ISSN: 2664-9640

Local Path Planning Based on Radial Basis Neural Network and Improved Dynamic Window Algorithm

Wenhu Zhang, Jing Peng *

School of Shanghai Maritime University Merchant marine college, Shanghai 201306, China.

* Corresponding Author

Abstract

Taking Radial Basis Function Neural Network (RBFNN) as the prediction model and Dynamic Window Approach (DWA) algorithm as the local obstacle avoidance algorithm, the local path planning problem of a class of unmanned surface vessel (USV) with unknown navigation environment and dynamic obstacles is studied. According to the position information of the dynamic obstacle in the first four moments, the RBFNN prediction model is used to predict the position information of the obstacle at the next moment, and the dynamic obstacle avoidance problem is converted into the obstacle avoidance of the instantaneous static obstacle. The improved DWA algorithm is used to realize local path planning. The algorithm can improve the real-time performance and safety of dynamic obstacle avoidance, and the simulation experiment proves the feasibility and applicability of the algorithm.

Keywords

USV, local path planning, RBFNN, DWA.

1. Introduction

At present, people are more and more interested in unmanned surface vessels (USV). As one of the key technologies to promote the development of USV, the level of path planning reflects the intelligence of USV to a certain extent [1]. However, most of the current research on path planning mainly focuses on static path planning, and the actual navigation conditions of USVs are complex and changeable and full of uncertain factors. Static path planning cannot meet the requirements of USV intelligent navigation. Therefore, dynamic path planning has gradually become a hot topic of research [2].

USV navigates when the navigation environment is completely unknown or partially unknown. The most important issues to consider are accessibility, safety and timeliness in local path planning. So far, a number of methods and strategies have been proposed to solve path planning problems in unknown environments [3-7], but there is a lack of effective planning methods for unknown environments where dynamic obstacles exist. At present, the commonly used local path planning methods include artificial potential field method [8], simulated annealing algorithm [9], D^* algorithm [10], dynamic window algorithm [11], etc., but these algorithms are not predictive. Therefore, it is difficult to meet the real-time and security requirements of local path planning.

In view of the above problems, this paper adds the prediction function to the obstacle avoidance algorithm to improve the real-time and safety of dynamic obstacle avoidance in local path planning. Commonly used forecasting methods include artificial neural network method, regression analysis method, time series method and grey forecasting method. Among them, artificial neural network, as a parallel computing model, has many advantages that traditional prediction methods do not have, such as better nonlinear mapping ability and less requirement for prior knowledge of the modeled object. These advantages make it stand out among many

ISSN: 2664-9640

forecasting methods. According to the characteristics of artificial neural network, this paper adopts the Radial Basis Function Neural Network (RBFNN) which is simple in structure, convenient in training, fast in learning convergence and can overcome local minima to establish the prediction model. The improved Dynamic Window Approach (DWA) algorithm solves the problem of USV local dynamic obstacle avoidance in a dynamic uncertain environment, and improves the real-time and security of USV dynamic planning.

2. Obstacle Information Preprocessing

2.1. Information collection

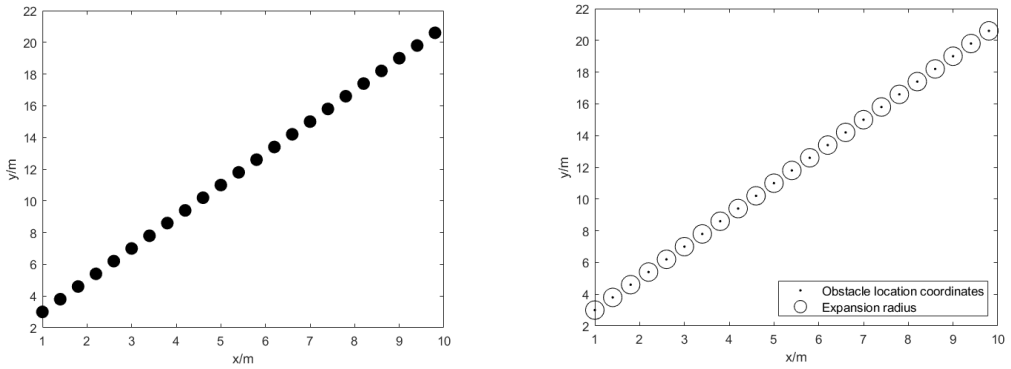
USV uses sensors and other detection devices to collect water surface information, the main purpose is to identify dynamic obstacles and record their moving trajectories. Assuming that the dynamic obstacle is a circle, its center point is taken as the real position point, and the expansion radius of the obstacle is set to ensure the safety of navigation. If the dynamic obstacle moves in a straight line at a uniform speed with the speed v, its motion equation satisfies:

$$y = kx + b \tag{1}$$

At this time, the position coordinates x, y and time t of the dynamic obstacle should satisfy the following relationship:

$$\begin{cases} x = x_0 + vtcos(arctan(k)) \\ y = y_0 + vtsin(arctan(k)) \end{cases} \tag{2}$$

In the above formula (x_0, y_0) is the initial position of the dynamic obstacle. The collected moving trajectory of the dynamic obstacle is shown in Figure 1(a), its real position point and expansion radius are shown in Figure 1(b), and the change trajectory of the position coordinates x and y with time t is shown in Figure 2.



（a）Obstacle moving track ($k = 2, b = 1, v = 2$)

（b）Obstacle location point and expansion radius ($k = 2, b = 1, v = 2$)

Figure 1: Obstacle information

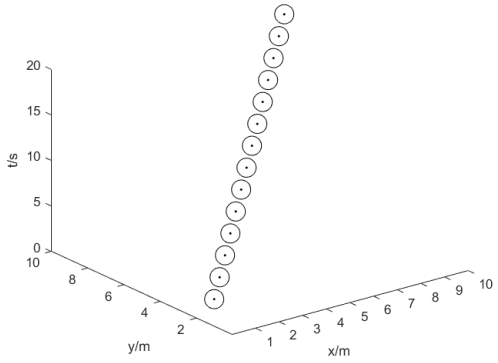


Figure 2: Dynamic obstacle trajectory over time

ISSN: 2664-9640

2.2. Data division

The moving trajectory of dynamic obstacles can be regarded as discrete points that change with time series, assuming this discrete point sequence is: $C = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ Now it is hoped that the position coordinates of the dynamic obstacles at the next s times can be predicted by the position coordinates of the dynamic obstacles at the first l times in the discrete point sequence. As shown in Table 1, a data division method is demonstrated, that is, the collected n discrete points are divided into m groups of data, and each group of data is $l + s$ discrete points.

$$m = n - (l + s) + 1 \tag{3}$$

The data is divided into the following table:

Table 1: Dynamic obstacle position coordinate data division

l Inputs	s Outputs
$(x_1, y_1), \dots, (x_l, y_l)$	$(x_{l+1}, y_{l+1}), \dots, (x_{l+s}, y_{l+s})$
$(x_2, y_2), \dots, (x_{l+1}, y_{l+1})$	$(x_{l+2}, y_{l+2}), \dots, (x_{l+s+1}, y_{l+s+1})$
...	...
$(x_m, y_m), \dots, (x_{m+l-1}, y_{m+l-1})$	$(x_{m+l}, y_{m+l}), \dots, (x_n, y_n)$

The first l values of each set of data are used as the input of RBFNN, and the last s values are used as the output. Through training, the mapping from the input space R^l to the output space R^s can be realized, so as to achieve the purpose of prediction.

Considering the real-time requirements of USV obstacle avoidance, the uncertainty of dynamic obstacle speed, the operating efficiency of the algorithm and other factors, five consecutive sampling values are used as a set of training data, the first four are used as the input of RBFNN, and the last is used as the input of RBFNN. Therefore, the number of nodes in the input layer of the neural network is $l = 4$, and the number of nodes in the output layer is $s = 1$. At this time, the difference equation describing the movement trajectory of the dynamic obstacle can be expressed as:

$$(x_{l+1}, y_{l+1}) = f\big((x_l, y_l) + (x_{l-1}, y_{l-1}) + (x_{l-2}, y_{l-2}) + (x_{l-3}, y_{l-3})\big) \tag{4}$$

The position coordinates (x_{l+1}, y_{l+1}) of the $l + 1$th moment of the dynamic obstacle are the four observations before the union (x_l, y_l), (x_{l-1}, y_{l-1}), (x_{l-2}, y_{l-2}), (x_{l-3}, y_{l-3}) are estimated.

