-logic based ASV path planning method is employed, which enables the ASV to pass through heavy harbour traffic to an intended destination in a collision-free manner. In [11], an evolutionary-based path planning approach is proposed for an ASV to accomplish environmental monitoring tasks. In [12], an extension of 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 [13] for energy-optimized ASV trajectory planning with experimental validation. In [14], a novel receding horizon multi-objective planner is developed for an ASV performing path planning in complex urban waterways. In [15], an essential visibility graph approach is proposed to generate optimal paths for an ASV with real-time collision avoidance. Further in [16], the particle swarm optimization algorithm together with the visibility graph is applied to the ASV path planning problem among complex shorelines and spatio-temporal environmental forces. Recently in [17], 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 [10–17] are dedicated to path planning of a single ASV.

 doi:10.20944/preprints202312.1824.v1

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 [18–28]. Therefore, 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 the RRT method [29], the satellite maps [30], the priority target assignment method [31], the voronoi-visibility roadmap [32] and the genetic algorithm [33]. However, due to the uncertainties of obstacles, and limitations in sensor range and communication bandwidth, trajectory planning of multiple ASVs in an unknown, especially in obstacle-rich marine environment still presents a great challenge. In fact, to the best of the our knowledge, as for autonomous navigation of multiple ASVs, few work is available to construct 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 is 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 non-linear 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 takes the following features:

- In contrast to the existing trajectory planning methods dedicated to a single ASV [10–17], the proposed method is capable of solving the swarm trajectory planning problem of multiple ASVs. By utilizing local onboard sensor information, ASVs can accomplish autonomous navigation requiring no external localization and computation or a pre-built map in unknown 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.

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 $\lambda_{\max}(\cdot)$ represents 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:

$$\begin{cases} \dot{x} = u\cos\psi - v\sin\psi \\ \dot{y} = u\sin\psi + v\cos\psi \\ \dot{\psi} = r \end{cases} \tag{1}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

where $[x, y]$ and ψ 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 trajectory planning task, it is assumed that $v = 0$ such that

$$\begin{cases} \dot{x} = u\cos\psi \\ \dot{y} = u\sin\psi \\ \dot{\psi} = r \end{cases} \tag{2}$$

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.

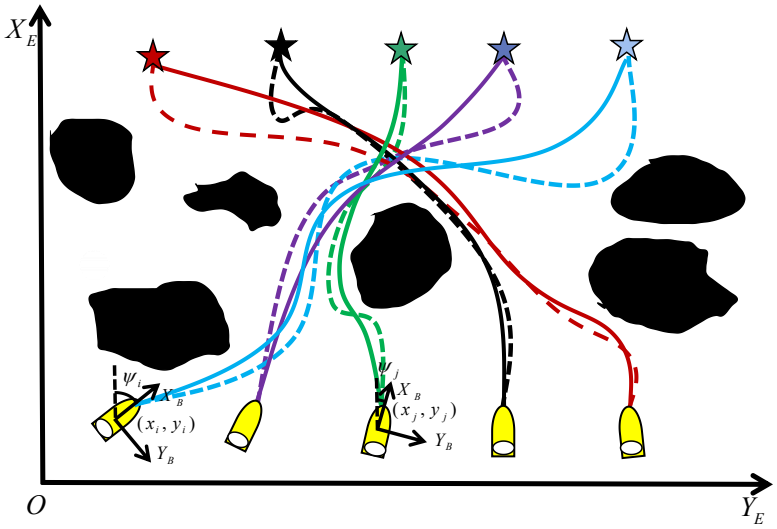

Figure 1. An illustration of trajectory planning for multiple ASVs to reach the designated locations. The dashed lines represent trajectories that have not yet been optimized, the solid lines indicate the optimized trajectories, and the black obstacles represent the complex static environments.

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.

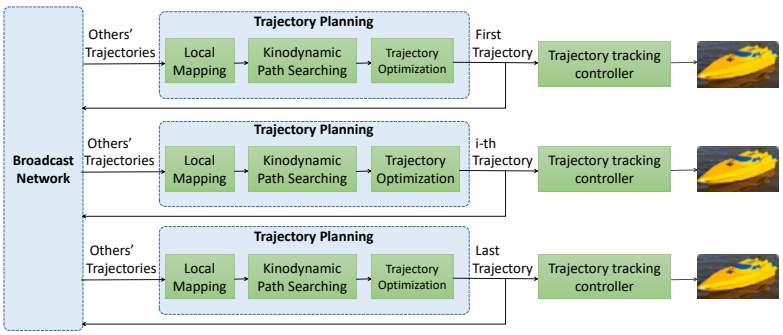

Figure 2. The trajectory planning framework for multiple ASVs.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

3.1. Construction of Occupancy Grid Map

The sensing radius of the sensor is set to r_{id} and take 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 is set to $\alpha \in \mathbb{R}$, then the constraints for point cloud acquisition are given as

$$\begin{cases} \dfrac{z_{tid} - z_{id}}{r_{id}} \leq tan(\alpha) \\ \begin{bmatrix} x_{tid} - x_{id} \\ y_{tid} - y_{id} \\ z_{tid} - z_{id} \end{bmatrix} \begin{bmatrix} 1 - 2(y_{qid}^2 + z_{qid}^2) \\ 2(x_{qid}y_{qid} + z_{qid}w_{qid}) \\ 2(x_{qid}z_{qid} - y_{qid}w_{qid}) \end{bmatrix} \geq 0 \end{cases} \tag{3}$$

