The Journal of Navigation (2023), **76**:4–5 487–505
doi:10.1017/S037346332300019X

RESEARCH ARTICLE

Radar-based path planning of autonomous surface vehicle with static and dynamic obstacles in a Frenet Frame

Zhihuan Hu, Ziheng Yang, Xiaocheng Liu, and Weidong Zhang

Department of Automation, Shanghai Jiao Tong University, Shanghai, China.
Corresponding author: Weidong Zhang; Email: wdzhang@sjtu.edu.cn

Received: 21 August 2022; **Accepted:** 28 May 2023

Keywords: path planning; autonomous surface vehicle; collision avoidance; millimetre wave radar

Abstract

Navigation safety at sea is vital for each autonomous surface vehicle (ASV), which involves the problem of motion planning in dynamic environments and their robust tracking through feedback control. We present a practical path-planning method that generates smooth trajectories for a marine vehicle traveling in an unknown environment, where obstacles are detected in real time by millimetre wave (mmWave) radar. Our approach introduces a polynomial curve to describe the lateral and longitudinal trajectories in the Frenet frame, known as the 'motion primitives', whose combination ensures that the planning area is properly covered. In addition, we can select a feasible, optimal and collision-free trajectory from such a set of motion primitives that is generated by considering the vehicle dynamics and comfort. The capabilities of proposed algorithm are demonstrated in the experiment with static and dynamic obstacles.

1. Introduction

ASVs are vehicles capable of operating safely with little or no human input. The autonomous technology reveals a potential to enhance the safety, efficiency and convenience of maritime transport. The motion planning system is essential for ASVs, which ensures that a vehicle navigates through the water with safety. Clearly, there is a lot of research work being done on this motion-planning algorithm.

Generally, the motion-planning problem can be partitioned into a hierarchical structure (González et al., 2015): at the top level, the route and mission for a vehicle are computed on a scale of continent; the behavioural layer decides on a local navigational scenario while following the *International Regulations for Preventing Collisions at Sea* (COLREGS) (Cockcroft and Lameijer, 2003). As shown in Figure 1, a path-planning module then generates a continuous trajectory to accomplish the short-term objectives, such as collision avoidance, speed maintenance and stopping, and so on. Here, much attention has been paid to the path-planning algorithm, and thus the COLREGS is not considered in our approach.

A number of techniques have been developed to address the challenges of traffic-adapted trajectory. The resulting trajectory is expressed as a time-varying function $\pi(t) : [0, \Delta T] \rightarrow \mathit{\Gamma}$, where $\mathit{\Gamma}$ is the configuration space of a robot, and ΔT is the planning horizon. The problem of determining such a trajectory could be easily stated as a non-convex optimisation problem. However, because the practical methods for finding any exact solution are unavailable (Reif and Wang, 1998), one must resort to the heuristic algorithms as a reasonable approximation for the optimal solution.

The geometric method represents the obstacles and vehicles as convex bodies. Accounting for the robot constraints, these methods are useful for computing the lower and upper bounds on the feasible area or trajectory. Especially for the planar motions ($\Gamma \subseteq \mathbb{R}^2$), there are several efficient algorithms with polygonal representation of obstacles, such as generalised Voronoi diagrams (Takahashi and Schilling,

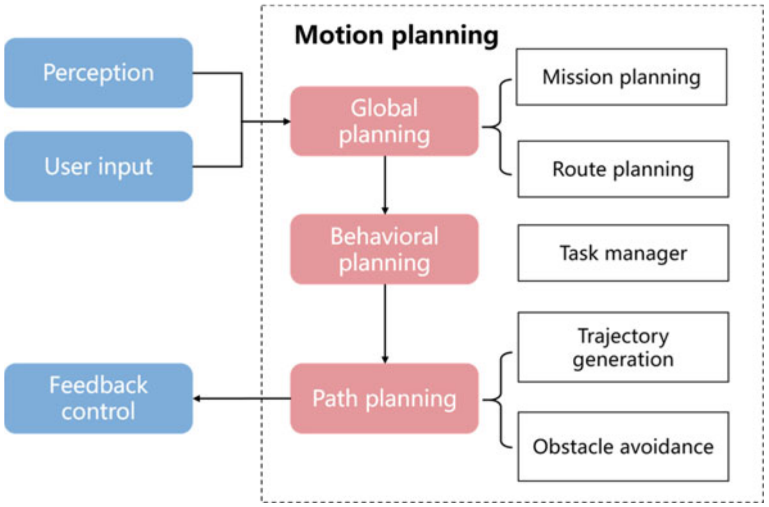

Figure 1. *Motion planning architecture for ASVs. It consists of global planning, behavioural layers and local path planning (Paden et al., 2016).*

1989) and the velocity obstacles method (Kuwata et al., 2013). The COLREGS could also be encoded in the velocity space of ASVs and obstacles using the velocity obstacles method.

The graph search methods, including the Dijkstra algorithm, A* algorithm and D* algorithm and their modified version, have been implemented in a wheeled mobile robot (Howard and Kelly, 2007), a self-driving car (Dolgov et al., 2008) and ASVs (Blaich et al., 2012a), and so on. For the graph-based search, the state space of a robot is discretised as a graph, where a vertex can be seen as a vehicle state and the edges represent state transitions. The goal is to find a path with the smallest cost in the graph. For instance, in Blaich et al. (2012b) and Schuster et al. (2014), the modified A* algorithm is presented with a T-shaped neighbourhood, which is used to take the physical limits of a vehicle into account. Moreover, the hybrid A* method, along with the Voronoi potential field, could be implemented in the auto-docking system and shuttle ferry, which require a complicated manoeuvre at slow speeds.

Curve-interpolating methods are commonly used to construct a new path with a set of waypoints given by the high-level planner. The implementation of curves such as polynomials, splines, lines and circles are simple and computationally efficient. In Wilthil et al. (2018), the authors present a dynamic window algorithm, using the linear combination of forward speed and rate of turn to construct a desired path. And an extended dynamic window method (Eriksen et al., 2019) is developed using the linear piecewise functions of acceleration. A path-tracking algorithm then ensures that the desired path is achieved by the line-of-sight method and model predictive control. However, the line and circle interpolating method may be not continuous, reducing passenger comfort. In addition, the line and circle interpolating methods are less robust to the noisy tracking of obstacles, causing the unsteady planning manoeuvre. These issues motivate us to develop a new path-planning algorithm with more robust capability and less computational complexity. The paper consists of two main contributions: first, it introduces a cost function to penalising the jerk of a vehicle, which could generate a smooth trajectory considering the passenger comfort. This also allows for the modelling of acceleration constraints and the reactive obstacle avoidance in unexpected situations. Second, the proposed algorithm is computationally efficient and will ensure real-time performance. The polynomial curves are implemented to construct a feasible path, allowing for the replanning frequency of 100 Hz. Along with the radar-based target tracking and situational awareness of obstacles in real time, the experiments have been performed with static and dynamic obstacles.