3. Build an RBFNN Prediction Model

3.1. Design of RBFNN

The radial basis function was introduced by Powell in 1985 [12] and is a real-valued function that takes values that depend only on the distance from the origin, i.e., $\Phi(x) = \Phi(\| x \|)$, or the distance to any point c, the point c being called the centroid, i.e., $\Phi(x, c) = \Phi(\| x - c \|)$. Any function Φ that satisfies the characteristic of $\Phi(x) = \Phi(\| x \|)$ is called a radial basis function, and the distance standard of this function is generally Euclidean distance.

RBFNN is an artificial neural network that uses radial basis function as an activation function, and its output is a linear combination of input radial basis function and neuron parameters. RBFNN is a feedforward network with fast learning speed and good nonlinear transformation ability. From the above analysis, it can be concluded that the input layer of the RBFNN should be designed with 4 nodes, the output layer should be designed with 1 node, and the number of nodes in the hidden layer can be adaptively determined during the training process. The topology of the RBFNN is shown in Figure 3. The left side is the input layer, the middle is the hidden layer, and the right side is the output layer.

ISSN: 2664-9640

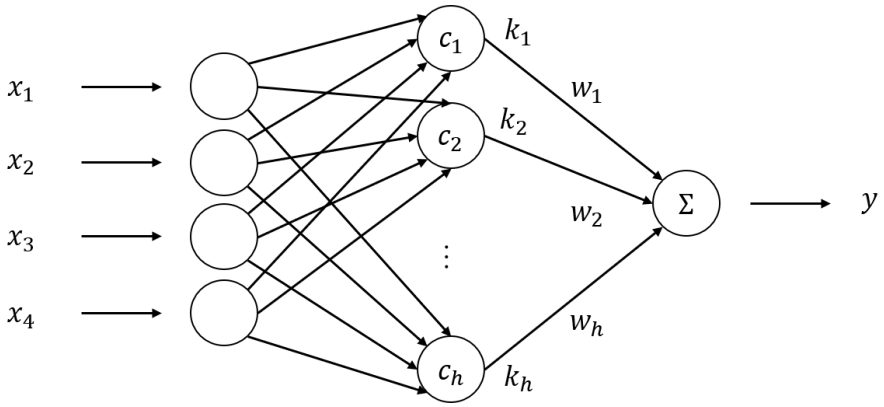


Figure 3: RBFNN topology diagram

The activation function uses a Gaussian kernel function.

$$k(x,c) = e^{-\frac{\|x-c\|^2}{2\sigma^2}} \tag{5}$$

Among them, c is the center point of the kernel function, $\|x-c\|^2$ is the Euclidean distance between the vector x and the vector c, and the influence range of the σ control function, the larger the value, the larger the local influence range, and vice versa, the smaller the influence range. From this, the output of the hidden layer node can be obtained as:

$$k_i = \exp\left(-\frac{1}{2\sigma^2}\|x-c_i\|^2\right) \tag{6}$$

Among them, σ is often replaced by $spread$ in the practical application of MATLAB neural network toolbox, and the relationship between the two can be expressed as:

$$\sigma = \frac{spread}{1.177} \tag{7}$$

At this time, the output of the hidden layer node can be expressed as:

$$k_i = \exp\left(-0.693 \times \left(\frac{\|x-c_i\|}{spread}\right)^2\right) \tag{8}$$

The output of the output layer is the weighted sum of the output of each hidden layer, so the output is:

$$\mathrm{y} = \sum_{i=1}^{h} k_i \times \mathrm{w}_i \tag{9}$$

3.2. 3.2 Training and testing of RBFNN

With the continuous movement of dynamic obstacles, the position information of dynamic obstacles is collected every other sampling period dt, and 16 are collected continuously. Take RBFNN input node $l = 4$, output node $s = 1$, according to this, it is divided into 12 groups of experimental data, the first 8 groups are used as training data, and the last 4 groups are used as test data. The specific data are shown in Table 2.