According to (3), a dense point cloud map of the ASVs' forward direction can be obtained. To represent the presence of obstacles, an occupied grid map is constructed by partitioning the point cloud into grids, with each being one of two states: filled or empty. For a grid m_i in occupancy grid map, the probability of the grid state being occupied is denoted as $p(m_i)$ and the probability of the grid state being free is denoted as $p(\overline{m_i})$. The t-th observation is set to z_t. To calculate the posterior probability of the grid state based on existing observations, $p(m_i|z_{1:t})$ and $p(\overline{m_i}|z_{1:t})$ should be calculated. A grid is occupied if $p(m_i|z_{1:t}) \geq \kappa$ with κ being a preset threshold. The probability of the grid state being occupied can be specifically denoted as

$$p(m_i|z_{1:t}) = \frac{p(z_t|z_{1:t-1}, m_i)p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{4}$$

and the t-th observation is obtained as follows:

$$p(m_i|z_t) = \frac{p(z_t|m_i)p(m_i)}{p(z_t)} \tag{5}$$

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

$$p(z_t|z_{1:t-1}, m_i) = p(z_t|m_i) \tag{6}$$

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

$$p(m_i|z_{1:t}) = \frac{p(m_i|z_t)p(z_t)p(m_i|z_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1})} \tag{7}$$

and

$$p(\overline{m_i}|z_{1:t}) = \frac{p(\overline{m_i}|z_t)p(z_t)p(\overline{m_i}|z_{1:t-1})}{p(\overline{m_i})p(z_t|z_{1:t-1})} \tag{8}$$

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

$$\begin{cases} l_t(m_i) = \log \dfrac{p(m_i|z_{1:t})}{p(\overline{m_i}|z_{1:t})} \\ l_{t-1}(m_i) = \log \dfrac{p(m_i|z_{1:t-1})}{p(\overline{m_i}|z_{1:t-1})} \end{cases} \tag{9}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

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

$$l_t(m_i) = \log \frac{p(m_i|z_t)}{p(\overline{m_i}|z_t)} - \log \frac{p(m_i)}{p(\overline{m_i})} + l_{t-1}(m_i) \tag{10}$$

Substituting (5) into (10) yields

$$l_t(m_i) = \log \frac{p(z_t|m_i)}{p(z_t|\overline{m_i})} + l_{t-1}(m_i) \tag{11}$$

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

$$p(m_i|z_{1:t}) = \frac{a^{l_t(m_i)}}{1 + a^{l_t(m_i)}} \tag{12}$$

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.

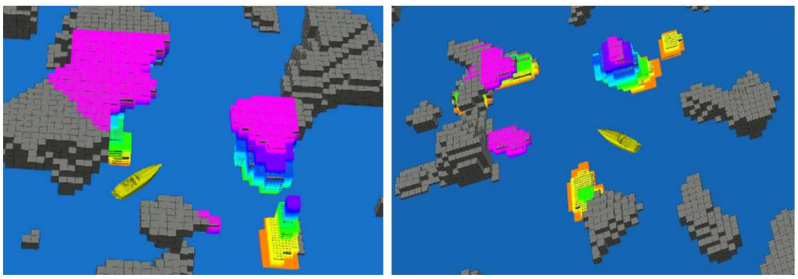

Figure 3. An illustration of the occupied grid map in a complex and obstacle-rich environment. The colored point cloud part represents the environmental information perceived by the ASV, and the gray part represents the unknown obstacle environment.

3.2. Kinodynamic Path Searching

Inspired by the hybrid A* search proposed for autonomous vehicles in [34], 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.

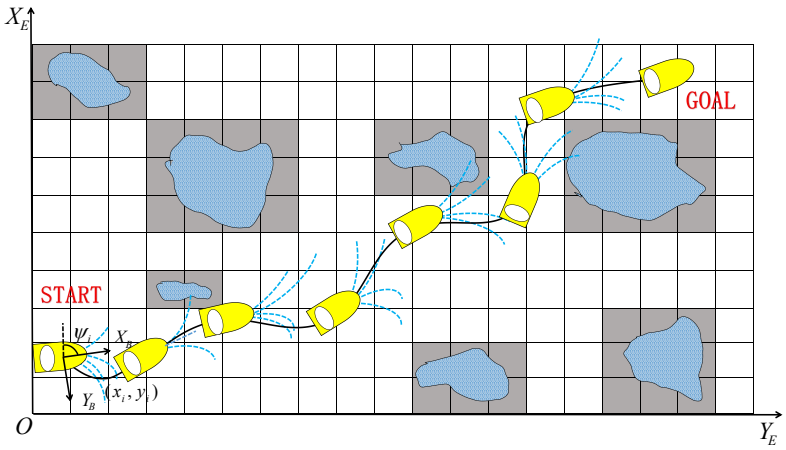

Figure 4. A schematic diagram of the kinodynamic path searching. The blue dashed line represents the motion primitives obtained by expanding the state point by Equation (14), and the black curve represents the optimal curve selected based on the heuristic function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

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, and the g_c and f_c cost (Subection 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 goal successfully or the AnalyticExpand(·) succeeds. Details of the kinodynamic path searching are shown in Algorithm 1.

Algorithm 1 Kinodynamic Path Searching

```
while openlist.empty() do
  openlist pop n_c
  closelist insert n_c
  if Reachgoal(n_c) or AnalyticExpand(n_c) then
    return Path()
  end if
  expandnodes ← NodeExpand(n_c)
  nodes ← NodePrune(expandnodes)
end while
for each n_i ∈ nodes do
  if closelist contain n_i and CheckCollision(n_i) then
    g_t ← n_c.g_c + CurrentCost(n_i)
    if openlist contain n_i then
      openlist add n_i
      if g_t ≥ n_i.g_c then
        continue
      end if
    end if
  end if
end for
n_i.parent ← n_c, n_i.g_c ← g_t
n_i.f_c ← n_i.g_c + h(n_i)
```

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_i, 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:

$$\begin{cases} r_i \in \{r_0, r_1, ...r_m\} \\ T_j \in \{T_0, T_1, ..., T_n\} \\ u_k \in \{-u_{max}, u_{max}\} \end{cases} \tag{13}$$

where r_i, $i = 1, 2, \cdots, m$ 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 $p_t = [x_t, y_t, \psi_t] \in \mathbb{R}^3$ 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

$$\begin{cases} x_t = x_0 + \int_0^{T_j} u_k \cos(\psi_t) d\tau \\ y_t = y_0 + \int_0^{T_j} u_k \sin(\psi_t) d\tau \\ \psi_t = \psi_0 + \int_0^{T_j} r_i d\tau \end{cases} \tag{14}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

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*. 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:

$$\begin{cases} J(T) = T + \sum\limits_{\mu \in \{x,y\}} (\frac{1}{3}\alpha_\mu^2 T^3 + \alpha_\mu \beta_\mu T^2 + \beta_\mu^2 T) \\ \begin{bmatrix} \alpha_\mu \\ \beta_\mu \end{bmatrix} = \begin{bmatrix} -\frac{12}{T^3} & \frac{6}{T^2} \\ \frac{6}{T^2} & -\frac{2}{T} \end{bmatrix} \begin{bmatrix} p_{\mu f} - p_{\mu 0} - v_{\mu 0} T \\ v_{\mu f} - v_{\mu 0} \end{bmatrix} \end{cases} \tag{15}$$

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, α_μ and β_μ 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 $T_{\min}$, and $J(T_{\min})$ 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 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

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 ψ_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(·) 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.

