A Rapid Solution Method for Surface Unmanned Vessels Motion Planning Problem based on Optimal Control

Yousheng Hu[1,2a], Lei Sun[3b*], Qingnian Zhang[1], Yaping Ma[3], Kai Yi[3]
[1]School of Transportation and Logistics Engineering, Wuhan university of technology, Wuhan, China
[2]Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai), Zhuhai, China
[3]Beijing Highlander Digital Technology Co., Ltd., Beijing, China
[a]swagdab@qq.com, [b*] Corresponding author: sunl@highlander.com.cn

ABSTRACT

Unmanned Surface Vehicle (USV) technology has been widely applied in various fields, due to the nonlinear, multi-constraint, high-dimensional nature of motion planning problems, as well as the dynamic characteristics of USV movement and the uncertainty of the environment, solving USV motion planning problems is highly complex and has an unpredictable failure rate. This paper focuses on the motion planning problem of surface USVs and studies the solution methods based on the optimal control model in modern control theory. By introducing the trapezoidal collocation method from the direct method, the optimal control formula describing the entire USV motion planning problem is fully discretized into a large-scale non-linear programming problem (NLP), which is then quickly solved using an interior-point solver. In random scenario simulation experiments with different ship interaction conditions, the motion planning algorithm's solution time is within 250ms, basically meeting the practical engineering requirements. The highlights of this paper are as follows: 1. The optimal control method is used to ensure that the output control input is suitable for the USV power chassis; 2. The direct method is used to fully discretize the problem, transforming the optimal control problem, which essentially seeks the functional extremum in infinite space, into a NLP that can be quickly solved using mature mathematical tools to meet the low-latency requirements of USV working conditions; 3. The adoption of trapezoidal collocation discretization ensures that the constraints describing the control system in NLP are all first-order algebraic terms and their linear combinations, guaranteeing the low difficulty of solving NLP and effectively reducing the failure rate to within 10%.

Keywords: USV; Optimal control; Motion planning; Trapezoidal collocation.

1. INTRODUCTION

Unmanned boat motion planning refers to the process of formulating appropriate motion trajectories for unmanned boats in the automatic driving module[1], based on current environmental information and predetermined tasks. The trajectory serves as the ideal motion state for the unmanned boat, and its value is used as the reference for the boat's chassis control[2]. Firstly, the unmanned boat will calculate multiple possible paths based on current environmental information[3], such as underwater maps, underwater obstacles, and heading information. Then, the boat will evaluate the suitability of these paths based on its tasks, such as conserving energy[4], avoiding obstacles[5], or increasing speed[6], and choose the most optimal trajectory for navigation[7]. At present, research on unmanned boat motion planning based on reinforcement learning, graph search technology, and traditional control theory has made significant progress, but there are still corresponding difficulties that need to be overcome [8].

(1) Unmanned boat motion planning based on reinforcement learning has been a research hotspot in recent years[9]. This approach enables unmanned boats to learn the optimal motion strategy through autonomous interaction with the environment, continuously trying and learning from experience to achieve efficient and robust motion planning [10]. Riccardo[11]and others proposed a reinforcement learning method based on a physical model, which gradually improves control efficiency during the learning process to achieve tasks such as trajectory tracking. Additionally, Wu[12] and others also proposed a reinforcement learning method based on a simulation model for obstacle avoidance and tracking tasks in unmanned boats. However, this method still has some problems, such as low motion planning efficiency and high data requirements[13]. Moreover, the entire planning process must be completed offline[14], which cannot meet the real-time operation requirements of unmanned boats in complex environments[15] .

International Conference on Smart Transportation and City Engineering (STCE 2023),
edited by Miroslava Mikusova, Proc. of SPIE Vol. 13018, 1301819
 · 0277-786X · doi: 10.1117/12.3024824

(2) Unmanned boat motion planning based on graph search technology has high planning efficiency and scalability [16]. This method usually employs search algorithms like the A* algorithm to find the shortest path in a map as the unmanned boat's motion route. As shown in Figure 1, the Probabilistic Roadmap (PRM) algorithm[17]uses feasible vertices obtained through random sampling to construct a collision-free graph in a static environment, meeting the requirements of multiple shortest-path queries. The rapidly-exploring random tree (RRT) algorithm[18] has outstanding advantages such as probabilistic completeness and good real-time performance, but it does not consider dynamic constraints, resulting in non-smooth trajectories and insufficient obstacle avoidance safety due to the lack of a speed dimension[19] .

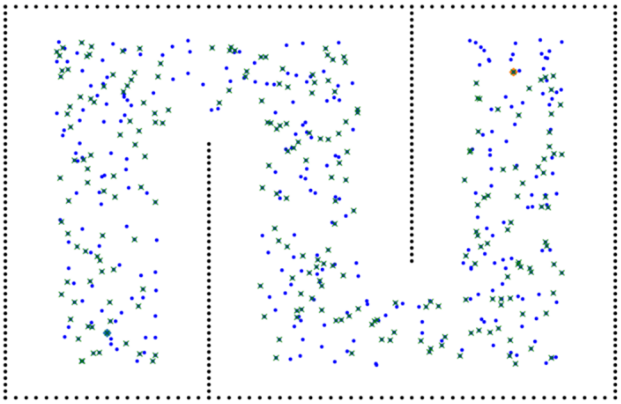


Figure 1: PRM algorithm

(3) Methods based on traditional control theory are the most widely used in unmanned boat decision-making [20]. These methods typically utilize physical models and control theories, including feedback control and model predictive control (MPC), among others[21]. These methods do not require large amounts of data for training and are well-suited for applications requiring efficient and real-time decision-making[22]. For example, Chen[23]applied MPC controllers for attitude control of unmanned boats and used feedback control for speed control, successfully applying them to boat models. However, traditional control methods struggle to handle nonlinear systems. The dynamic equations of nonlinear systems are often high-order, nonlinear, and coupled, making modeling and control design more difficult for complex systems like unmanned boats.