2. Theoretical background

2.1. *Vessel model*

Let $\boldsymbol{\eta}(t) = [x, y, \varphi]$ be the state of vessel, which contains the x, y coordinates and heading (φ) in the global reference frame. Let $\boldsymbol{v}(t) = [u, v, r]$ be the velocity state, which contains the linear speed (u, v) and rate of turning (r) in the body-fixed reference frame. The 3-degree of freedom (DoF) marine craft equations of motion can be introduced as follows:

$$\dot{\boldsymbol{\eta}}(t) = \boldsymbol{T}(\boldsymbol{\eta}(t)) \cdot \boldsymbol{v}(t) \tag{1}$$

$$\boldsymbol{M} \cdot \dot{\boldsymbol{v}}(t) + \boldsymbol{D} \cdot \boldsymbol{v}(t) = \boldsymbol{\tau}(t) + \boldsymbol{\tau}_\mathrm{e} \tag{2}$$

$$\boldsymbol{T}(\boldsymbol{\eta}(t)) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Here, $\boldsymbol{M}$ and $\boldsymbol{D}$ denote the time-invariant mass and damping matrix. The element of matrix $\boldsymbol{M}$ contain the mass, the moment of inertia and added mass. The linear drag coefficients and nonlinear drag coefficients are considered in the damping matrix $\boldsymbol{D}$. More detailed description of the elements of these matrices can be found in Fossen (2011). $\boldsymbol{\tau}(t)$ and $\boldsymbol{\tau}_\mathrm{e}$ indicate the propulsive force and environmental load on the marine craft. Let $\boldsymbol{x}(t) = [\boldsymbol{\eta}(t), \boldsymbol{v}(t)]$ be the state vector, the state dynamics are given by,

$$\dot{\boldsymbol{x}}(t) = A_c \cdot \boldsymbol{x}(t) + B_c \cdot \boldsymbol{u}(t) + \omega(t) \tag{4}$$

For a vessel,

$$\boldsymbol{u}(t) = \boldsymbol{\tau}(t), \quad \omega(t) = \boldsymbol{\tau}_\mathrm{e}$$

$$A_c = \begin{bmatrix} 0 & \boldsymbol{T}(\boldsymbol{\eta}(t)) \\ 0 & -\boldsymbol{M}^{-1}\boldsymbol{D} \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ \boldsymbol{M}^{-1} \end{bmatrix}$$

Let $\boldsymbol{\tau}(k) = \boldsymbol{\tau}(t_k)$, $\boldsymbol{x}(k) = \boldsymbol{x}(t_k)$, $\boldsymbol{x}(k+1) = \boldsymbol{x}(t_{k+1})$, where t_k denotes the discrete time. Then the discrete-time state space model can be described as follows:

$$\begin{gathered} \boldsymbol{x}(k+1) = \boldsymbol{A} \cdot \boldsymbol{x}(k) + \boldsymbol{B} \cdot \boldsymbol{\tau}(k) \\ \boldsymbol{A} = \boldsymbol{I} + \Delta t \cdot A_c, \quad \boldsymbol{B} = \Delta t \cdot B_c \end{gathered} \tag{5}$$

where the time duration $\Delta t = t_{k+1} - t_k$. Such a discrete–time state space model enables us to perform the real-time Kalman filtering (Bishop and Welch, 2001).

2.2. *Collision risk assessment*

In this study, collision risk is assessed using the closest point of approach (CPA) and time to closest point of approach (TCPA). The d_{CPA} (see Figure 2) is used to describe the closest distance if the own ship and target ship maintain course and speed. A small d_{CPA} means the two ships will collide. Let $[x_0,\ y_0]$ be the initial position of the own ship, $[v_x,\ v_y]$ denote the velocity vector of the own ship, and the position of the own ship can be given by:

$$x(t) = v_x t + x_0$$

$$y(t) = v_y t + y_0$$

Let $[\hat{x}_0,\ \hat{y}_0]$ and $[\hat{v}_x,\ \hat{v}_y]$ be the initial position and velocity of target ship:

$$\hat{x}(t) = \hat{v}_x t + \hat{x}_0$$

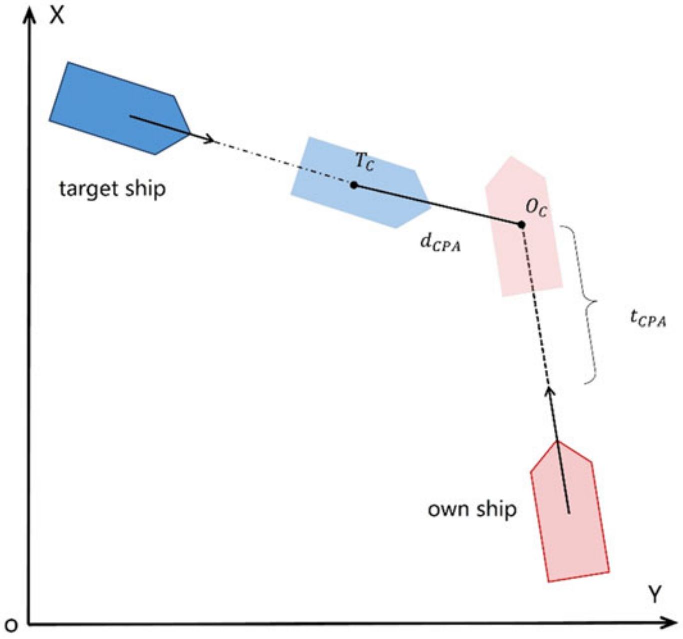

Figure 2. *CPA and TCPA.*

$$\hat{y}(t) = \hat{v}_y t + \hat{y}_0$$

The distance between the own ship and target ship can be computed:

$$D^2 = (x(t) - \hat{x}(t))^2 + (y(t) - \hat{y}(t))^2$$

The time derivative of D^2:

$$\frac{dD^2}{dt} = ((v_x - \hat{v}_x)t + (x_0 - \hat{x}_0))(v_x - \hat{v}_x) + ((v_y - \hat{v}_y)t + (y_0 - \hat{y}_0))(v_y - \hat{v}_y) = 0$$

With such information, we can compute the position $[x_{\mathrm{CPA}},\ y_{\mathrm{CPA}}]$ of target ship at t_{CPA}(TCPA):

$$t_{\mathrm{CPA}} = \max\left(0, \frac{(x_0 - \hat{x}_0)(v_x - \hat{v}_x) + (y_0 - \hat{y}_0)(v_y - \hat{v}_y)}{(v_x - \hat{v}_x)^2 + (v_y - \hat{v}_y)^2}\right) \tag{6}$$

$$x_{\mathrm{CPA}} = \hat{v}_x t_{\mathrm{CPA}} + \hat{x}_0, \quad y_{\mathrm{CPA}} = \hat{v}_y t_{\mathrm{CPA}} + \hat{y}_0 \tag{7}$$

The collision risk can be evaluated based on these formulas, as well as the relative distance, velocity and azimuth of target ship measured by the mmWave radar.

2.3. *Frenet frame*

The trajectory tracking and planning of ASVs forces a vehicle to reach and follow a time-parameterised geometric path (i.e. orientation, curvature), namely the reference line. Such a reference line can be represented in different forms (e.g. spline, Euler spiral, etc.). However, the vehicle will adjust the motion behaviour for obstacle avoidance. Instead, such behaviours cannot follow through the reference line exactly and thus result in the trajectory replanning.

The Frenet frame (Frenet, 1852) introduces a novel way to plan trajectories to manoeuvre a robot. It is more intuitive for Frenet frame to represent a vehicle's motion than the traditional Cartesian coordinate.

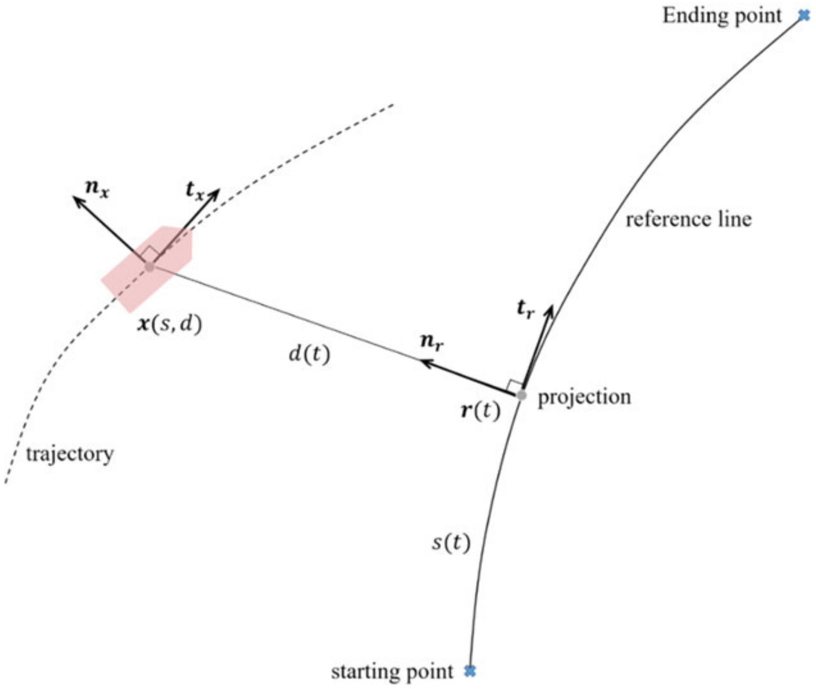

Figure 3. *Trajectory generation based on the reference line in the Frenet frame.*