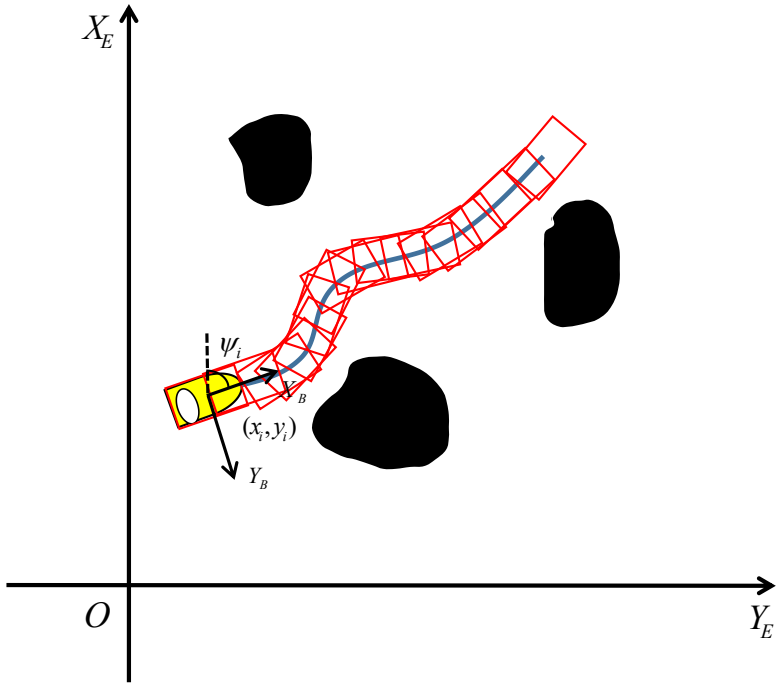

Figure 5. An illustration of the swath along trajectory. The red rectangle represents the area occupied by the ASV. The black area represents complex static obstacles.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

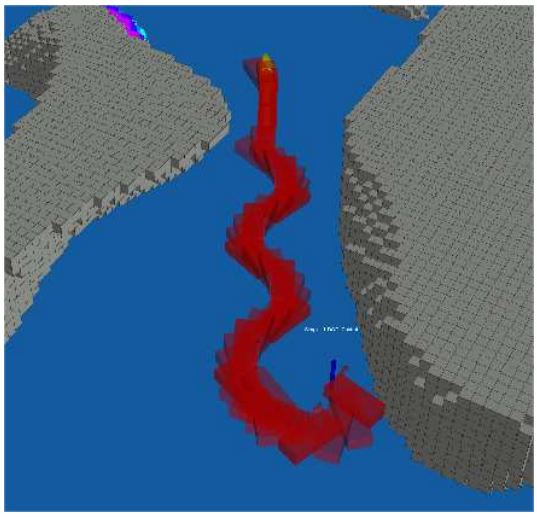

Figure 6. Schematic diagram of collision checking during autonomous navigation of an ASV. The red cube area represents the area occupied by the ASV, while the gray area represents complex static obstacles

3.2.4. Analytic Expansion

Generally speaking, it is difficult for discretized input to reach the end point accurately. To this end, an analytic expansion scheme AnalyticExpand(·) in Algorithm 1 is induced to speed up the trajectory searching. When the current node $p_{\mu 0}$ is close to the end point $p_{\mu f}$, the same approach used in Subsection 3.2.2 is directly applied to compute a trajectory from $p_{\mu 0}$ to $p_{\mu f}$ 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 can not be strictly guaranteed, as kinodynamic path searching ignores the distance information of obstacles and does not consider curve smoothness. Therefore, B-spline curve is used in this section to fit the path curve.

3.3.1. Cubic Uniform B-Spline

Let $q = \{q_0, q_1, ..., q_n\}$ be the control points obtained from kinodynamic path searching, and $\Theta = \{\theta_0, \theta_1, ..., \theta_m\}$ 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, $n + 1$ control points q and Θ. A cubic B-spline trajectory, used to fit the above control points, is parameterized by θ. 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 $\theta \in [\theta_i, \theta_{i+1})$, $i = 0, 1, \cdots, m - 1$, the four control points of a cubic uniform B-spline trajectory set within the knot vector θ are $q_{k-3}, q_{k-2}, q_{k-1}, q_k$, where $3 \leq k \leq n$. To construct B-spline curves, B-spline basis functions is

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

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_k-order basis function are given as follows:

$$\begin{cases} N_{k,0} = \begin{cases} 0 & if\ \theta_i \le \theta < \theta_{i+1} \\ 1 & otherwise \end{cases} \\ N_{k,p} = \dfrac{\theta - \theta_i}{\theta_{i+p} - \theta_i} N_{k,p-1}(\theta) + \dfrac{\theta_{i+p+1} - \theta}{\theta_{i+p+1} - \theta_{i+1}} N_{k+1,p-1}(\theta) \end{cases} \tag{16}$$