In contrast, unmanned boat motion planning methods based on modern control theory are more capable of handling nonlinear and complex systems. Nonlinear control methods can better address motion planning problems for nonlinear systems like unmanned boats. Naturally, using optimal control methods that do not require accurate system modeling but only require some understanding of the system's motion equations and the ability to write optimal control methods that meet different system constraints has great potential in today's mature computer technology. However, solving optimal control problems requires numerical computation, which has high computational complexity and requires significant computational resources and time. During motion planning, these methods are sensitive to noise and uncertainty, leading to a need for constant updates to the actual controller. This requires the entire solution process to be performed repeatedly with low latency, which is almost a step-by-step "calculation" process in practical engineering. In this study, we focus on the motion planning problem of unmanned boats. According to the unmanned boat system, we establish the optimal control model based on safety, obstacle avoidance, and smoothness and comfort as indicators. The trapezoidal discrete point configuration in the direct method is used to discretize the optimal control problem completely, transforming the entire unmanned boat motion planning problem into a large-scale but non-high-order-constraint nonlinear programming problem (NLP). This makes it convenient to call excellent open-source tools for solving simple NLP problems, such as the Interior Point Optimizer (IPOPT). The output results can be directly used as input values for the unmanned boat control system. Finally, according to the three types of ship interaction scenarios defined by the International Regulations for Preventing Collisions at Sea (COLREGS): head-on situations, overtaking situations, and crossing situations, we design random unmanned boat operation simulation scenarios to verify the algorithm's performance.

2. MATHMATIC MODEL ESTABLISHING

2.1 USV Motion Model

The method for describing the motion state of an unmanned vessel can be derived from the early positioning systems of unmanned aerial vehicles (UAVs). For devices in the world space or Cartesian coordinate system, in addition to using the x, y, and z values to represent their position in the three-dimensional space, the yaw, pitch, and roll angles are also needed to represent the UAV's own posture. Thus, the combination of x, y, z, yaw, pitch, and roll constitutes a set of units in the state space that can represent all possible poses. Taking the speedboat in the open-source Virtual RobotX (VRX) boat simulator on Github as an example, as shown in Figure 2, the x, y, and z coordinates of the center point of the boat's horizontal axis are taken as the world coordinate values, which is where the red, blue, and green axes intersect in the figure. The rotation angle around the z-axis (blue axis) is taken as the yaw angle, representing the direction in the horizontal space. The rotation angle around the y-axis (green axis) is taken as the pitch angle, representing the amplitude of the bow and stern rising or sinking. The rotation angle around the x-axis (red axis) is taken as the roll angle to describe the state of lateral rolling of the body.

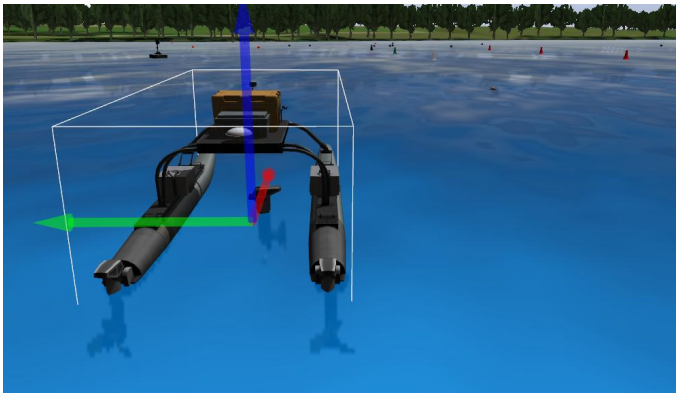


Figure 2: Speedboat Model in VRX

After determining the definition of the ship's motion state, it is necessary to discuss the motion characteristics that conform to the ship's dynamics. In order to simplify the algorithm design, this article will ignore the pitch and roll of the ship, and the consideration of the pose will be purely focused on the yaw angle.

Therefore, in the design of the entire motion planning module in this article, a simplified version of the industry-standard six-degree-of-freedom model for unmanned surface vehicles is used to characterize a physical motion model that is more in line with the ship's dynamics.

$$\begin{cases} \dot{\boldsymbol{\eta}} = \mathbf{J}(\psi)\mathbf{v} \\ \mathbf{M}\dot{\mathbf{v}} = -\mathbf{C}(v) - \mathbf{D}(v)\mathbf{v} + \mathbf{B}\boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}} \end{cases} \tag{1}$$

$$\begin{cases} \boldsymbol{\eta} = \begin{bmatrix} x & y & \psi \end{bmatrix}^{\mathrm{T}} \\ \mathbf{v} = \begin{bmatrix} u & v & r \end{bmatrix}^{\mathrm{T}} \\ \boldsymbol{\tau} = \begin{bmatrix} \tau_{\mathrm{u}} & 0 & \tau_{\mathrm{r}} \end{bmatrix}^{\mathrm{T}} \end{cases} \tag{2}$$

$$\mathbf{J}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3}$$

$$\mathbf{M}=\begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{11} & m_{33} \end{bmatrix} \tag{4}$$

$$\mathbf{C}(v)=\begin{bmatrix} 0 & 0 & -m_{11}v \\ 0 & 0 & m_{11}u \\ m_{22}v & -m_{11}u & 0 \end{bmatrix} \tag{5}$$

$$\mathbf{D}(v)=\begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & d_{23} \\ 0 & d_{32} & d_{33} \end{bmatrix} \tag{6}$$

$$\mathbf{B}=\begin{bmatrix} b_{11} & 0 \\ 0 & b_{22} \\ 0 & b_{32} \end{bmatrix} \tag{7}$$

In the above equation, x, y and ψ represent the position and orientation of the unmanned boat in the inertial coordinate system; u, v, and r represent the forward speed, lateral drift speed, and heading angular velocity in the boat coordinate system; τ_u and τ_u represent the forward force and yaw moment; τ_{wind} and τ_{wave} represent the forces and moments caused by wind and waves. The remaining variables characterize the distribution of impulse and torque in the Cartesian three-axis system acting on the rigid body motion of the boat, causing motion or rotation in various directions. In this paper, the parameters of the OceanAlpha M40 speedboat in VRX are configured.