With the Frenet coordinate (see Figure 3), we use the variables $s(t)$ to represent the arclength along which the vehicle has moved on the reference line in time t, and $d(t)$ to describe the perpendicular offset with respect to the reference line. These variables $s(t)$ and $d(t)$ are also known as longitudinal and lateral displacement, respectively. And the Cartesian coordinate of the resulting trajectory $\boldsymbol{x}(s(t),\ d(t))$ can be given by the normal and tangential vector $\boldsymbol{n_r},\ \boldsymbol{t_r}$ at a certain point on the reference line:

$$\boldsymbol{x}(s(t),\ d(t)) = \boldsymbol{r}(s(t)) + d(t)\ \cdot \boldsymbol{n_r}(s(t)) \tag{8}$$

where the vectors $\boldsymbol{n_r},\ \boldsymbol{t_r}$ are defined at the projection of the vehicle onto the reference line, and $\boldsymbol{r}(s(t))$ denotes the Cartesian coordinate of such a projection. $\boldsymbol{n_x},\ \boldsymbol{t_x}$ are normal and tangential vectors of the resulting trajectory. Since the resulting trajectory and reference line provide the tracking reference for the vehicle's controller, higher-order parameters must be given: the speed v, the orientation θ, the curvature κ and the acceleration a.

2.4. Coordinate transformation

The coordinate transformation between the Frenet and Cartesian frames makes it possible to perform online path planning for a vehicle. Given the motion $[s, \dot{s}, \ddot{s}; d, \dot{d}, \ddot{d}]$ or $[s, \dot{s}, \ddot{s}; d,\ d', d'']$ in the Frenet frame, and $[\boldsymbol{x}, \theta_x, \kappa_x,\ v_x, a_x]$ in the Cartesian frame, we let $\dot{(\cdot)} := \frac{\partial}{\partial t}(\cdot)$ be the partial derivative with respect to time and $(\cdot)' := \frac{\partial}{\partial s}(\cdot)$ denote the partial derivative with respect to arclength. In this paper, the vehicle is assumed to travel along the reference line, excluding extreme situations. In this case, we have $1 - \kappa_r d > 0$ and $|\Delta\theta| < \pi/2$ with $\Delta\theta := \theta_x - \theta_r$.

Applying the Frenet-Serret formulas (Kühnel, 2015):

$$\frac{\mathrm{d}\boldsymbol{n}_r}{\mathrm{d}s} = -\kappa_r \boldsymbol{t}_r, \quad \frac{\mathrm{d}\boldsymbol{n}_r}{\mathrm{d}t} = -\dot{s}\kappa_r \boldsymbol{t}_r \tag{9}$$

$$\frac{\mathrm{d}\boldsymbol{t}_r}{\mathrm{d}s} = \kappa_r \boldsymbol{n}_r, \quad \frac{\mathrm{d}\boldsymbol{t}_r}{\mathrm{d}t} = \dot{s}\kappa_r \boldsymbol{n}_r \tag{10}$$

Where κ_r denotes the curvature of reference line:

$$\boldsymbol{t}_r(s) = [\cos\theta_r(s) \ \sin\theta_r(s)]^{\mathrm{T}} \quad \boldsymbol{n}_r(s) = [-\sin\theta_r(s) \ \cos\theta_r(s)]^{\mathrm{T}}$$

With Equation (8) and Frenet-Serret formulas, we have:

$$d = [\boldsymbol{x} - \boldsymbol{r}(s)]^{\mathrm{T}} \cdot \boldsymbol{n}_r \tag{11}$$

$$\dot{d} = [\dot{\boldsymbol{x}} - \dot{\boldsymbol{r}}(s)]^{\mathrm{T}} \cdot \boldsymbol{n}_r + [\boldsymbol{x} - \boldsymbol{r}(s)]^{\mathrm{T}} \cdot \dot{\boldsymbol{n}}_r = v_x \sin\Delta\theta \tag{12}$$

For the vehicle's speed v_x on the resulting trajectory, the time derivative of $\boldsymbol{x}$ yields:

$$v_x = ||\dot{\boldsymbol{x}}||_2 = \sqrt{(1-\kappa_r d)^2 \dot{s}^2 + \dot{d}^2}$$

With v_x, we calculate:

$$d' = \frac{\dot{d}}{\dot{s}} = \frac{v_x \sin\Delta\theta}{\dot{s}} = \sin\Delta\theta \cdot \sqrt{(1-\kappa_r d)^2 + d'^2} \Rightarrow d' = (1-\kappa_r d)\tan\Delta\theta \tag{13}$$

Where $||\cdot||_2$ is the Euclidean norm. Taking the time derivative for $(\boldsymbol{x} - \boldsymbol{r}(s))^{\mathrm{T}}\boldsymbol{t}_r = 0$, we also have:

$$\dot{s} = \frac{v_x \cos\Delta\theta}{1-\kappa_r d} \tag{14}$$

Let s_x represent the arclength of the resulting trajectory $\boldsymbol{x}$, and the curvatures $\kappa_x = \mathrm{d}\theta_x/\mathrm{d}s_x$, yields:

$$\Delta\theta' = \frac{\mathrm{d}(\theta_x - \theta_r)}{\mathrm{d}s} = \frac{1-\kappa_r d}{\cos\Delta\theta}\kappa_x - \kappa_r$$

Taking derivative of Equation (13) with respect to s yields:

$$d'' = -(\kappa_r d)' \tan\Delta\theta + \frac{1-\kappa_r d}{\cos^2(\Delta\theta)}\Delta\theta' \tag{15}$$

The acceleration of resulting trajectory is given by:

$$a_x := \dot{v}_x = \ddot{s}\frac{1-\kappa_r d}{\cos\Delta\theta} + \frac{\dot{s}^2}{\cos\Delta\theta}[d'\Delta\theta' - (\kappa_r d)']$$

With a_x, we can calculate the second order derivative of arclength:

$$\ddot{s} = \frac{a_x \cos\Delta\theta - \dot{s}^2[d'\Delta\theta' - (\kappa_r d)']}{1-\kappa_r d} \tag{16}$$

The time derivative of d is given by:

$$\dot{d} = \frac{\mathrm{d}s}{\mathrm{d}t}\frac{\mathrm{d}}{\mathrm{d}s}d = \dot{s}d' \tag{17}$$

$$\ddot{d} = d''\dot{s}^2 + d'\ddot{s} \tag{18}$$

2.4.1. Transform Cartesian frame to Frenet frame
Given the vehicle's motion $[\boldsymbol{x}, \theta_x, \kappa_x, v_x, a_x](t_0)$, we need to project it onto the reference line and derive the corresponding states $[s_0, \dot{s}_0, \ddot{s}_0; d_0, \dot{d}_0, \ddot{d}_0]$ or $[s_0, \dot{s}_0, \ddot{s}_0; d_0, d'_0, d''_0]$. For instance, when the vehicle is located at the starting point of the reference line, we have $s_0 = 0$, $d_0 = 0$ in the Frenet frame. Firstly, the arclength s_0 can be determined by:

$$s_0 = \arg\min_{\sigma} ||\boldsymbol{x} - \boldsymbol{r}(\sigma)||_2$$

The numerical method for solving the previous minimum problem is trivial. As the function of the reference line is known, we can calculate more information, including the curvature κ_r, κ'_r and the orientation θ_r when the arclength s_0 is determined. With $\Delta\theta = \theta_x - \theta_r$ and Equations (9)–(16), the remain variables such as $\dot{s}_0, \ddot{s}_0, d_0, \dot{d}_0, \ddot{d}_0$, d'_0, d''_0 can also be computed.