Table 2: Sample

Sampling point number	Dynamic Obstacle Position Coordinates (x, y)
1	(1.229, 3.456)
2	(2.372, 5.733)
3	(3.508, 8.015)
4	(4.647, 10.291)
5	(5.784, 12.565)

ISSN: 2664-9640

6	(6.922, 14.838)
7	(8.057, 17.114)
8	(9.192, 19.388)
9	(10.335, 21.668)
10	(11.471, 23.940)
11	(12.606, 26.216)
12	(13.751, 28.492)
13	(14.883, 30.772)
14	(16.029, 33.050)
15	(17.162, 35.320)
16	(18.301, 37.597)

In order to improve the training speed of the prediction model, this paper adopts the method of training the horizontal and vertical coordinates of dynamic obstacles separately, and divides the data in Table 2 into training samples and test samples of x and y accordingly. Using the training sample data, take the $spread$ as 2, 4, 6, 8, 10, and 12 to train the RBFNN. After the training is completed, use the test sample to test, and then compare the predicted value with the actual value to obtain the x-coordinate data. As a sample, the error of RBFNN when $spread$ takes different values, as shown in Figure 4.

It can be clearly seen from Figure 4 that when $spread = 8$, the system has the best performance. In RBFNN, $spread = 8$ is taken as the distribution density of the Gaussian kernel function.

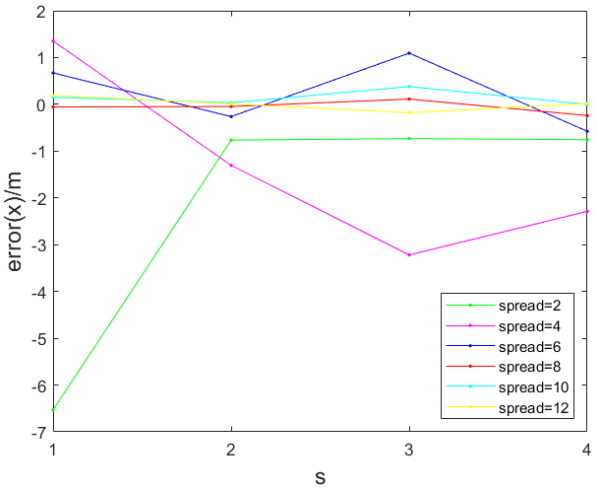


Figure 4: The prediction error $error(x)$ of x when $spread$ takes different values

4. DWA Algorithm

4.1. Dynamic window

In order to ensure the safety of USV autonomous navigation and the convenience of research, the obstacles are expanded, the USV is abstracted into points, and their center points are taken as the actual location points for planning.

Assuming that the obstacles are all circular, the velocity v_o and the rotatable angle θ_o of the dynamic obstacle are within the interval $[v_{min}, v_{max}]$, $[-\theta_{max}, \theta_{max}]$ respectively, and the expansion radius is r, as shown in Figure 5.

ISSN: 2664-9640

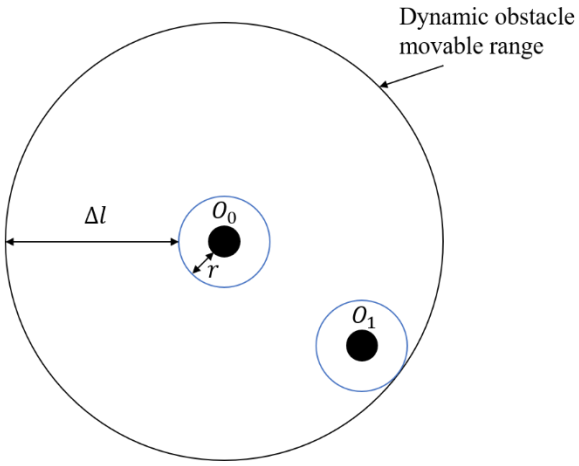


Figure 5: Dynamic obstacles and their range of movement

In the above figure, the black dots represent circular obstacles, O_o represents the current location of the center of the obstacle, and O_1 represents the possible location of the center of the obstacle at the next moment, r represents the expansion radius of the obstacle, and Δl represents the moving distance of the dynamic obstacle within the sampling period dt. If the dynamic obstacle moves in a straight line at a uniform speed, then $\Delta l = v_o \times dt$, and the outermost layer represents the movable range of the dynamic obstacle.