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

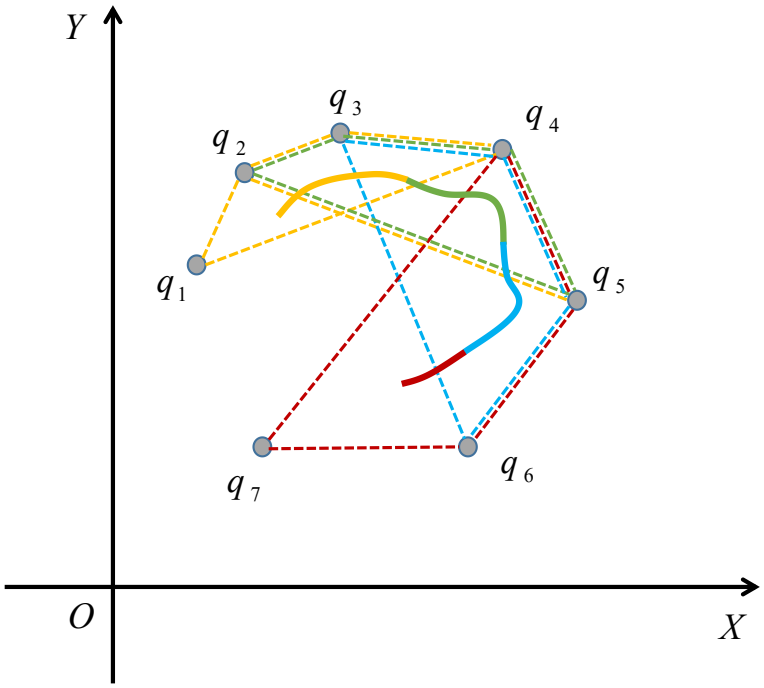

Figure 7. The convex hull property of B-spline curves with gray points representing the control points. Each segment of the curve is included in the convex hull constructed by every four control points.

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

$$\begin{cases} N_{k-3,3} = \dfrac{1}{6}(1 - s(\theta))^3 \\ N_{k-2,3} = \dfrac{1}{2}s(\theta)^3 - s(\theta)^2 + \dfrac{2}{3} \\ N_{k-1,3} = -\dfrac{1}{2}s(\theta)^3 + \dfrac{1}{2}s(\theta)^2 + \dfrac{1}{2}s(\theta) + \dfrac{1}{6} \\ N_{k,3} = \dfrac{1}{6}s(\theta)^3 \end{cases} \tag{17}$$

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

$$\begin{cases} p(s(\theta)) = n_k Q_k \\ n_k = \begin{bmatrix} N_{k-3,3} & N_{k-2,3} & N_{k-1,3} & N_{k,3} \end{bmatrix} \\ Q_k = \begin{bmatrix} q_{k-3} & q_{k-2} & q_{k-1} & q_k \end{bmatrix}^T \end{cases} \tag{18}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

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 $\{q_0, q_1, ..., q_n\}$ and basis functions $N_{k,p}$ defined above, a p-degree B-Spline $C(\theta)$ can be obtained as follows:

$$C(\theta) = \sum_{k=0}^{n} N_{k,p}(\theta) q_k \tag{19}$$

with its first-order and second-order derivatives being

$$\begin{cases} \dot{C}(\theta) = \sum_{k=0}^{n-1} N_{k+1,p-1}(\theta) v_k \\ \ddot{C}(\theta) = \sum_{k=0}^{n-2} N_{k+2,p-2}(\theta) a_k \end{cases} \tag{20}$$

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

$$\frac{dN_{k,p}(\theta)}{d\theta} = \frac{pN_{k,p-1}(\theta)}{\theta_{i+p} - \theta_i} + \frac{-pN_{k+1,p-1}(\theta)}{\theta_{i+p+1} - \theta_{i+1}} \tag{21}$$

Substituting (21) into (20) yields

$$\dot{C}(\theta) = \sum_{k=0}^{n-1} N_{k+1,p-1}(\theta) \left(\frac{pq_{k+1}}{\theta_{i+p+1} - \theta_{i+1}} + \frac{-pq_k}{\theta_{i+p+1} - \theta_{i+1}} \right) \tag{22}$$

Thus, the control points of the B-spline $C(\theta)$'s first derivative v_k can be computed by

$$v_k = \frac{pq_{k+1}}{\theta_{i+p+1} - \theta_{k+1}} + \frac{-pq_k}{\theta_{i+p+1} - \theta_{i+1}} \tag{23}$$

Since the B-spline trajectory used in Subsection 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 B-spline trajectory can be further derived as follows:

$$\begin{cases} a_k = \frac{v_{k+1} - v_k}{\Delta\theta} = \frac{1}{\Delta\theta^2}(q_{k+2} - 2q_{k+1} + q_k) \\ j_k = \frac{a_{k+1} - a_k}{\Delta\theta} = \frac{1}{\Delta\theta^3}(q_{k+3} - 3q_{k+2} + 3q_{k+1} - q_k) \end{cases} \tag{25}$$

3.4. Nonlinear Optimization

For the B-spline trajectory defined by a set of control points $\{q_0, q_1, ..., q_n\}$ in Subsection 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:

$$f_{total} = \lambda_1 f_{sm} + \lambda_2 f_{ob} + \lambda_3 (f_v + f_a) + \lambda_4 f_{sw} + \lambda_5 f_t, \tag{26}$$

where f_{sm} and f_{ob} are optimization terms for trajectory smoothness and collision distance, respectively. f_v and f_a are constrained optimization term of velocity and acceleration, respectively. f_{sw} and f_t are optimization terms for collision distance between ASVs and end point arrival, respectively. $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ and λ_5 are designed weight coefficients.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

3.4.1. Smoothness Penalty

The smoothness cost f_{sm} is defined by a function that uses 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_{sm} is defined as follows:

$$f_{sm} = \sum_{k=2}^{n-3} ||j_k||^2 = \sum_{k=2}^{n-3} ||q_{k+2} - 3q_{k+1} + 3q_k - q_{k-1}||^2 \tag{27}$$

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 though the obstacle in Figure 8. Therefore, the naive trajectory needs to be optimized by A* algorithm to obtain a collision-free trajectory Γ.