The above kinematic model can represent a kind of motion characteristic exhibited by a rigid body with a certain length, such as the unmanned boat OceanAlpha M40, when it moves under its own dynamics. This characteristic can be used to describe some motion situations of the rigid body. In this paper, to simplify the complex fluid dynamics to the greatest extent, this model is used to characterize the motion of the unmanned boat. According to control theory, the kinematic model of the unmanned boat is used as the standard system equation shown in equation (8), and the system state equation characterizing the relationship between the control input of the unmanned boat and its position and posture can be obtained.

$$\frac{dx(t)}{dt}=f(x(t),u(t),t) \tag{8}$$

2.2 Optimal Control Modeling

Firstly, let's return to the motion planning problem itself. This problem refers to finding a path or trajectory between the initial configuration of a ship at the starting time and the final configuration of the ship at the ending time, which meets the constraint conditions. A trajectory, as opposed to a path, includes an additional dimension of information, which is the timestamp, making the speed controllable. Since the obstacle constraints and dynamic conditions involved in general ship navigation problems are time-dependent, setting the output of the ship motion planning as a trajectory form is most appropriate. In order to describe the general problem of ship motion planning uniformly, accurately, and directly, this paper adopts the form of optimal control problem for modeling.

The goal is to solve the optimal control problem in the *Bolza* form, which is to obtain the control variable $u(t)$ and determine the termination time t_f, so that the performance index functional can be achieved.

$$J=\Phi(x(t_0),t_0,x(t_f),t_f)+\int_{t_0}^{t_f}L(x(t),u(t),t)dt \tag{9}$$

Along the corresponding state trajectory, the minimum value is obtained. The constraints for the entire system include:

System dynamic equation constraints

$$\frac{dx(t)}{dt} = f(x(t), u(t), t) \tag{10}$$

Endpoint constraints

$$\varphi(t_0, t_f, x(t_0), x(t_f), u(t_0), u(t_f)) = 0 \tag{11}$$

path constraints

$$C(x(t), u(t), t) \le 0, t \in [t_0, t_f] \tag{12}$$

Comparing equation (1) with the general system dynamic equation constraint (3), it is clear that varibles in equal (1) belong to the state variables describing the current ship system, they are the inputs for the entire system, which are the control variables. The entire control system can be intuitively understood as: if the vehicle's motion state at the initial moment is given, and then the motion domain input is provided, the motion state in that domain can be uniquely determined through integration or finite accumulation, which corresponds to the unique travel trajectory of the ship that the algorithm process aims to solve.

In this paper, the performance index function of the motion planning problem is chosen as the cumulative sum of the distance derivative $jerk$ to the obstacle and the second-order derivative of the dynamics throughout the motion process. This index can be intuitively understood as maximizing the distance from obstacles while ensuring a smooth trajectory and passenger comfort. At this point, the complete unmanned boat motion planning task that meets the design requirements can be described as the following optimal control problem:

$$\begin{aligned} &\textit{Opt}: \quad \min J = \int_{t_0}^{t_f} \left(j(t)^2 + distance(x(t), obstacle) \right) dt \\ &\textit{subject to}: \\ &\qquad \textit{USV System Constrain} \ (10) \\ &\qquad \textit{Initial \& end Constrain} \ (11) \\ &\qquad \textit{Varible Path Constrain} \ (12) \end{aligned} \tag{13}$$

3. FULLY DISCRETE SOLUTION

3.1 Optimal Control Discretization

In the optimal control problem described in the previous section, which covers all the requirements for the unmanned vessel motion planning process, variables $u(t_i)$ and $x(t_i)$ are discretized into numerical sequences based on time for all and. The optimal control input is obtained by solving the resulting Nonlinear Programming (NLP) problem after discretization, which is referred to as the direct method. The sequence can be considered as the control input $u(t)$ to be solved after fitting. As shown in Figure 3, the direct method discretizes NLP variables into discrete configuration points on $t(i)$. The $u(t)$ and $x(t)$ are discretized with equal importance, resulting in a large-scale NLP problem. However, it is very convenient for handling complex constraints, often requiring only simple linear operations. It is particularly suitable for solving large-scale optimization problems with constraints like the dynamics of unmanned vessels. This method is essentially equivalent to the implicit Runge-Kutta method, offering good solution stability and effectively solving the optimal control problem formulated based on the dynamics of unmanned vessels.

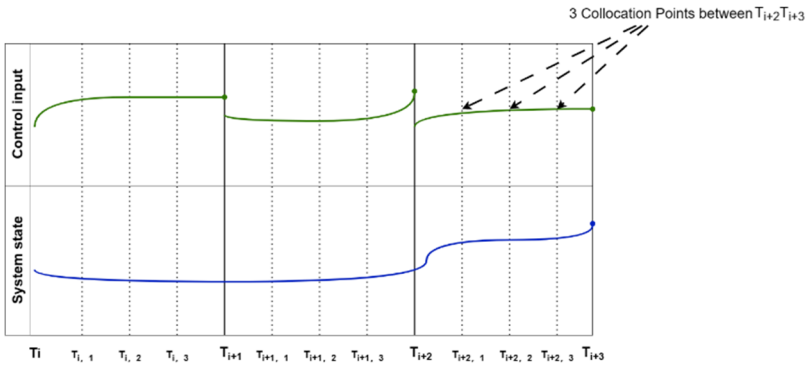


Figure 3: Schematic of Discrete Point Configuration for the System

When rewriting equation (13) using the fully discretized direct method, the optimization time domain $[t_0, t_f]$ is first divided into N equal-length finite elements:

$$\begin{aligned} &\bar{t}_{t+1}\text{-}\bar{t}_t = \frac{t_f - t_0}{N}, i \in [1, N], \\ &\bar{t}_0 = t_0, \\ &\bar{t}_N = t_f \end{aligned} \tag{14}$$

To describe each variable that describes the entire unmanned vessel motion process, an interpolation function is used to describe any discretized time interval on the finite element $[t_i, t_{i+1}]$:

$$\begin{cases} u(t) = \sum_{j=0}^{K} \bar{l}_j(\tau) u_{i,j} \\ x(t) = \sum_{j=1}^{K} \bar{l}_j(\tau) x_{i,j} \end{cases} \tag{15}$$