2.4.2. Transform Frenet frame to Cartesian frame
When the state lattices are generated, we need to determine the corresponding states $[\boldsymbol{x}, \theta_x, \kappa_x, v_x, a_x]$ in the Cartesian coordinate. This can be done using Equations (9)–(16), provided that $\dot{s} > 0$ for a vehicle travelling forward.

2.5. *Optimal trajectory generation*

To reduce the computational complexity of path planning, the heuristic trajectory-generation algorithm is proposed by selecting the optimal trajectory from a discrete set of manoeuvres covering the free configuration space. When constructing the manoeuvre sets, referred to as 'state lattice', we consider comfort, vehicle constraints, curve smoothness and obstacle avoidance. The comfort is mathematically described by the jerk, which is defined by the rate of acceleration change with respect to time. The following theorem enables us to generate a feasible and comfortable path.

Theorem: Given the state $P_0 = [p_0, \dot{p}_0, \ddot{p}_0]$ at the start time t_0, and $P_1 = [\dot{p}_1, \ddot{p}_1]$ at $t_1 = t_0 + \Delta T$, a quantic polynomial is the minima of the cost function:

$$C = k_J J_t + k_g g(\Delta T) + k_h h(p_1)$$

$$J_t(p(t)) := \int_{t_0}^{t_1} \dddot{p}^2(\tau)\mathrm{d}\tau$$

Where $g(\cdot)$ and $h(\cdot)$ are arbitrary functions and $k_J, k_g, k_h > 0$.

The proof of this theorem is given in Takahashi et al. (1989). The term $\dddot{p}(t)$ is known as 'jerk' and $J_t(p(t))$ can qualitatively describe the jerk within the time interval ΔT. From Werling et al. (2010), we know that, in the Frenet frame, the lateral $d(t)$ and longitudinal movement $s(t)$ can be determined independently for vehicles at higher speed. Thus, the state lattice could be generated by combining lateral and longitudinal trajectories. As for the jerk, $\dddot{d}$ and $\dddot{s}$ are introduced.

2.5.1. Lateral movement
Given the start state $D_0 = [d_0, \dot{d}_0, \ddot{d}_0]$, the cost function is expressed as:

$$C_{\text{lat}} = k_J J_t(d(t)) + k_t \Delta T + k_d d_1^2$$

We let $g(\Delta T) = \Delta T$ and $h(d_1) = d_1^2$, to penalise the slow convergence and large lateral offset. The vehicle is suggested to travel parallel to the reference line, which means $\dot{d}_1 = \ddot{d}_1 = 0$ at end time $t_0 + \Delta T$. The optimal solution for Equation (19) is a quintic polynomial in the indeterminate t

$$d(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{19}$$

Different coefficients in the end state are considered to generate a set of sufficiently covered and collision-free manoeuvres. It means the state lattice in the lateral direction Π_{lat} consists of polynomials with different end conditions d_1 and ΔT:

$$[d_1, \dot{d}_1, \ \ddot{d}_1, \Delta T]_{ik} = [d_i, 0, 0, \Delta T_k]$$

2.5.2. Longitudinal movement

Usually, marine vehicle moves by either keeping a desired speed $\dot{s}_d$ = const or by changing the course. The course-changing manoeuvre can be achieved by the lateral movement that we have mentioned. For the speed-keeping behaviour, a quartic polynomial:

$$\dot{s}(t) = a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

can be used to minimise the cost function:

$$C_{lon} = k_J J_t(s(t)) + k_t \Delta T + k_{\dot{s}}(\dot{s}_1 - \dot{s}_d)^2 \tag{20}$$

whose start state $S_0 = [s_0, \dot{s}_0, \ddot{s}_0]$ at t_0, and end state $S_1 = [\dot{s}_1, \ \ddot{s}_1]$ at $t_1 = t_0 + \Delta T$. The acceleration is undesirable, resulting in $\ddot{s}_1 = 0$. By slightly varying the end conditions $\dot{s}_1$, we can determine the coefficients for the quartic polynomial:

$$[\dot{s}_1, \ \ddot{s}_1, \Delta T]_{jk} = [\dot{s}_d + \Delta\dot{s}_j, \ 0, \Delta T_k]$$

In this case, the state lattice in the longitudinal direction Π_{lon} is generated.

2.5.3. Combined trajectory

The state lattice is generated by taking the Cartesian product of lateral movement and longitudinal movement $\Pi_{\text{lat}} \times \Pi_{\text{lon}}$. The total cost of each trajectory in the state lattice can be computed by simple algebraic operation:

$$C_{\text{tot}} = k_{\text{lat}} C_{\text{lat}} + k_{\text{lon}} C_{\text{lon}}$$

where $k_{\text{lat}}, \ k_{\text{lon}} > 0$. The state lattice is shown in Figure 4.

2.6. Path-following algorithm

The path-following algorithm forces an underactuated ship to follow the target path generated by the path-planning module. In this paper, the pure-pursuit algorithm is used to generate the target speed and course for the ASV. As shown in Figure 5, the look-ahead distance L_T is a predefined value, which is related to the ship length. A point on the target path is said to be a target point if the distance between such a point and ship CoG is nearest to L_T. The target point will determine the angle error θ_T, and a PD controller is applied to compute the thruster command based on the speed and course error. By virtue of the calculation of future course error, the pure-pursuit algorithm ensures robust and efficient tracking capability.

2.7. Framework of path-planning algorithm

Given a desired trajectory known as the ‘reference line’, the algorithm only requires the estimated motion in the Cartesian coordinate as an input to compute the output, as shown in Figure 6. The vehicle motion is mapped to the state in the Frenet frame, which enables the state lattice generation. Then we exclude the trajectories exceeding the maximum acceleration of vehicle. It is also undesirable to use the trajectory with high risk of causing a collision, on which the ship will pass close to the static obstacles