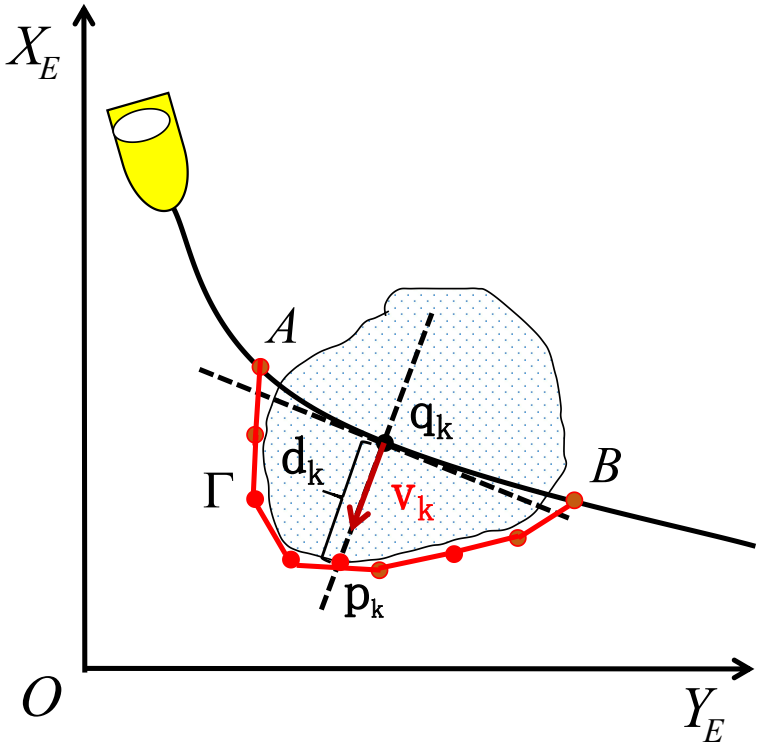

Figure 8. The black solid line passing though the obstacle represents the naive B-spline trajectory to be optimized. The red solid line represents the edge of the obstacle obtained by A* search.

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:

$$\begin{cases} c_k = s_f - d_k \\ F_c = \begin{cases} 0 & c_k \le 0 \\ c_k^3 & 0 < c_k \le s_f \\ 3s_f c_k^2 - 3s_f^2 c_k + s_f^3 & s_f < c_k \end{cases} \end{cases} \tag{28}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

The collision penalty function is denoted as f_c and its derivative can be obtained as follows:

$$\begin{cases} f_c = \sum_{k=2}^{N-3} F_c(q_k) \\ J_c = \frac{\delta f_c}{\delta q_k} = \begin{cases} -3c_k v_k & c_k \leq s_f \\ -6s_f c_k + 3s_f^2 & s_f < c_k \end{cases} \end{cases} \tag{29}$$

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 $\{q_0, q_1, ..., q_n\}$. 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:

$$\begin{cases} f_v = \sum_{k=3}^{N-3} F_v(v_k) \\ F_v(v_k) = \begin{cases} 0 \ \ (0 \leq v_k \leq v_{max}) \\ (v_k - v_{max})^3 \ \ (v_{max} \leq v_k \leq v_j) \\ a_1 v_k^2 + b_1 v_k + c_1 \ \ (v_k \geq v_j) \end{cases} \\ f_a = \sum_{k=4}^{N-3} F_a(a_k) \\ F_a(a_k) = \begin{cases} 0 \ \ (0 \leq a_k \leq a_{max}) \\ (a_k - a_{max})^3 \ \ (a_{max} \leq a_k \leq a_j) \\ a_2 a_k^2 + b_2 a_k + c_2 \ \ (a_k \geq a_j) \end{cases} \end{cases} \tag{30}$$

where $a_1, b_1, c_1, a_2, b_2, c_2$ are design parameters to ensure the second-order continuous differentiability of F_v and F_a. v_{max} and a_{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:

$$\begin{cases} \frac{\delta v_k}{\delta q_k} = -\frac{1}{\Delta\theta} \\ \frac{\delta f_{v_k}}{\delta q_k} = \begin{cases} \sum_{k=3}^{N-3} 3\frac{\delta v_k}{\delta q_k}(v_k - v_{max})^2 \ \ (v_{max} \leq v_k \leq v_j) \\ \sum_{k=3}^{N-3} 2\frac{\delta v_k}{\delta q_k} a_2 v_k + \frac{\delta v_k}{\delta q_m} b_2 \ \ (v_k \geq v_j) \end{cases} \\ \frac{\delta a_k}{\delta q_k} = \frac{1}{\Delta\theta^2} \\ \frac{\delta f_{a_k}}{\delta q_k} = \begin{cases} \sum_{k=3}^{N-3} 3\frac{\delta a_k}{\delta q_k}(a_k - a_{max})^2 \ \ (a_{max} \leq a_k \leq a_j) \\ \sum_{k=3}^{N-3} 2\frac{\delta a_k}{\delta q_k} a_2 a_k + \frac{\delta a_\mu}{\delta q_k} b_2 \ \ (a_k \geq a_j) \end{cases} \end{cases} \tag{31}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

3.4.4. Swarm Distance Penalty

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

$$\begin{cases} d_{sw} = (q_{i,k} - q_{j,k})^2 \\ s_\mu = s_{sw} - d_{sw} \\ f_{sw} = \sum_{k=0}^{1/2N} F_s(s_\mu) \\ F_{sw}(s_\mu) = \begin{cases} s_\mu^2 \ \ (s_\mu \geq 0) \\ 0 \ \ (s_\mu < 0) \end{cases} \end{cases} \tag{32}$$

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:

$$J_{sw} = \frac{\delta f_{sw}}{\delta q_k} = \begin{cases} \sum_{k=0}^{1/2N} -2(q_{i,k} - q_{j,k}) \ \ (s_\mu \geq 0) \\ 0 \ \ (s_\mu < 0) \end{cases} \tag{33}$$

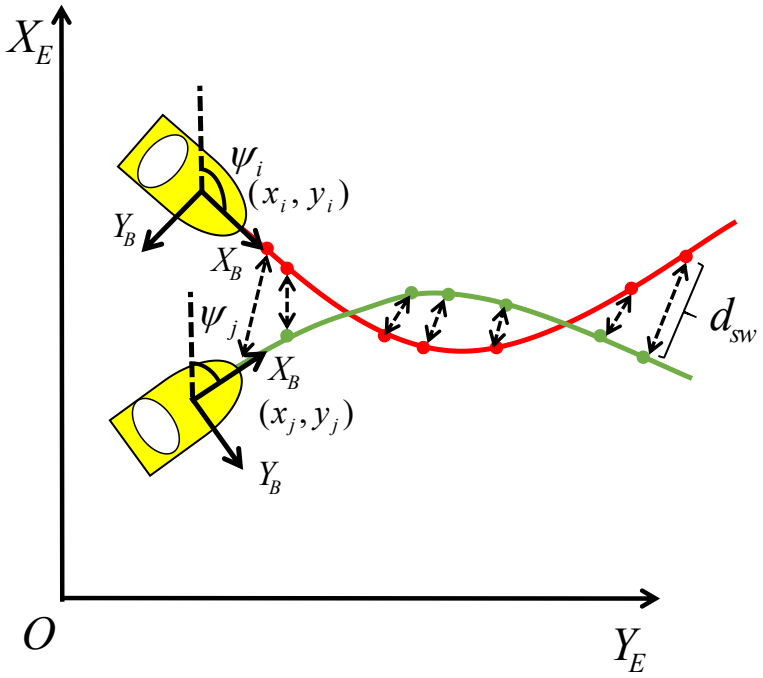

Figure 9. An illustration of swarm distance penalty. The red line represents the i-th trajectory, while the green line represents the j-th trajectory. $q_{i,k}$ and $q_{j,k}$ represent the control points corresponding to the i-th and j-th trajectories respectively at time k.