In equation (15), K represents the order of the interpolation function, $u_{i,j}$ are the discrete configuration points for the differential variables of the control input, and $x_{i,j}$ are the discrete configuration points for the system's differential variables, and $\bar{l}$ represents the basis function corresponding to the differential variable $u(t)$ or $x(t)$. Basis functions represent methods of fitting discrete points into a continuous curve, commonly used such as Lagrange interpolation functions. If Lagrange interpolation functions are used, the point pairing method is orthogonal point pairing, which is similar to trapezoidal point pairing, but generally uses higher-order polynomials. Orthogonal points are located at the roots of orthogonal polynomials, usually Chebyshev or Legendre polynomials[24]. Improving the accuracy of the solution is usually achieved by increasing the number of trajectory segments or the order of the polynomial for each segment. An important reason for using high-order orthogonal polynomials for function approximation is that they can achieve spectral convergence. This means that if the basic functions are smooth enough, the convergence rate is exponential. In the case of approximating the entire trajectory with a single high-order polynomial, the resulting method is called pseudospectral point pairing method or global point pairing method. However, this leads to a large number of high-order constraints in the resulting NLP, which slows down the solution rate and increases the probability of failure.

To improve this problem while maintaining the convergent and accurate properties of the discrete system description function, this paper chooses to use the trapezoidal collocation method to configure the control input and system state at the same discrete time interval point. The trapezoidal point pairing method works by approximating the control trajectory and the system dynamics as piecewise linear functions, also known as linear splines, as shown in Figure 4. When constructing splines, each node is used to represent a configuration point that connects two polynomial segments. For the trapezoidal point pairing method, the spline nodes coincide with the configuration points [25].

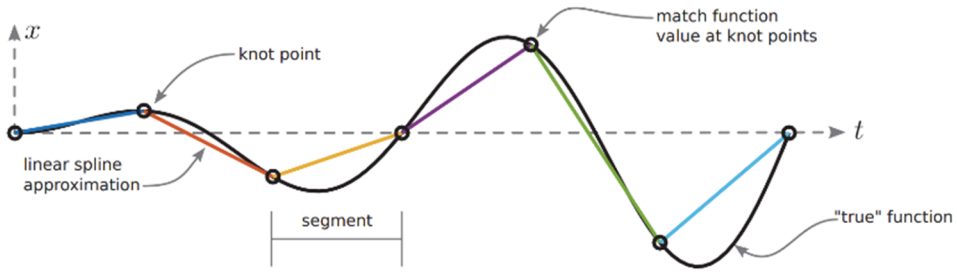


Figure 4 A schematic of the spline discretization.

For example, the trapezoidal collocation constraints are constructed by writing the dynamics in integral form and then approximating that integral using trapezoidal quadrature, as shown in the following formula.

$$\dot{x} = f \tag{16}$$

$$\int_{t_k}^{t_{k+1}} \dot{\mathbf{x}} dt = \int_{t_k}^{t_{k+1}} \mathbf{f} dt \tag{17}$$

$$\mathbf{x}_{k+1} - \mathbf{x}_k \approx \frac{1}{2} h_k \cdot \left(\mathbf{f}_{k+1} + \mathbf{f}_k \right) \tag{18}$$

Thus, the optimal control problem formulated in the previous section is discretized into the form of equation (19)

$$\begin{aligned} &\min J(t_f) \\ &s.t. \sum_{k=0}^{K} \left(\frac{d(\bar{l}_k)}{dt}\bigg|_{t=t_k} \cdot u_{i,k} \right) = (\bar{t}_i - \bar{t}_{i-1}) \cdot F(u_{i,j}, x_{i,j}), \\ &G(u_{i,j}, x_{i,j}) \le 0, \\ &\bar{t}_i - \bar{t}_{i-1} = \frac{t_f - t_0}{N}, \bar{t}_0 = t_0, \bar{t}_N = t_f, \\ &i = 1, ..., N, j = 0, ..., K. \end{aligned} \tag{19}$$

3.2 Optimization problem solving

In the previous section, the entire optimal control problem was discretized into the form of Equation (19). It is observed that this equation is an NLP. There are many mature tools in the industry for solving NLP problems with rapid convergence to the optimal solution. In this paper, IPOPT, an open source project maintained by the University of Oxford for many years, is chosen as the solving tool.

IPOPT uses an interior-point algorithm to solve NLP problems, and the standard format of an NLP problem is as follows:

$$\begin{aligned} &min \qquad f(x) \\ &subject\ to\ gl \le g(x) \le gh \\ &\qquad\qquad xl \le x \le xh \end{aligned} \tag{20}$$

During the optimization process, IPOPT solver internally calls various linear solvers, and different linear solvers exhibit different efficiency and robustness for the same problem. When working with IPOPT, the NLP information is described using C++ or Python language. In the process of solving the problem, the following intermediate quantities are used as criteria for convergence, which are iteratively calculated from the initial values: objective function value, gradient of the objective function, constraint values, Jacobian matrix (i.e. the first derivative of constraints), and Hessian matrix (i.e. the second derivative of Lagrangian function). The NLP solving process and definitions of intermediate quantities are explained below.

Firstly, the number of variables and constraints need to be determined based on the problem dimension. The problem bounds, including the variable bounds, constraint bounds, and initial values (set as a zero vector in this paper), need to be restricted. The structure of the constraints also needs to be described, such as the non-zero values and sparse structure of the Jacobian matrix (constraints) and Hessian matrix (Lagrangian equation). The Jacobian matrix is the differential coordinate form of a real-valued function with respect to the coordinate system, which can be seen as the first-order coefficient of the multivariate Taylor expansion. When there are m constraints and n variables, the constraint equations can be regarded as an m-dimensional mapping from the variable space to the constraint space of n dimensions, and the matrix composed of the first-order partial derivatives of the constraint equation in order is the Jacobian matrix. The Hessian matrix of the Lagrangian equation can be used to solve the optimization problem with constraints. When the derivative of the objective function and the derivative of the constraint differ only by a multiplication factor at a point, it means that the normal of the constraint and the objective function itself are tangent at that point. According to the Hessian matrix of the Lagrangian equation, it can be determined whether the second-order derivative is non-zero at the point where the first-order derivative is zero and whether it is a maximum or a minimum value. IPOPT also has quasi-Newton method, which designs an approximate matrix to replace the complex Hessian matrix. Therefore, when the IPOPT solver option uses quasi-Newton method, it is not necessary to calculate the value of the Hessian matrix.