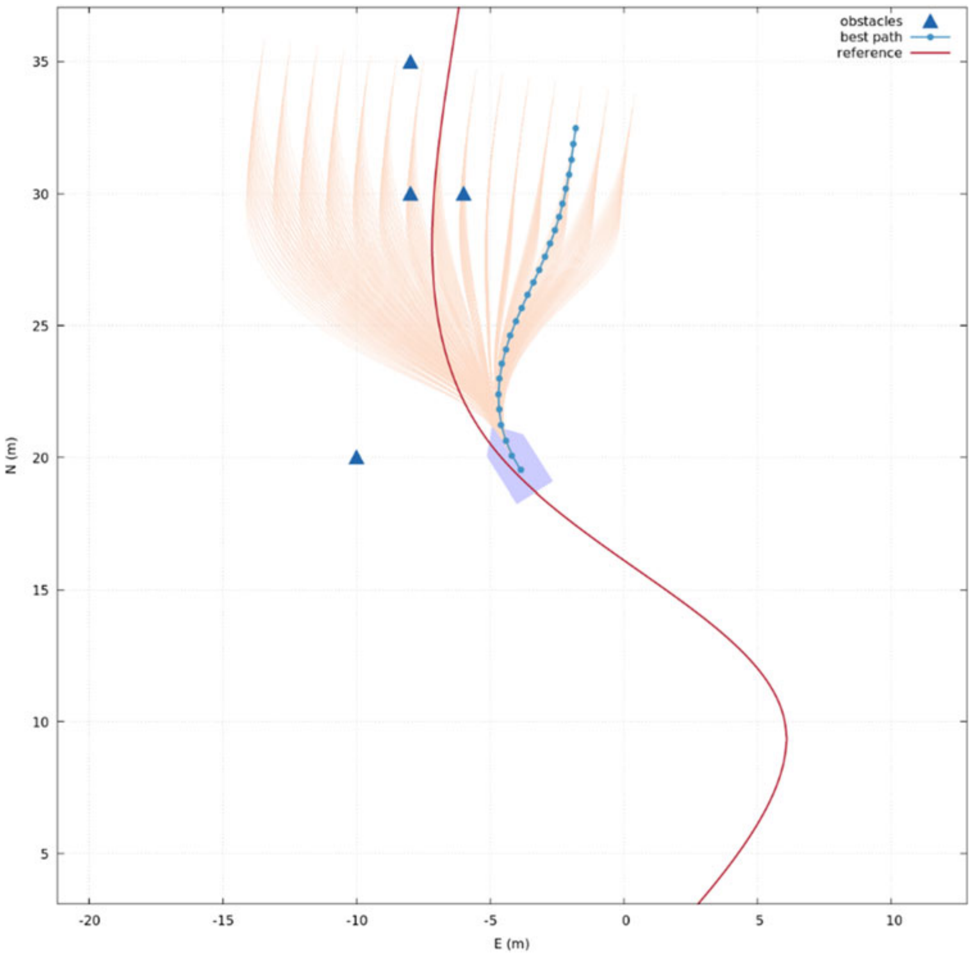

Figure 4. *Optimal trajectory generation: blue triangle indicates static obstacles, red line indicates the reference line, pink lines indicate the state lattices, and blue dotted line indicates the best planning path.*

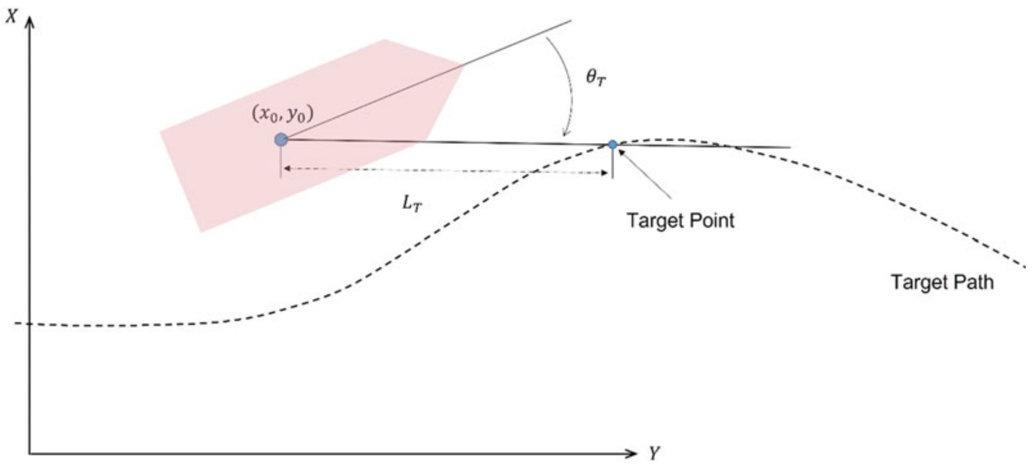

Figure 5. *Pure pursuit algorithm: L_T indicates the look ahead distance, θ_T means the angle error between target and estimated course.*

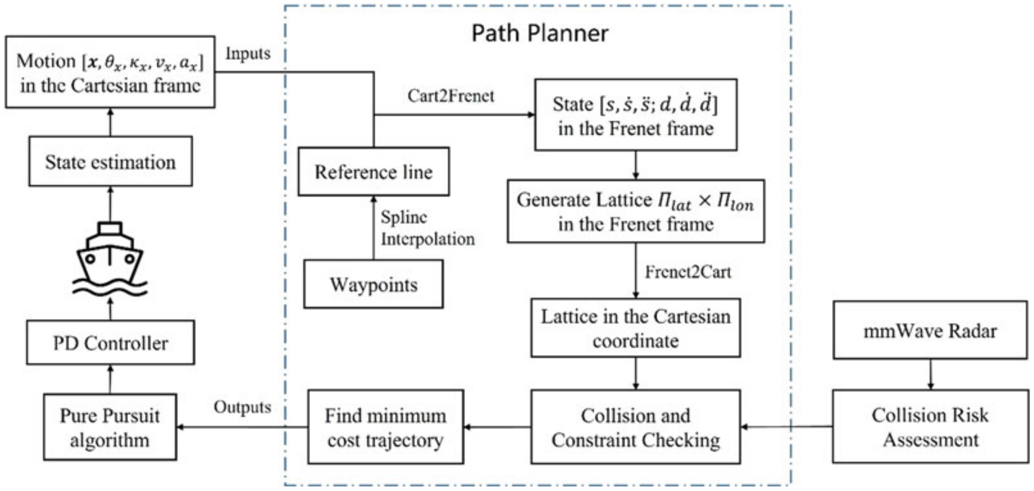
Path Planner
Motion $[x, \theta_x, \kappa_x, v_x, a_x]$ in the Cartesian frame
Inputs
Cart2Frenet
State $[s, \dot{s}, \ddot{s}; d, \dot{d}, \ddot{d}]$ in the Frenet frame
State estimation
Reference line
Generate Lattice $\Pi_{lat} \times \Pi_{lon}$ in the Frenet frame
Spline Interpolation
Waypoints
Frenet2Cart
Lattice in the Cartesian coordinate
mmWave Radar
PD Controller
Outputs
Pure Pursuit algorithm
Find minimum cost trajectory
Collision and Constraint Checking
Collision Risk Assessment

Figure 6. *Algorithm overview. The path-planning algorithm inputs a desired trajectory from route planner or an operator, and outputs the local replanning trajectory for the vehicle's controller ('Cart2Frenet' means the coordinate transform from Cartesian to Frenet frame, and 'Frenet2Cart' means the inversion).*

Table 1. *Vehicle specifications.*

Items	Unit	Value
Length	m	3·10
Width	m	1·60
Weight	kg	244·0
CoG_X	m	0·68
CoG_Y	m	0·00

or CPA of dynamic obstacles. The final step is to find the minimum cost C_{tot} among the collision-free and feasible trajectories. The desired trajectory is fed to the pure-pursuit method and tracking controller.

3. Experimental setup

The experiment was developed on a catamaran (see Figure 7) by integrating the mmWave radar, GNSS sensor, thrusters, wireless modem, batteries, and so on. The vehicle is equipped with a pair of unsteerable propellers, making it underactuated. A 77GHz mmWave radar (Continental ARS404-21) was used for obstacle detection, which enables the real-time radar data transmission via CANBUS. The field of view (FoV) of radar ranges from −60 to 60°. The GNSS sensor provides the positioning, heading and velocity of the vehicle, with a positioning accuracy of 0·5 m and heading accuracy of 0·5 degree. Table 1 lists the vehicle parameters in detail. The target ship (1·8*0·9 m) is equipped with the same GNSS sensor and is remotely controlled by humans during the experiment.

The ASV software was developed on a multi-threaded C++ application by integrating various modules, such as state estimation, feedback control, path planning, and so on. Such a software was implemented on a computer with an ARM 1·5GHz CPU, running Ubuntu 18·04. For the parameters used in path planning algorithm, the step length of reference line is 0·05 m; The lateral offset d_1 ranges from −10 m to 10 m with the common difference 1 m; The planning horizon ΔT varies from 8 s to 10 s with the common difference 0·5 s. Replanning frequency is 5 Hz.