3.4.5. End Point Arrival Penalty

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

$$f_t = (\frac{1}{6}(q_{k-2} + 4q_{k-1} + q_k) - G)^2, \tag{34}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

where $G \in \mathbb{R}^2$ represents the end point. The first derivative of f_t is obtained as

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

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 is 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*
  $x^{k+1} \leftarrow x^k + td$
  $g^{k+1} \leftarrow \nabla f(x^{k+1})$
  $B^{k+1} \leftarrow LBFGS(g^{k+1} - g^k, x^{k+1} - x^k)$
  $k \leftarrow k + 1$
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:

$$x^{k+1} = x^k - tB^k g^k, \tag{36}$$

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

for $i = k-1, k-2, ..., k-m$ **do**
  $\alpha^i \leftarrow \rho^i {s^i}^T d$
  $d \leftarrow d - \rho^i y^i$
end for
$\gamma \leftarrow \rho^{k-1} {y^{k-1}}^T y^{k-1}$
$d \leftarrow d/\gamma$
for $i = k-m, k-m+1, ..., k-1$ **do**
  $\beta \leftarrow \rho^i {y^i}^T d$
  $d \leftarrow d + s^i(\alpha^i - \beta)$
end for

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

Algorithm 4 Lewis Overton line search

Initialize $l \leftarrow 0, u \leftarrow +\infty, \alpha \leftarrow \alpha_0$
  if $S(\alpha) fails$ **then**
    $u \leftarrow \alpha$
  else if $C(\alpha) fails$ **then**
    $l \leftarrow \alpha$
  else
    return α
  end if
  if $u < +\infty$ **then**
    $\alpha \leftarrow (l+u)/2$
  else
    $\alpha \leftarrow 2l$
  end if

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

$$strong\ wolfe\ conditions: \begin{cases} f(x^k) - f(x^k + td) \geq -c_1 * td^T \nabla f(x^k) \\ |d^T \nabla f(x^k + td)| \leq |c_2 d^T \nabla f(x^k)| \end{cases} \tag{37}$$

$$weak\ wolfe\ conditions: \begin{cases} f(x^k) - f(x^k + td) \geq -c_1 * td^T \nabla f(x^k) \\ d^T \nabla f(x^k + td) \geq c_2 d^T \nabla f(x^k) \end{cases} \tag{38}$$

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 collision condition when 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.

4. Simulation Results

In this section, a simulation is provided to illustrate the performance of the proposed distributed trajectory planning method for multiple ASVs in an unknown environment with lots of static obstacles. The proposed method is compared with the enhanced conflict based search (ECBS) method, in terms of sail distance (d_{sail}), sail time (t_{sail}), and collision times per ASV.

In the simulation, trajectories was planned for four ASVs by various planners in the same scenario. It is observed from Table 1 and Figure 10 that the proposed method generates collision-free trajectories with shorter sail distance and sail time, as composed to the ECBS method.

Table 1. Comparison between different planners.

planner	$d_{sail}(m)$	$t_{sail}(s)$	**collision**
ECBS	12.6	11.6	0
Proposed	10.6	8.3	0

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

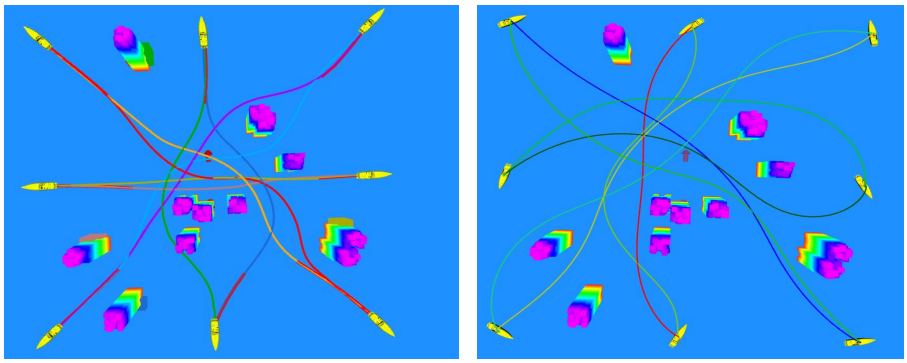

Figure 10. Trajectories planned by various planners in the same scenario.

Meanwhile, we conducted simulation experiments on seven ASVs in a random map generated by the Berlin algorithm, where the generated trajectories are shown in Figure 11. The evolution of the velocity and acceleration of the ASVs are shown in Figure 12 and Figure 13, respectively. The distance between seven ASVs and the goals are shown in Figure 14. Due to the existence of speed and acceleration optimization terms, the navigation speed and acceleration of ASVs do not exceed 2.5 m/s and 3m/s^2, respectively. 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.