For the fully discretized obstacle avoidance decision and trajectory planning problem of unmanned ships, the result of the entire process is two high-dimensional discrete vector groups, which respectively describe the input vector group of the unmanned ship's kinematics and the system vector group. The former includes the first derivative of the rudder angle, the first derivative of the main engine speed, etc., while the latter includes the roll angle of the hull, Cartesian coordinate values, etc.

4. SIMULATION AND ANALYSIS

4.1 Simulation Scenario Setup

In this chapter, all simulation cases are run on Windows10. The specific content of the simulation is: for different working conditions, the entire program is run by discretizing the optimal control problem and generating NLP, and obtaining the analytical solution of the solver. The time consumption feedback and diagnostic information of the calculation performance provided by IPOPT will be an important reference for algorithm evaluation. The hardware environment for simulation is: CPU model Intel (R) Core (TM) i7-10870H CPU @ 2.20GHz 2.21 GHz, RAM 32GB, the compiler is Visual Studio Code, and IPOPT is called by Casadi through Python language. Visualization work is completed by calling the Matplotlib library.

According to international regulations for preventing collisions at sea(COLREGS), the encounter situations of all ships are divided into three types: head-on, overtaking, and crossing situations. The rules also give ships the attributes of "give-way ship" or "stand-on ship". Under different encounter situations, the applicable rules for each attribute ship are shown in Table 1. Among them, Rule 8 of the maritime rules stipulates that in the interaction process, the four principles of "early, large, single, and clear" must be followed. Among them, "early" means taking action as early as possible for the logic that meets the rules; "large" means taking large measures in the corresponding logic of the rules, whether it is avoidance or crossing actions, so that the other party can easily perceive it and the action taken should be large enough to ensure safety distance; "single" means that only turning single way alone is most effective action to avoid a pressing situation when there is enough space and the avoidance process should avoid making a series of small changes in speed or direction; "clear" means that the effectiveness of the avoidance should be observed in real time until the ship is finally cleared. When designing the entire unmanned boat motion planning simulation, it should also comply with the above principles.

Table 1: The Rules to be Followed by Vessels with Different Attributes under Different Encounter Situations According to COLREGS

Encounter Situation	Attributes of Vessels	Rule
Head-on situation	Both vessels are "vessels meeting head-on"	Each vessel shall alter its course to starboard so that each shall pass on the port side of the other.
Crossing situation	One vessel is "the give-way vessel" and the other is "the stand-on vessel"	The give-way vessel shall keep out of the way of the stand-on vessel.
Overtaking situation	The overtaking vessel and the vessel being overtaken	The overtaking vessel shall keep out of the way of the vessel being overtaken.

Therefore, the output of motion planning algorithms in different simulation scenarios should also comply with the above action rules, and the planning results should logically coincide with the terms indicated. The simulation scenarios meet the specific requirements of unmanned boats for the three types of encounters: head-on, overtaking, and crossing. The following are the designs for randomly generated scenarios under the three types of encounters, each with an equal probability of 33.3%:

(1) Head-on encounter scenario. Scene description: In a body of water with a width of 10m and a variance of 5m2 in the motion space, an OceanAlpha M40 unmanned boat is head-on with another vessel. The sailing directions of the two vessels are opposite or the speed angle is between 175 degrees and 180 degrees.

General strategy: In this case, the unmanned boat adopts the principle of keeping to the right in the corresponding condition of the scene.

(2) Overtaking scenario. Scene description: In a body of water with a width of 20m and a variance of 10m2 in the motion space, the OceanAlpha M40 unmanned boat is trying to catch up and overtake a slower vessel from behind. The sailing directions of the two vessels are the same or there is an angle less than 45 degrees, but their speeds should be different, with a difference not exceeding twice their own speed. Careful handling is required to avoid collisions.

General strategy: In this case, the unmanned boat should follow the principle of "the overtaking vessel shall keep out of the way with sufficient sea room and maintain a clear view of the situation." The unmanned boat should first keep a safe distance from the vessel ahead, and then accelerate and overtake from the left or right of the slow boat when it is judged safe to do so. Throughout the process, the unmanned boat should ensure a sufficient safe distance from the vessel ahead.

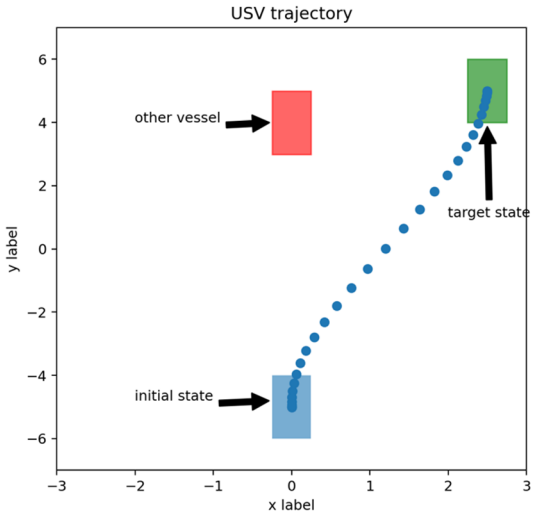
(3) Crossing scenario. Scene description: In a body of water with a width of 30m and a variance of 15m2 in the motion space, the OceanAlpha M40 unmanned boat intersects with another vessel at a perpendicular or nearly perpendicular angle. The sailing directions of the two vessels are different, and measures need to be taken to avoid collisions at the crossing point.