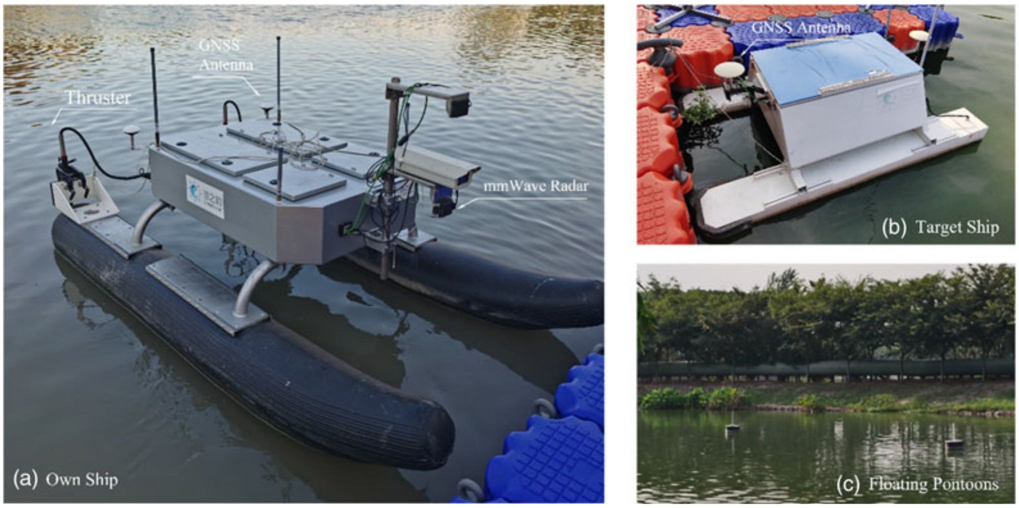

Figure 7. *Sketch of the experimental system, including the own ship (a), the remote-controlled vehicle (b) and the floating pontoons with diameter of 0·9 m (c).*

4. Results

4.1. Pure-pursuit test

The pure-pursuit algorithm looks ahead at the target path and determines the future heading error. For validation of the pure-pursuit algorithm, we present the results of path following tests under calm water. Figure 8 shows the tracking performance (the parameters in PD heading controller Kp = 3 Kd = 300 are set, and the look-ahead distance in the pure-pursuit algorithm is 3 m). This ensures the stability of heading control and allows the vehicle to converge to a geometrical path in an elegant manner.

4.2. Collision-avoidance test with a single static obstacle (Scenario I)

In the first scenario, the own ship was given a user-specified straight-line trajectory with constant speed. When the own ship approached the floating pontoon, the estimated velocity provided by the mmWave radar was small enough that the ASV treated it as a stationary hazard. The proposed path-planning method generated an optimal collision-free path, based on the CPA of the tracked obstacle and the estimated state of the own ship. The controller with pure-pursuit algorithm took the real-time target path and managed to make a smooth and safe manoeuvre (see Figure 9). The reduced speed ensured the manoeuvrability of the own ship (see Figure 10 at time duration from 1447 to 1454 s). As the own ship passed in front of the obstacle, it started to return towards the reference line with a slightly increased target speed. It should be noted that the short-term memory of static obstacles is considered in the collision-risk assessment.

4.3. Collision-avoidance test with two static obstacles (Scenario II)

In Scenario II, the desired trajectory inputted to the path-planning algorithm was a straight-line trajectory, where the distance between two floating pontoons was approximately 35 m. In combination with the pure-pursuit tracking algorithm, the generated trajectory enabled the ship to make a smooth manoeuvre to avoid both floating pontoons. Figure 11 shows the target path at different time instants and planar trajectories of the own ship. It can be seen from Figure 11 that the own ship was approximately 8 m in front of the first pontoon when doing the portside manoeuvre. As no COLREGS constraint was applied, the own ship choose to do the starboard manoeuvre, as the path-planning algorithm defined it to be the

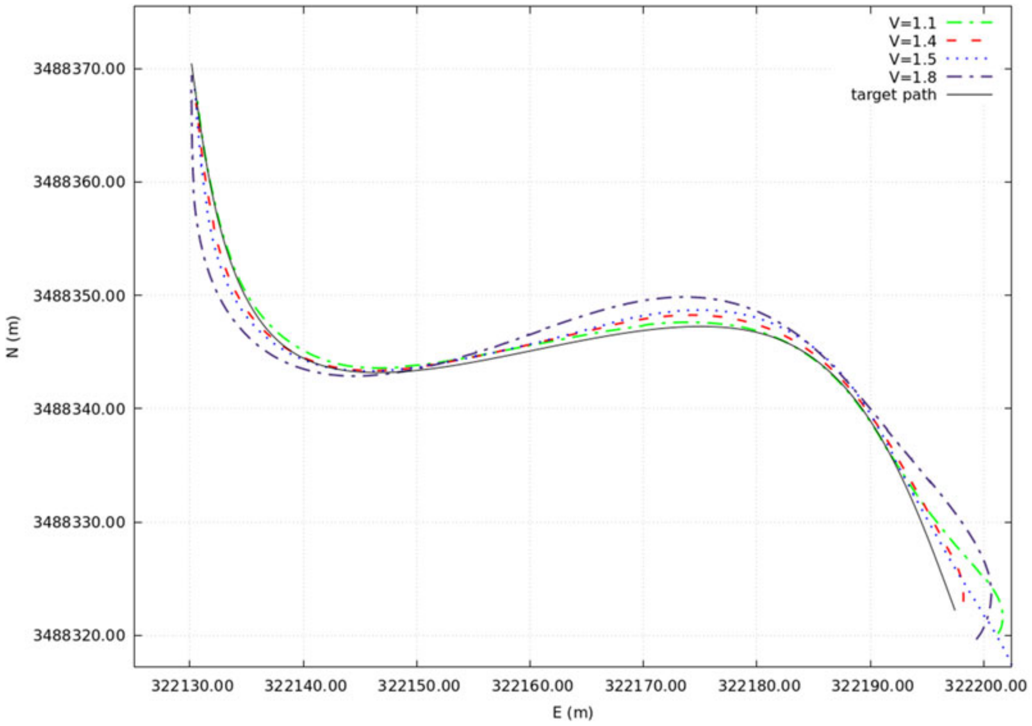

Figure 8. *Planar trajectory for the pure pursuit tests at different target speeds (black solid line indicates the target path). The controller with pure pursuit algorithm was run at a rate of 10 Hz.*

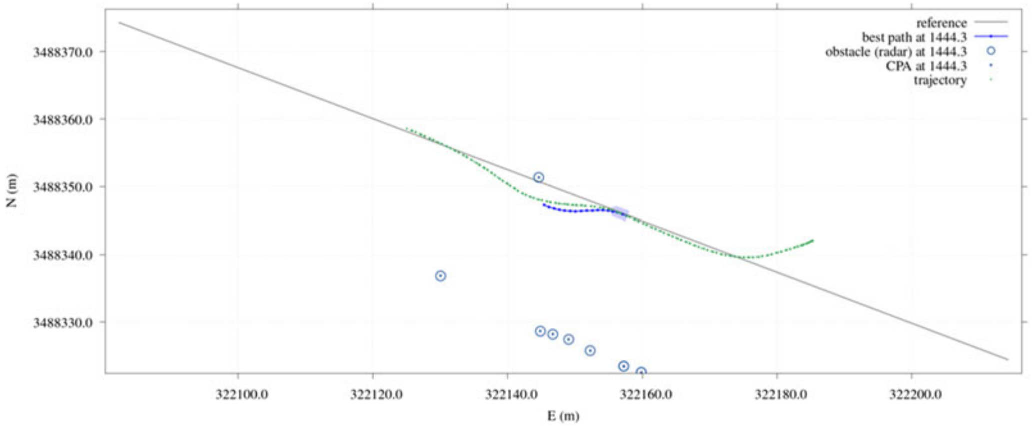

Figure 9. *Planar trajectory in scenario I (grey line indicates reference line, the blue dot-line indicates the optimal target path, and the estimated obstacles are marked with blue circle).*

optimal path. Figure 12 shows that the target speed was slightly reduced when the ship's deviation from the reference line began to increase. Meanwhile, a sufficient course change made it safe for the own ship to travel. As there were no obstacles at 1046 s, the optimal trajectory of the constraint-only problem was chosen, which led the ship to approach the reference line and the desired speed.