In order to abstract the USV into a point shape, because the detection devices such as sensors are generally installed directly in front of the USV, only the water surface information of 180° in front of the USV can be collected, and because the ship has the limitation of the limit rudder angle. The dynamic window radius thus set is r_d, as shown in Figure 6.

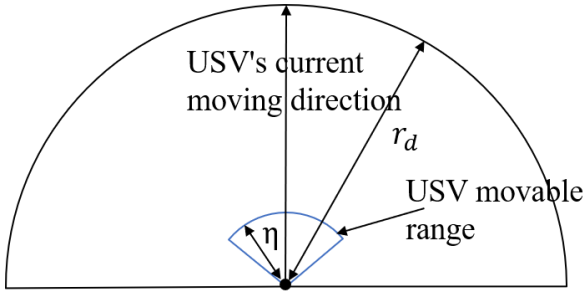


Figure 6: Example of a USV dynamic window

In the above figure, the black dot represents the abstracted USV, and η represents the moving step size of the USV within the sampling period dt. If the USV moves in a straight line at a uniform speed v_d, then $\eta = v_d \times dt$, and r_d represents the dynamic window radius, and the outermost layer represents the dynamic window.

4.2. Dynamic Window Approach algorithm

The Dynamic Window Approach (DWA), proposed by Fox et al [11] in 1997, is one of the classical local path planning algorithms and is mainly used for dynamic obstacle avoidance. The basic idea of the DWA algorithm is to sample multiple sets of linear and angular velocities in velocity space (v, w) and simulate the trajectory of these velocities over a certain period of time, then score these trajectories by an evaluation function, select the path with the highest score, and send the corresponding velocity pair control command to the USV.

The main steps of the algorithm are as follows:

(1) The velocity search space, which is sampled in velocity space according to the following three points.

ISSN: 2664-9640

a. Circular arc trajectory V_s: Based on the kinematic constraints of the USV, a two-dimensional velocity search space is created by considering all possible velocity pairs (v, w) based on their kinematic accessibility, which is limited to a selection of velocity pairs V_s.

b. Allowable speed V_a: This sampling speed will be evaluated if the USV is able to stop before it hits the nearest obstacle. The $dist(v, w)$ is the distance between the USV and the nearest obstacle, and let the acceleration when braking be $\dot{v}_b$ and $\dot{w}_b$, then V_a is the set of velocities at which the USV does not collide with the obstacle.

$$V_a = \left\{ v, w \mid v \le \sqrt{2 * dist(v, w) * \dot{v}_b} \cap w \le \sqrt{2 * dist(v, w) * \dot{w}_b} \right\} \tag{10}$$

c. Dynamic window V_d: Once the allowed velocity V_a has been established, a dynamic window is established. The dynamic window is the sub velocity search space which consists of velocity pairs of velocities V_d that are reachable within the next time interval for a given acceleration constraint. Let dt be the time interval and (v_a, w_a) be the actual velocity pair, the set of velocities for the dynamic window is V_d.

$$V_d = \{ v, w \mid v \in [v_a - \dot{v} * dt, v_a + \dot{v} * dt] \cap w \in [w_a - \dot{w} * dt, w_a + \dot{w} * dt] \} \tag{11}$$

The global velocity search space is based on the circular trajectory V_s of the USV, the allowed velocity V_a and the dynamic window V_d to form the possible velocities V_{space}.

$$V_{\text{space}} = V_s \cap V_a \cap V_d \tag{12}$$

(2) Use the cost function to find the optimal value.

Substitution function:

$$G(v, w) = \alpha * heading(v, w) + \beta * dist(v, w) + \gamma * vel(v, w) \tag{13}$$

Finding the optimal value means finding the maximum value. The value $heading$ is used to evaluate the angle between the USV and the target point, and is taken to be the maximum value when the USV is moving towards the target point. The $dist$ is used to represent the distance of the USV from the nearest obstacle. The vel represents the forward movement speed of the USV. The weighting constants α, β and γ are the values between [0, 1] that affect the optimal value of the cost function. The optimal value of the surrogate function affects the trajectory of the USV from the starting point to the target point. A higher weight assigned to $dist(v, w)$ and $vel(v, w)$ tends to cause the trajectory not to move towards the target point. If the bias towards $vel(v, w)$ means that the USV has no incentive to maintain a safe distance away from obstacles in its path, it is easy to see that the three sub-functions of the cost function and their weights all contribute to the choice of the best velocity pair, which defines the trajectory for each sampling period.