General strategy: In this case, the unmanned boat should follow the principle of "the vessel on the starboard side shall keep out of the way." When the other vessel is on the right side of the unmanned boat, the unmanned boat should slow down or change its course to let the other vessel pass first. Then, when the other vessel has passed safely, the unmanned boat can resume its original speed and continue sailing. Throughout the process, the unmanned boat needs to pay close attention to the surrounding environment to ensure safety.

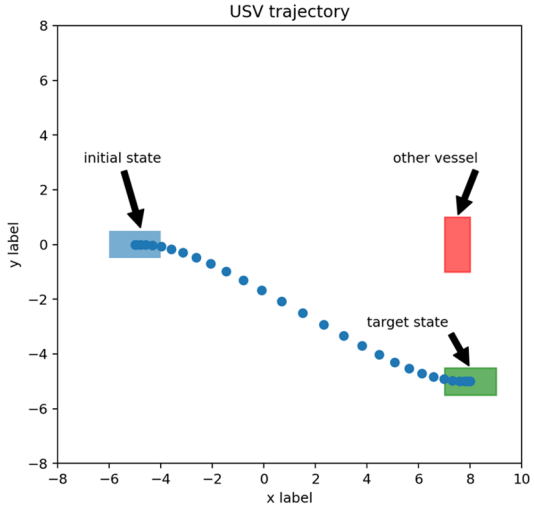
4.2 Result Analysis

The following multiple figures show the results of obstacle avoidance decision-making and trajectory planning by the OceanAlpha M40 based on the initial information of the unmanned vessel and the other ships in several randomly generated scenarios. The output includes the path points and the final pose data. The blue box represents the enveloping line that surrounds the navigation area of the unmanned vessel, representing the initial pose of the ship. The position and orientation of each enveloping line represent the position and pose of the unmanned vessel. The green envelope represents the target position and corresponding pose, and the red envelope represents the position and pose of the other ship. Based on the perceived information of the other ship's position and velocity and the current positioning information of the unmanned vessel, the current scenario type is determined. The corresponding real-time IPOPT message record is also provided, which records the number of iterations of the entire optimization problem, the iteration time of each intermediate quantity, the various indicators of the nonlinear constraint problem, and the critical solution time of the entire algorithm.

In all the different scenarios generated according to the maritime regulations and different initial and terminal positions and poses, the algorithm can converge quickly and complete the solution, and the generated trajectories all meet the requirements for solving the unmanned vessel obstacle avoidance decision-making and trajectory planning problems. The entire solution time is also very fast. Using the interior-point method solver, the number of iterations for solving the large-scale nonlinear programming problem is generally within 50 iterations, indicating that the generated NLP is easy to solve.



(a) Motion planning results for head-on encounter scenarios



(b) Motion planning results for right crossing scenarios

```
Number of Iterations....: 40

                                   (scaled)                 (unscaled)
Objective...............:  1.8644843384781288e+002   1.8644843384781288e+002
Dual infeasibility......:  7.4420549446895020e-009   7.4420549446895020e-009
Constraint violation....:  7.8553830107352951e-012   7.8553830107352951e-012
Complementarity.........:  2.6892436435412311e-009   2.6892436435412311e-009
Overall NLP error.......:  7.4420549446895020e-009   7.4420549446895020e-009


Number of objective function evaluations             = 51
Number of objective gradient evaluations             = 35
Number of equality constraint evaluations            = 51
Number of inequality constraint evaluations          = 51
Number of equality constraint Jacobian evaluations   = 42
Number of inequality constraint Jacobian evaluations = 42
Number of Lagrangian Hessian evaluations             = 40
Total CPU secs in IPOPT (w/o function evaluations)   =      0.110
Total CPU secs in NLP function evaluations           =      0.047

EXIT: Optimal Solution Found.
      solver  :   t_proc      (avg)   t_wall      (avg)    n_eval
       nlp_f  |   2.00ms ( 39.22us)   2.00ms ( 39.31us)        51
       nlp_g  |   5.00ms ( 98.04us)   5.01ms ( 98.16us)        51
  nlp_grad_f  |   1.00ms ( 27.78us)   1.00ms ( 27.83us)        36
  nlp_hess_l  |  25.00ms (641.03us)  25.02ms (641.54us)        39
   nlp_jac_g  |  15.00ms (348.84us)  14.93ms (347.12us)        43
       total  | 161.00ms (161.00ms) 161.00ms (161.00ms)         1
```

(c) NLP solver information for head-on encounter scenarios

```
Number of Iterations....: 38

                                   (scaled)                 (unscaled)
Objective...............:  1.8644291202830263e+002   1.8644291202830263e+002
Dual infeasibility......:  1.7234652908586081e-009   1.7234652908586081e-009
Constraint violation....:  2.5535129566378600e-014   2.5535129566378600e-014
Complementarity.........:  2.5121705185153769e-009   2.5121705185153769e-009
Overall NLP error.......:  2.5121705185153769e-009   2.5121705185153769e-009


Number of objective function evaluations             = 61
Number of objective gradient evaluations             = 34
Number of equality constraint evaluations            = 61
Number of inequality constraint evaluations          = 61
Number of equality constraint Jacobian evaluations   = 40
Number of inequality constraint Jacobian evaluations = 40
Number of Lagrangian Hessian evaluations             = 38
Total CPU secs in IPOPT (w/o function evaluations)   =      0.113
Total CPU secs in NLP function evaluations           =      0.042

EXIT: Optimal Solution Found.
      solver  :   t_proc      (avg)   t_wall      (avg)    n_eval
       nlp_f  |        0 (       0)        0 (       0)        61
       nlp_g  |   7.00ms (114.75us)   5.99ms ( 98.25us)        61
  nlp_grad_f  |        0 (       0)        0 (       0)        35
  nlp_hess_l  |  26.00ms (702.70us)  25.79ms (696.97us)        37
   nlp_jac_g  |   9.00ms (219.51us)   9.01ms (219.66us)        41
       total  | 159.00ms (159.00ms) 158.78ms (158.78ms)         1
```

(d) NLP solver information for right crossing scenarios

Figure 5: Partial results display

As shown in Figures 5a and 5c, the algorithm performs well in a randomly generated head-on situation, and the real-time IPOPT solving information shows that the entire solution time is only 159.00ms, which is relatively short. The entire output sequence generated by the algorithm controls the unmanned boat to accelerate and move to the right to keep a safe distance from the approaching ship. As shown in Figures 5b and 5d, in a randomly generated right-crossing situation, the algorithm takes 161.00ms and completes NLP convergence after 40 iterations. The decision sequence generated by the algorithm makes the unmanned boat move to the right to avoid the target ship, and the entire trajectory is very smooth, and the initial and final orientations of the unmanned boat do not change, which does not affect the original course.