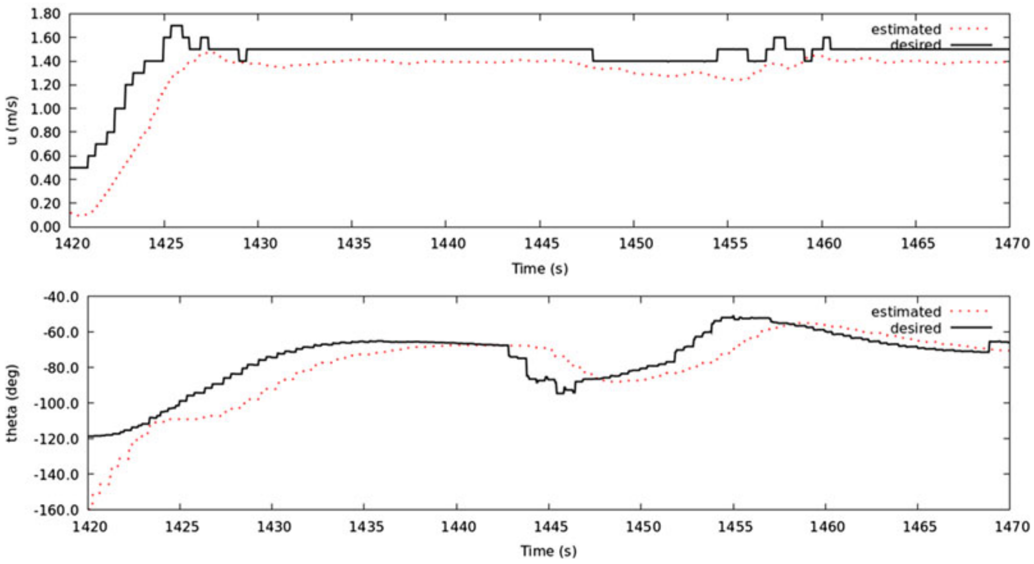

Figure 10. *Time series of estimated (red dot line) and desired (black line) state in Scenario I.*

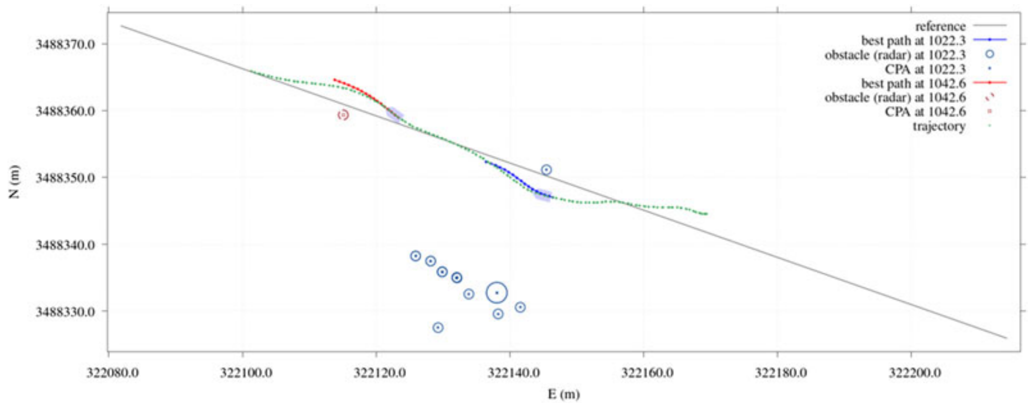

Figure 11. *Planar trajectory in the second scenario (grey line indicates reference line, the blue and red dot-lines indicate the optimal target path at time 1022·3 s and 1042·6 s, the green dots denote the trajectory of own ship and the estimated obstacles are marked with blue circle).*

4.4. Collision-avoidance test with crossing-from-portside obstacles (Scenario III)

Three scenarios were considered for the moving obstacles: crossing from portside (Scenario III), crossing from starboard (Scenario IV) and head-on (Scenario V). In each scenario, the speed and course of the moving obstacle remained constant, which required the ASV to preform collision check and turning manoeuvres. We assigned each obstacle with a unique ID. Using this ID, we introduced a short-term memory of trajectory and CPA of moving obstacle. If such a memory was ignored, the own ship might oscillate along the reference line and result in an undesirable behaviour. This is because the uncertainty in the radar detection could make the CPA prediction portside and starboard frequently, which drastically changes the lateral offset of target path.

As no navigation rule was followed by the given path-planning algorithm, the own ship performed a portside manoeuvre when the obstacle started to approach from portside (see Figure 13). This can be considered to be optimal as the own ship choose to pass behind the moving obstacle. The GNSS-estimated trajectory shows that the moving obstacle was out of the FOV of radar at 67·4 s. This means

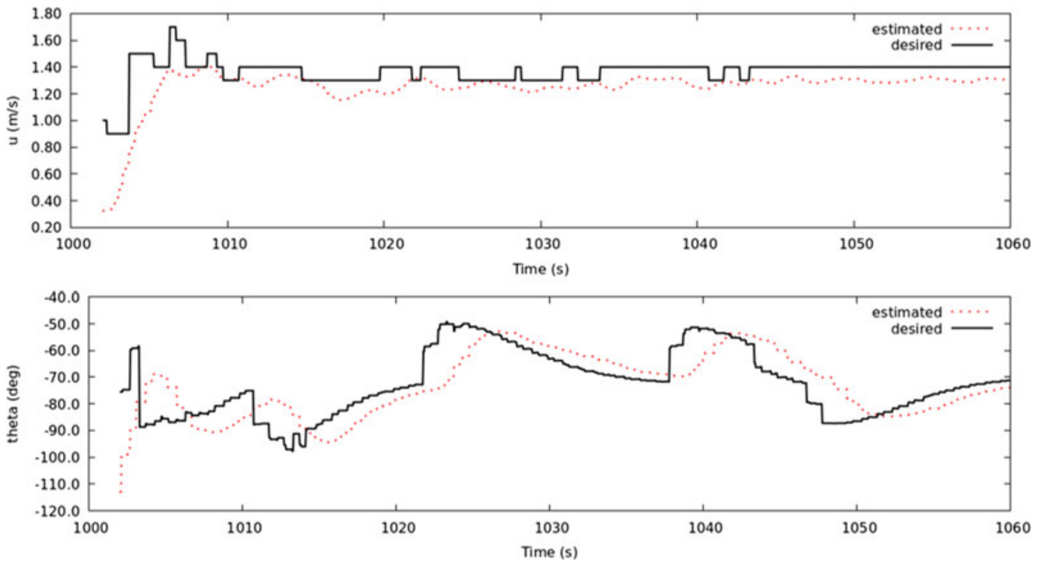

Figure 12. *Time series of estimated and desired state in Scenario II.*

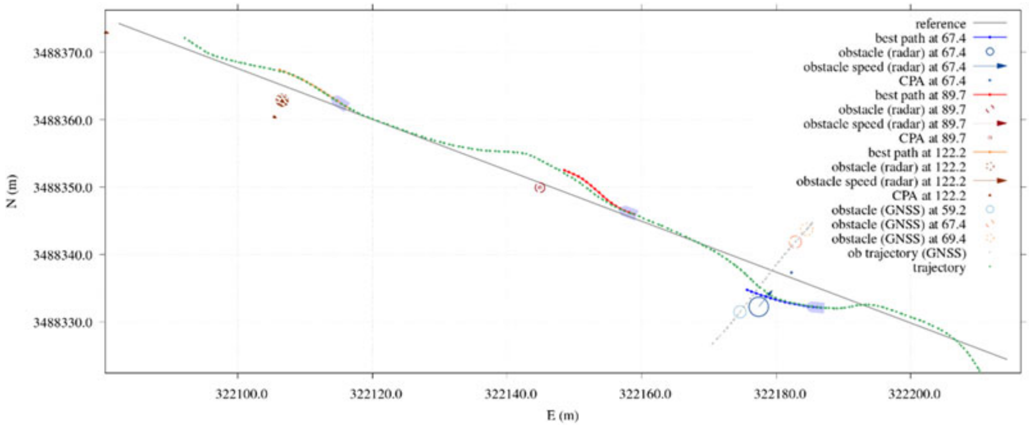

Figure 13. *Planar trajectory in Scenario III. The GNSS-estimated and radar-estimated positions of obstacles are marked with circles. The obstacle velocity measured by radar are marked with arrows. Best path indicates the optimal path generated by real-time path planning algorithm.*

that we retained the memory of radar-based position and CPA of the moving obstacle, which lasted about 5 s. Such a memory might result in a large clearance to the moving obstacle. The course change manoeuvre was accompanied by the reduced target speed (see Figure 14). After the avoidance manoeuvre was finished, the own ship passed the two floating pontoons with sufficient clearance.