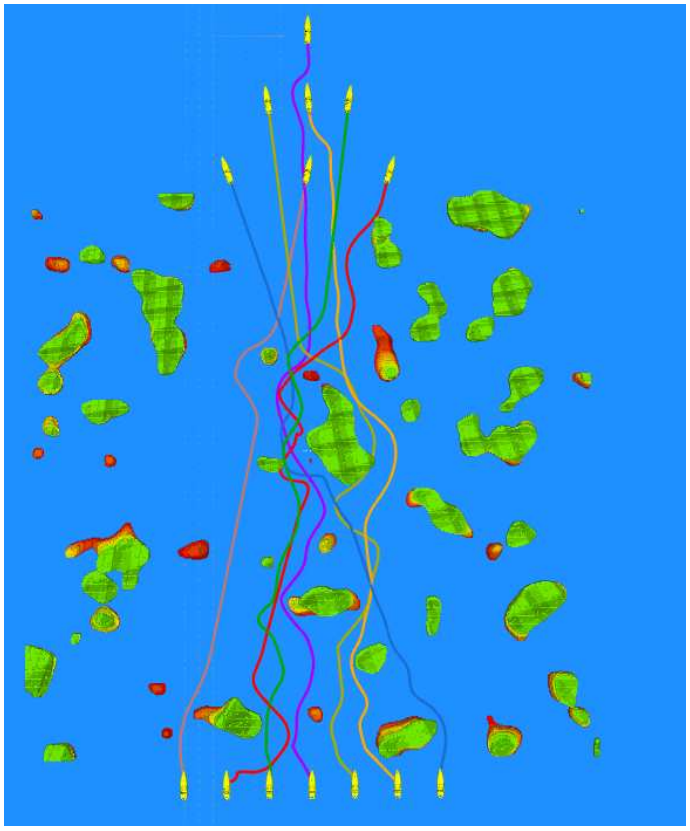

Figure 11. Seven ASVs conduct autonomous navigation in a simulated environment.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2023 doi:10.20944/preprints202312.1824.v1

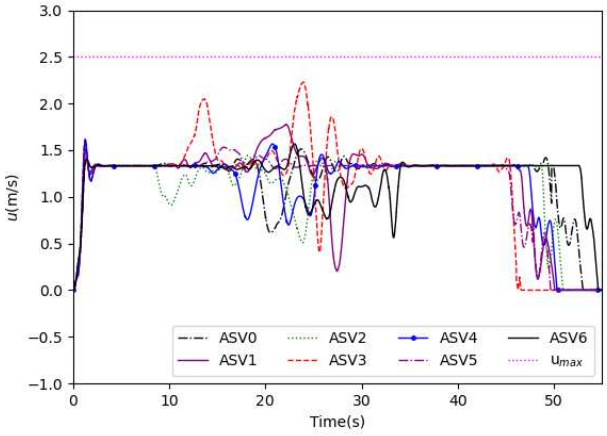

Figure 12. The speed of seven ASVs, with each type of line corresponding to one ASV.

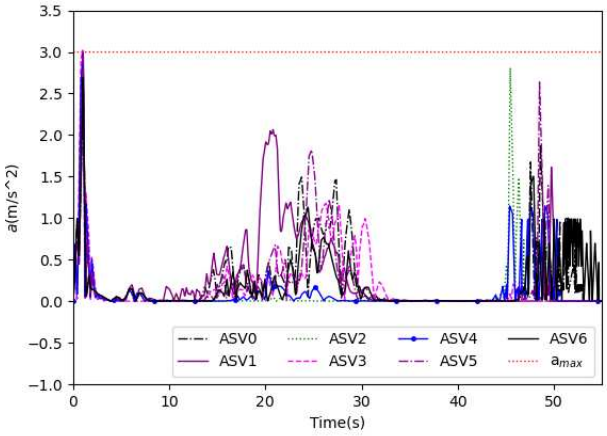

Figure 13. The acceleration of seven ASVs, with each type of line corresponding to one ASV.

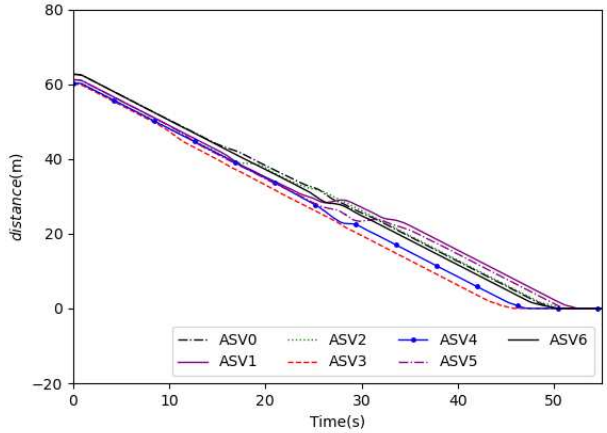

Figure 14. The distance between seven ASVs and the goals, with each type of line corresponding to one ASV.

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 such 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 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, Anqing Wang; resources, data curation, writing—original draft preparation, Longwei Li; writing—review and editing, Haoliang Wang; visualization, supervision, Bing Han; project administration, funding acquisition, Zhouhua Peng. 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, 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 will be made available on request.

References

1. Peng, Z.; Gu, N.; Zhang, Y.; Liu, Y.; Wang, D.; Liu, L. Path-guided time-varying formation control with collision avoidance and connectivity preservation of under-actuated autonomous surface vehicles subject to unknown input gains. *Ocean Engineering* **2019**, *191*, 106501.
2. Peng, Z.; Wang, D.; Li, T.; Han, M. Output-Feedback Cooperative Formation Maneuvering of Autonomous Surface Vehicles With Connectivity Preservation and Collision Avoidance. *IEEE Transactions on Cybernetics* **2020**, *50*, 2527–2535.
3. Wang, D.; Huang, J. Adaptive neural network control for a class of uncertain nonlinear systems in pure-feedback form. *Automatica* **2002**, *38*, 1365–1372.
4. Peng, Z.; Wang, D.; Zhang, H.; Sun, G. Distributed neural network control for adaptive synchronization of uncertain dynamical multiagent systems. *IEEE Transactions on Neural Networks and Learning Systems* **2014**, *25*, 1508–1519.
5. Zhang, G.; Han, J.; Li, J.; Zhang, X. Distributed Robust Fast Finite-Time Formation Control of Underactuated ASVs in Presence of Information Interruption. *Journal of Marine Science and Engineering* **2022**, *10*, 1775.
6. Jing, Q.; Wang, H.; Hu, B.; Liu, X.; Yin, Y. A universal simulation framework of shipborne inertial sensors based on the ship motion model and robot operating system. *Journal of Marine Science and Engineering* **2021**, *9*, 900.
7. Wang, H.; Yin, Y.; Jing, Q. Comparative Analysis of 3D LiDAR Scan-Matching Methods for State Estimation of Autonomous Surface Vessel. *Journal of Marine Science and Engineering* **2023**, *11*, 840.
8. Lee, D.; Woo, J. Reactive Collision Avoidance of an Unmanned Surface Vehicle through Gaussian Mixture Model-Based Online Mapping. *Journal of Marine Science and Engineering* **2022**, *10*, 472.
9. Veitch, E.; Alsos, O.A. Human-centered explainable artificial intelligence for marine autonomous surface vehicles. *Journal of Marine Science and Engineering* **2021**, *9*, 1227.
10. Izzo, P.; Veres, S.M. Intelligent planning with performance assessment for Autonomous Surface Vehicles. In Proceedings of the OCEANS 2015-Genova. IEEE, 2015, pp. 1–6.