This section conducted a large number of simulation experiments on the algorithm under various working conditions. The results showed that the algorithm demonstrated good obstacle avoidance and trajectory planning capabilities in various situations and also performed well in terms of compliance with maritime regulations and ensuring comfort. It is worth mentioning that the entire NLP convergence time of all working conditions was completed in about 200ms, which can meet the requirements of unmanned boat automatic driving systems under actual working conditions.

After running 100 test samples for each working condition that comply with the scene design, the running time was statistically analyzed, as shown in Figure 6. From top to bottom, the three working condition running time data are: head-on situation, overtaking situation, and crossing situation. The translucent red line represents the average running time in milliseconds.

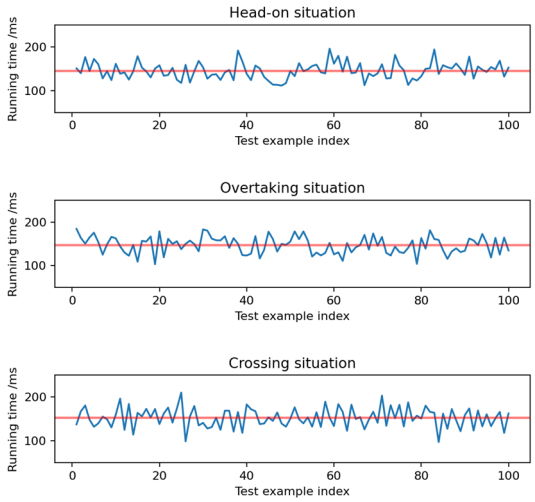


Figure 6. the computation time for each scenario

After obtaining the solving time statistics for the three types of situations, algorithm performance was evaluated based on efficiency:

(1) Average solving time: The average solving time for the three types of situations were calculated as follows: head-on situation: 145 ms, overtaking situation: 149 ms, crossing situation: 157 ms. Based on the average solving time, the algorithm performs best in head-on situations, while its performance in crossing situations is relatively poorer. The algorithm's performance differs to some extent when dealing with different situations.

(2) Stability (consistency): The standard deviation of the solving time for the three types of situations were calculated as follows: head-on situation: 50 ms, overtaking situation: 55 ms, crossing situation: 69 ms. Based on the standard deviation, the algorithm's stability is best in head-on situations, while its stability is relatively poorer in crossing situations. This indicates that the algorithm's consistency is better in head-on situations when conducting multiple experiments of the same situation.

(3) Extreme situations: The maximum and minimum values were extracted from the data as follows: head-on situation: minimum 98 ms, maximum 202 ms, overtaking situation: minimum 99 ms, maximum 190 ms, crossing situation: minimum 80 ms, maximum 210 ms. In extreme situations, the solving time for crossing situations is the shortest, but also the longest among the three types of situations. This index helps indicate that there are more potential performance bottlenecks or abnormal situations in crossing situations.

Figure 7 shows the solving time required for 100 groups of unmanned boat motion planning problems based on optimal control using three different methods: trapezoidal discretization, orthogonal collocation, and without using full-discretization. In the figure, the x-axis represents the index of the data point, and the y-axis represents the solving time. Different data sets are represented by different colors, and the black part represents NLP solving failures in that case.

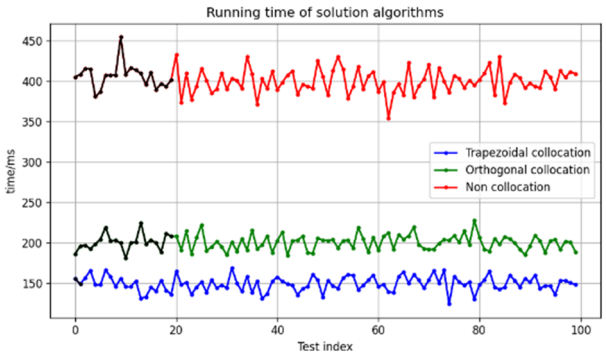


Figure 7 required time for solving the optimal control using different methods

According to Figure 7, using the trapezoidal point configuration method to solve the optimal control problem of unmanned boat motion planning requires the shortest average time and has a smaller fluctuation range. In contrast, the orthogonal configuration method and the method without discretization have significant differences in solving time, indicating that the acceleration effect brought by converting the unmanned boat motion planning problem into an NLP with full discretization is significant. The time required for the orthogonal configuration method is slightly longer, while the time required without discretization is the longest, and the fluctuation range is the largest. Additionally, in the non-trapezoidal point method experimental group, within 10% of the NLP solving failed, which suggests that converting NLP constraints into linear combinations of first-order terms can effectively reduce the difficulty of solving and improve the success rate of solving. The algorithm works as expected.

5. CONCLUSION

This paper proposes a real-time and efficient algorithm framework for unmanned surface vehicle (USV) motion planning problems, which meets the requirements of the USV dynamic system. By using optimal control to model the system and adopting trapezoidal discretization to transform the optimal control problem into a nonlinear programming problem with only simple constraints, the real-time performance and stability of the solution are effectively improved. With the help of an open-source mathematical solver, the USV motion planning problem is efficiently solved. The algorithm is tested in various random simulation scenarios with overtaking, head-on, and crossing navigation features. The overall time

consumption is controlled within 250ms, and the NLP solving failure rate is within 10%, which is apparently lower than that of existing algorithms. Experimental data demonstrates that the proposed algorithm can effectively plan dynamic and smooth trajectories for USV, which satisfy the safety obstacle avoidance and passenger comfort requirements and meets the design needs.