4.3. Planning strategies based on RBFNN and improved DWA algorithms

Based on the DWA algorithm, the USV requires a rolling plan for every further row, which takes longer. In order to improve the real-time obstacle avoidance, this paper adds the determination of the start and end of obstacle avoidance and the operation of resuming navigation to the DWA algorithm. The decision can be briefly described as when the distance d_n between the USV's own position and the nearest obstacle is within the dynamic window V_d reachable range, then obstacle avoidance starts, the optimal value (v, w) in the cost function is chosen and this velocity pair is passed to the USV, when d_n is outside the dynamic window V_d reachable range, it is not considered and obstacle avoidance ends. If the heading of the USV is not pointing to the target point at this time, the linear velocity is gradually adjusted within the global velocity search space V_{space} to maximize it, and the angular velocity is adjusted to make the USV heading point to the target point. Taking Figure 7 as an example, the planning strategy based on RBFNN and improved DWA algorithm is:

(1) When the USV moves to U_1, the dynamic obstacle is detected at the position of O_1 point, which is outside the dynamic window V_d, and the USV continues to move forward in the direction towards the target point.

ISSN: 2664-9640

(2) When the USV moves to U_2, the dynamic obstacle moves to the position of O_2, and it has entered the rolling window at this time. Before the USV makes the next movement decision, the position of the dynamic obstacle for the next sampling cycle, assumed to be O_3, is first predicted from the coordinates of the trajectory of the most recently detected dynamic obstacle for the previous 4 moments and the RBFNN prediction model, and the inflated dynamic obstacle is placed into the path planning map as a static obstacle to start obstacle avoidance using the improved DWA algorithm.

(3) Evaluate the best speed pair through the cost function, send the corresponding speed pair control command to the USV, and set the position of the USV as U_3 at this time.

(4) Once again, the dynamic obstacle's next moment position, assumed to be O_4, is predicted by the detected dynamic obstacle's moving trajectory coordinates of the previous 4 moments and the RBFNN prediction model, and this dynamic obstacle is found to have left the dynamic window after the expansion process, and obstacle avoidance is over.

(5) Gradually adjust the linear velocity of the USV within the range of the global velocity search space V_{space} to maximize it, and adjust the angular velocity to make the heading point to the target point.

(6) Repeat the similar planning process from step (1) until the USV reaches the target point.

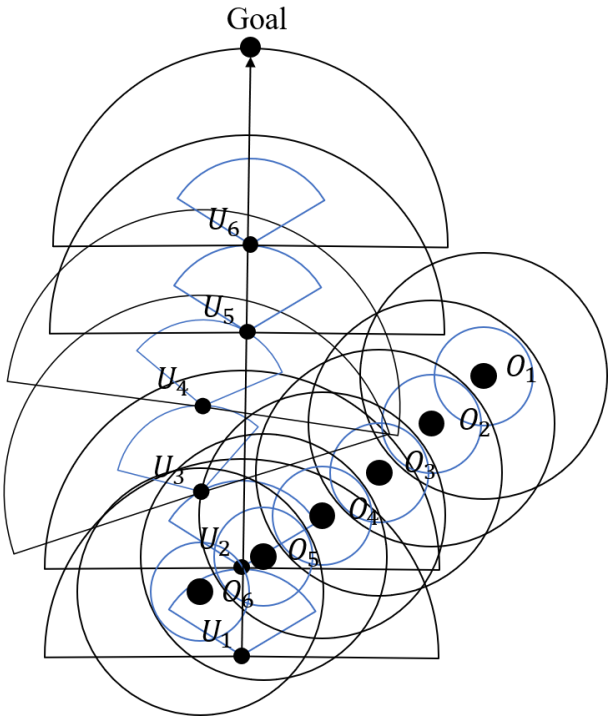


Figure 7: Example of local obstacle avoidance based on RBFNN and improved DWA algorithm