11. Arzamendia, M.; Gregor, D.; Reina, D.; Toral, S.; Gregor, R. Evolutionary path planning of an autonomous surface vehicle for water quality monitoring. In Proceedings of the 2016 9th International Conference on Developments in eSystems Engineering (DeSE). IEEE, 2016, pp. 245–250.
12. Willners, J.S.; Petillot, Y.R.; Patron, P.; Gonzalez-Adell, D. Kinodynamic Path Planning for Following and Tracking Vehicles. In Proceedings of the OCEANS 2018 MTS/IEEE CHARLESTON. IEEE Charleston, 2018. Conference on OCEANS MTS/IEEE Charleston, Charleston, SC, OCT 22-25, 2018.
13. Bitar, G.; Martinsen, A.B.; Lekkas, A.M.; Breivik, M. Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments. *IEEE ACCESS* **2020**, *8*, 199953–199969.
14. Shan, T.; Wang, W.; Englot, B.; Ratti, C.; Rus, D. A Receding Horizon Multi-Objective Planner for Autonomous Surface Vehicles in Urban Waterways. *2020 59th IEEE Conference on Decision and Control (CDC)* **2020**, pp. 4085–4092.
15. 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). IEEE, 2021, pp. 400–405.
16. Krell, E.; King, S.A.; Garcia Carrillo, L.R. Autonomous Surface Vehicle energy-efficient and reward-based path planning using Particle Swarm Optimization and Visibility Graphs. *Appl. Ocean Res.* **2022**, *122*, 103125.
17. Hu, S.; Tian, S.; Zhao, J.; Shen, R. Path Planning of an Unmanned Surface Vessel Based on the Improved A-Star and Dynamic Window Method. *Journal of Marine Science and Engineering* **2023**, *11*.
18. Liu, Z.; Zhang, Y.; Yu, X.; Yuan, C. Unmanned surface vehicles: An overview of developments and challenges. *Annual Reviews in Control* **2016**, *41*, 71–93.
19. Peng, Z.; Wang, D.; Li, T.; Han, M. Output-feedback cooperative formation maneuvering of autonomous surface vehicles with connectivity preservation and collision avoidance. *IEEE Transactions on Cybernetics* **2019**, *50*, 2527–2535.
20. Peng, Z.; Wang, J.; Wang, D.; Han, Q.L. An overview of recent advances in coordinated control of multiple autonomous surface vehicles. *IEEE Transactions on Industrial Informatics* **2020**, *17*, 732–745.
21. Zhang, G.; Huang, C.; Li, J.; Zhang, X. Constrained coordinated path-following control for underactuated surface vessels with the disturbance rejection mechanism. *Ocean Engineering* **2020**, *196*, 106725.
22. Liu, L.; Wang, D.; Peng, Z. Coordinated path following of multiple underacutated marine surface vehicles along one curve. *ISA Transactions* **2016**, *64*, 258–268.
23. Liu, L.; Wang, D.; Peng, Z.; Li, T.; Chen, C.L.P. Cooperative Path Following Ring-Networked Under-Actuated Autonomous Surface Vehicles: Algorithms and Experimental Results. *IEEE Transactions on Cybernetics* **2020**, *50*, 1519–1529.
24. Fossen, T.I.; Strand, J.P. Passive nonlinear observer design for ships using Lyapunov methods: full-scale experiments with a supply vessel. *Automatica* **1999**, *35*, 3–16.
25. Loria, A.; Fossen, T.I.; Panteley, E. A separation principle for dynamic positioning of ships: Theoretical and experimental results. *IEEE Transactions on Control Systems Technology* **2000**, *8*, 332–343.
26. Tan, G.; Wang, Z. Generalized dissipativity state estimation of delayed static neural networks based on a proportional-integral estimator with exponential gain term. *IEEE Transactions on Circuits and Systems II: Express Briefs* **2020**, *68*, 356–360.
27. Cui, R.; Yang, C.; Li, Y.; Sharma, S. Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2017**, *47*, 1019–1029.
28. Majid, M.H.A.; Arshad, M.R. Hydrodynamic Effect on V-Shape Pattern Formation of Swarm Autonomous Surface Vehicles (ASVs). *Procedia Computer Science* **2015**, *76*, 186–191.
29. Ouyang, Z.; WANG, H.; HUANG, Y.; Yang, K.; Yi, H. Path planning technologies for USV formation based on improved RRT. *Chinese Journal of Ship Research* **2020**, *15*, 18–24.
30. Yang, J.M.; Tseng, C.M.; Tseng, P. Path planning on satellite images for unmanned surface vehicles. *International Journal of Naval Architecture and Ocean Engineering* **2015**, *7*, 87–99.
31. Jin, X.; Er, M.J. Cooperative path planning with priority target assignment and collision avoidance guidance for rescue unmanned surface vehicles in a complex ocean environment. *Advanced Engineering Informatics* **2022**, *52*, 101517.
32. Niu, H.; Savvaris, A.; Tsourdos, A.; Ji, Z. Voronoi-visibility roadmap-based path planning algorithm for unmanned surface vehicles. *The Journal of Navigation* **2019**, *72*, 850–874.

 doi:10.20944/preprints202312.1824.v1

33. Li, B.; Moridian, B.; Mahmoudian, N. Autonomous oil spill detection: mission planning for ASVs and AUVs with static recharging. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston. IEEE, 2018, pp. 1–5.
34. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research* **2010**, *29*, 485–501.