ACKNOWLEDGMENT

Project supported by Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai) (SML2021SP101).

REFERENCES

[1] Unmanned boat design for Challenges and verification of unmanned surface ship intelligent navigation (2018). In 2018 IEEE 8th International Conference on Underwater System Technology: Theory and Applications (USYS) (pp. 01-03). IEEE.
[2] Bai, X., Li, B., Xu, X., & Xiao, Y. (2022). A Review of Current Research and Advances in Unmanned Surface Vehicles. Journal of Marine Science and Application.
[3] Liu, Z., Zhang, Y., Yu, X., & Yuan, C. (2016). Unmanned surface vehicles: An overview of developments and challenges. Annual Reviews in Control 41: 71-93.
[4] Setiawan Joga Dharma, Septiawan Muhammad Aldi, Ariyanto Mochammad, Caesarendra Wahyu, Munadi M., Alimi Sabri, Sulowicz Maciej. Development and Performance Measurement of an Affordable Unmanned Surface Vehicle (USV)[J]. Automation,2022,3(1).
[5] Wang Zhenyu, Liang Yan, Gong Changwei, Zhou Yichang, Zeng Cen, Zhu Songli. Improved Dynamic Window Approach for Unmanned Surface Vehicles' Local Path Planning Considering the Impact of Environmental Factors[J]. Sensors,2022,22(14).
[6] Karnani Chandanlal, Raza Syed Akhter,Asif Muhammad, Ilyas Muhammad. Adaptive Control Algorithm for Trajectory Tracking of Underactuated Unmanned Surface Vehicle (UUSV)[J]. Journal of Robotics,2023,2023.
[7] Zheng Jian, Hu Jiayin, Li Yun. Codesign of dynamic collision avoidance and trajectory tracking for autonomous surface vessels with nonlinear model predictive control[J]. Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment,2022,236(4).
[8] Yining Ma, Chen Sun, Junyi Chen, Dongpu Cao, Lu Xiong, "Verification and Validation Methods for Decision-Making and Planning of Automated Vehicles: A Review", IEEE Transactions on Intelligent Vehicles, vol.7, no.3, pp.480-498, 2022.
[9] B. R. Kiran et al., "Deep Reinforcement Learning for Autonomous Driving: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 6, pp. 4909-4926.
[10] Yanes Luis Samuel, Gutiérrez Reina Daniel, Toral Marín Sergio. A Dimensional Comparison between Evolutionary Algorithm and Deep Reinforcement Learning Methodologies for Autonomous Surface Vehicles with Water Quality Sensors[J]. Sensors,2021,21(8).
[11] Riccardo Polvara, Sanjay Sharma, Jian Wan, Andrew Manning, Robert Sutton. Autonomous Vehicular Landings on the Deck of an Unmanned Surface Vehicle using Deep Reinforcement Learning[J]. Robotica,2019,37(11).
[12] Wu, D., Lei, Y., He, M., Zhang, C., & Ji, L. (2022). Deep Reinforcement Learning-Based Path Control and Optimization for Unmanned Ships. Wireless Communications and Mobile Computing.
[13] Luis Samuel Yanes, Reina Daniel Gutierrez, Marin Sergio L. Toral. A Multiagent Deep Reinforcement Learning Approach for Path Planning in Autonomous Surface Vehicles: The Ypacarai Lake Patrolling Case[J]. IEEE ACCESS,2021,9.
[14] Wu, D., Lei, Y., He, M., Zhang, C., & Ji, L. (2022). Deep Reinforcement Learning-Based Path Control and Optimization for Unmanned Ships. Wireless Communications and Mobile Computing.
[15] Pan Chao, Peng Zhouhua, Liu Lu, Wang Dan. Data-driven distributed formation control of under-actuated unmanned surface vehicles with collision avoidance via model-based deep reinforcement learning[J]. Ocean Engineering,2023,267.
[16] Shi Jiahui, Liu Zhengjiang. Deep Learning in Unmanned Surface Vehicles Collision-Avoidance Pattern Based on AIS Big Data with Double GRU-RNN[J]. Journal of Marine Science and Engineering,2020,8(9).

[17] Riccardo Polvara, Sanjay Sharma, Jian Wan, Andrew Manning, Robert Sutton. Autonomous Vehicular Landings on the Deck of an Unmanned Surface Vehicle using Deep Reinforcement Learning[J]. Robotica,2019,37(11).
[18] Wu, D., Lei, Y., He, M., Zhang, C., & Ji, L. (2022). Deep Reinforcement Learning-Based Path Control and Optimization for Unmanned Ships. Wireless Communications and Mobile Computing.
[19] Engineering; Findings from JiMei University in the Area of Engineering Reported (Stability Analysis and T-s Fuzzy Dynamic Positioning Controller Design for Autonomous Surface Vehicles Based on Sampled-data Control) [J]. Journal of Engineering,2020.
[20] Peng, Z., Wang, D., Chen, Z., Hu, X., Lan, W.. Adaptive Dynamic Surface Control for Formations of Autonomous Surface Vehicles with Uncertain Dynamics[J]. IEEE transactions on control systems technology: A publication of the IEEE Control Systems Society,2013,21(2).
[21] Hengyang Wang, Biao Liu, Xianyao Ping, Quan An. Path Tracking Control for Autonomous Vehicles Based on an Improved MPC.[J]. IEEE Access,2019,7.
[22] Lakshmanan M, Justindhas Y, Ahamed Ali S, Moorthy A, "Survey on Autonomous Vehicles using Artificial Intelligence", 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC), pp.1237-1241, 2022.
[23] Shu-ping Chen, Guang-ming Xiong, Hui-yan Chen, Dan Negrut. MPC-based path tracking with PID speed control for high-speed autonomous vehicles considering time-optimal travel[J]. Journal of Central South University,2020.
[24] C. L. Darby, D. Garg, and A. V. Rao, Costate estimation using multiple-interval pseudo-spectral methods, J. Spacecraft Rockets, 48 (2011).
[25] Matthew Kelly, An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation, SIAM REVIEW 2017.