5. Simulation experiments

5.1. Local Path Planning under Static Obstacles

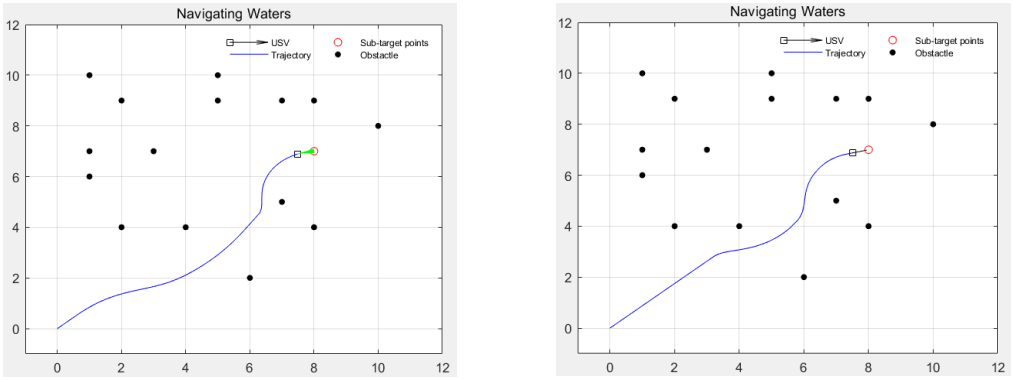
In order to verify the effectiveness of the improved DWA algorithm proposed in this paper, the path segment with only static obstacles in the navigation environment is selected for local path planning. In the simulation experiment, the starting point of the USV is (0,0), and the sub-target point is (8,7). In order to make the generated USV trajectory closer to the real motion model, restrictions are set on the parameters related to path planning, as shown in the following table:

ISSN: 2664-9640

Table 3: Parameters related to local path planning

Parameters related to local path planning	value
Inflation Radius ($grid$)	0.5
Sampling period (s)	0.1
Maximum linear speed (m/s)	1
Maximum angular velocity ($°/s$)	20
Maximum linear acceleration (m/s^2)	0.2
Maximum angular acceleration ($°/s^2$)	50
Linear velocity resolution (m/s)	0.01
Angular velocity resolution ($°/s$)	1
Sub-target Coverage Radius ($grid$)	0.5

Where the linear and angular velocity resolution is essentially the minimum velocity increment relative to the sampling time and the sub-target covers a radius of 0.5 grids, i.e., the USV is considered to have arrived when it is less than 0.5 grids away from the sub-target, The movement of the USV location point is guided by linear and angular velocities, maximizing the navigation capability at each time step. At this time, the path of the USV from the starting point to the sub-target point is shown in Figure 8. Figure 8(a) is the trajectory of the USV from the starting point to the sub-target point after using the traditional DWA algorithm, which takes 176.52s. Figure 8(b) In order to adopt the improved DWA algorithm, the trajectory of USV from the starting point to the sub-target point takes 126.90s, which is 28% shorter.



(a) Traditional DWA algorithm (b) Improved DWA algorithm

Figure 8: Example of local path planning (only static obstacles exist)

5.2. Local path planning under dynamic obstacles

In order to verify the applicability of the improved DWA algorithm proposed in this paper for dynamic obstacles, the path segments with multiple static obstacles and one dynamic obstacle in the USV navigation environment are selected for local dynamic obstacle avoidance. The algorithm parameters are the same as the local path planning under static obstacles, which will not be repeated here. The starting point of the USV is (0,0), the sub-target point is (6,5), the dynamic obstacle moves in a straight line at a constant speed of $0.2m/s$ along $y = 3$, and the starting point of the dynamic obstacle is (2,3).

During the process of USV reaching the sub-goal point from the starting point, the dynamic obstacle moves from point O_1 to point O_n. The dynamic obstacle enters the dynamic window of the USV when it moves to the point O_m. The RBFNN prediction model predicts that the dynamic obstacle will collide with the USV at the next moment at the point O_{m+1}, at which time the dynamic obstacle's position is replaced by the static obstacle in this moment for obstacle avoidance, which solves the local dynamic obstacle avoidance problem and successfully reaches the sub-target point, as shown in Figure 9.

ISSN: 2664-9640

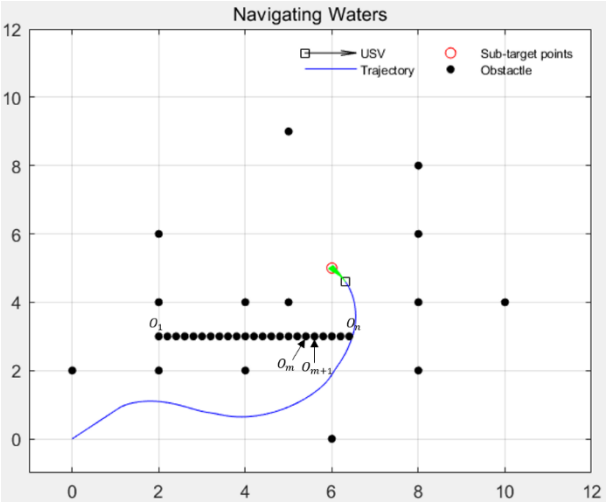


Figure 9: Example of local path planning (dynamic and static obstacle coexistence)

6. Conclusion

USV path planning is carried out when the navigational environment is totally or partially unknown. Due to the lack of global priority information, effective global path planning cannot be carried out, so only partial information obtained from detection devices such as sensors can be relied upon for local path planning. The traditional local path planning method lacks the prediction effect of dynamic obstacles, which makes it difficult to achieve the requirements of real-time and safety of dynamic obstacle avoidance. Therefore, this paper adopts RBFNN as the prediction model to predict the position coordinates of dynamic obstacles at the next moment, uses static obstacles instead of the position at this moment, and adopts the improved DWA algorithm to avoid obstacles in advance to achieve local path planning, so as to solve the dynamic obstacle avoidance problem.

Acknowledgements

Natural Science Foundation.

References

[1] Y. Liu and R. J. O. E. Bucknall, "Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment," vol. 97, pp. 126-144, 2015.
[2] Liu Wen, Hu Kunlin, Li Yan, and Liu. J. Chinese Journal of Intelligent Science and Technology, "A review of prediction methods for moving target trajectories," vol. 3, no. 2, pp. 149-160, 2021.
[3] Y. Abe, M. Shikano, T. Fukuda, F. Arai, and Y. Tanaka, "Vision based navigation system by variable template matching for autonomous mobile robot," in Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), 1998, vol. 2, pp. 952-957: IEEE.
[4] J. Borenstein, Y. J. I. t. o. r. Koren, and automation, "The vector field histogram-fast obstacle avoidance for mobile robots," vol. 7, no. 3, pp. 278-288, 1991.
[5] P. S. Lee and L. L. J. J. o. r. s. Wang, "Collision avoidance by fuzzy logic control for automated guided vehicle navigation," vol. 11, no. 8, pp. 743-760, 1994.
[6] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in Proceedings of IEEE international conference on robotics and automation, 1996, vol. 4, pp. 3375-3382: IEEE.
[7] J. Xiao, Z. Michalewicz, L. Zhang, and K. J. I. t. o. e. c. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," vol. 1, no. 1, pp. 18-28, 1997.
[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in Autonomous robot vehicles: Springer, 1986, pp. 396-404.

ISSN: 2664-9640

[9] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. J. s. Vecchi, "Optimization by simulated annealing," vol. 220, no. 4598, pp. 671-680, 1983.
[10] A. Stentz, "Optimal and efficient path planning for partially-known environments," in Proceedings of the 1994 IEEE International Conference on Robotics and Automation, 1994, pp. 3310-3317: IEEE.
[11] D. Fox, W. Burgard, S. J. I. R. Thrun, and A. Magazine, "The dynamic window approach to collision avoidance," vol. 4, no. 1, pp. 23-33, 1997.
[12] M. Powell, "Radial basis function for multivariable approximations: A review," in Proc. IMA Conf. On Algorithms for the Approx. Of Functions and Data. Oxford, 1985, pp. 143-167.