4.5. Collision-avoidance test with crossing-from-starboard obstacles (Scenario IV)

Figures 15 and 16 show the successful avoidance in Scenario IV, crossing-from-starboard obstacle. At the time duration from 520 s to 530 s, the own ship keeps a constant course and identifies on which side the moving obstacle would pass. The memory that the moving hazard approached from the starboard lasted for a short interval, which forced the own ship to alter its course to starboard and pass abaft of the moving obstacle. At time 538·3 s, the diameter of radar-estimated obstacle (about 7·0 m) was much

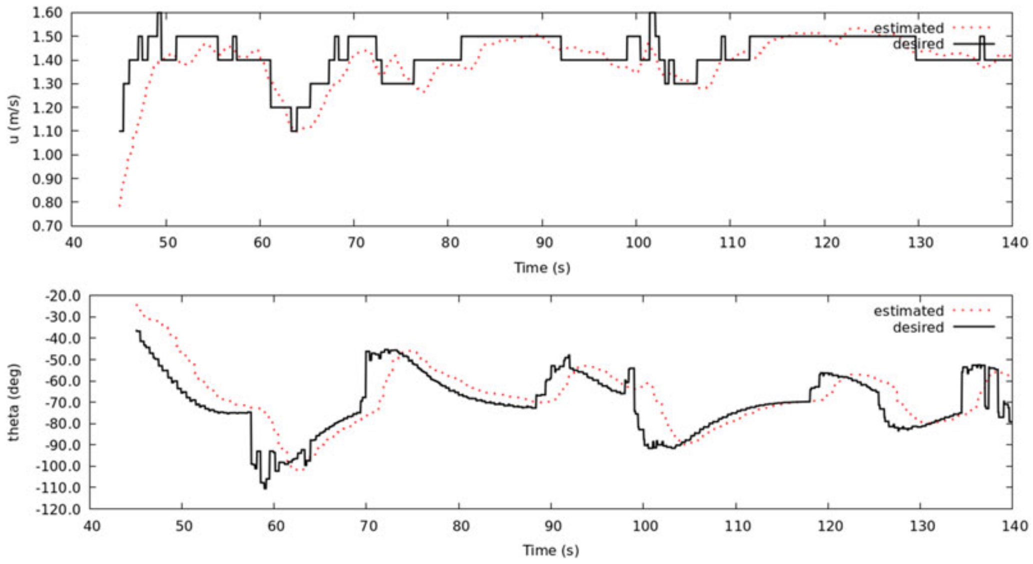

Figure 14. *Time series of estimated and desired state in Scenario III.*

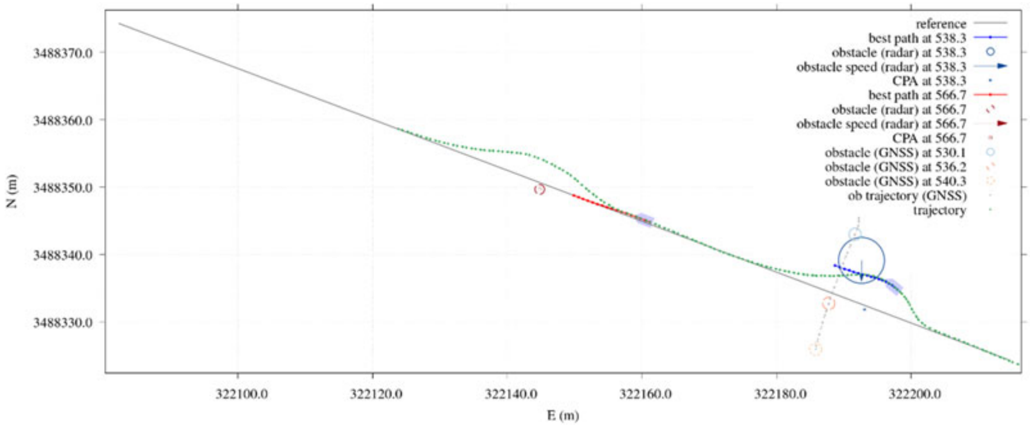

Figure 15. *Planar trajectory in Scenario IV. The GNSS-estimated and radar-estimated positions of obstacles are marked with circles. The obstacle velocity measured by radar are marked with arrows. Best path indicates the optimal path generated by real-time path planning algorithm.*

larger than the real value (2·0 m), which might have caused increased obstacle clearance. There was an abrupt change in the desired heading (see Figure 16) at time 537 s, when the path generation was affected by noise on obstacle detection. However, the proposed path-planning algorithm made the own ship travel smoothly and showed good robustness to noise. At time 566·7 s, the target path was aligned with the reference line, due to the low risk in collision. As the own ship approached the static pontoon, it performed the starboard manoeuvre to reduce the collision risk.

4.6. Collision-avoidance test with head-on obstacles (Scenario V)

In the head-on scenario (Scenario V), the target ship travelled with a constant speed. As the target ship took no action to reduce the collision risk, the own ship had to perform a starboard manoeuvre to avoid the obstacle approaching from the front (see Figure 17). The increase in the rate of turn of the own

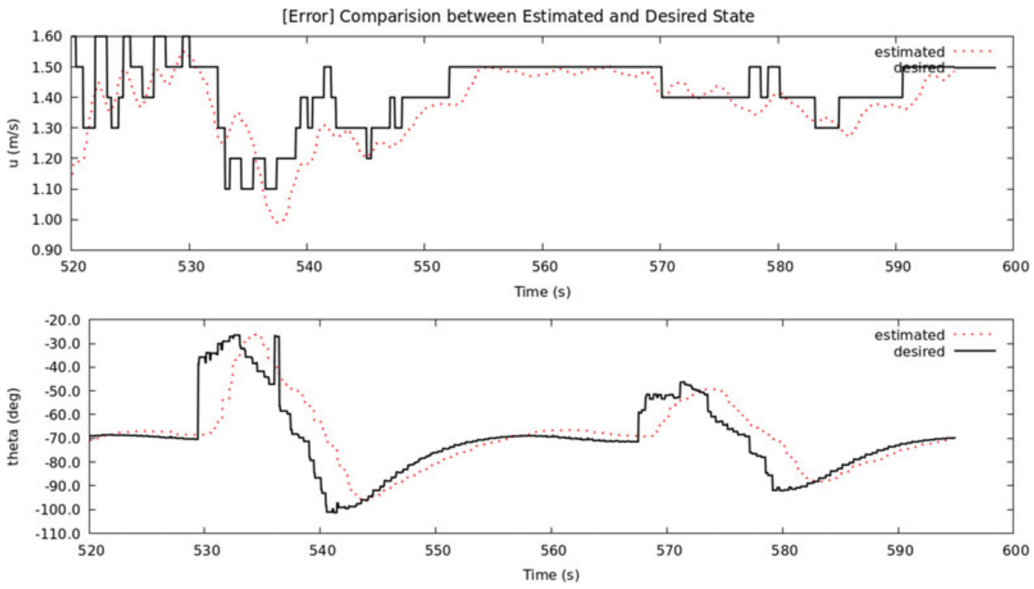

Figure 16. *Time series of estimated and desired state in Scenario IV.*

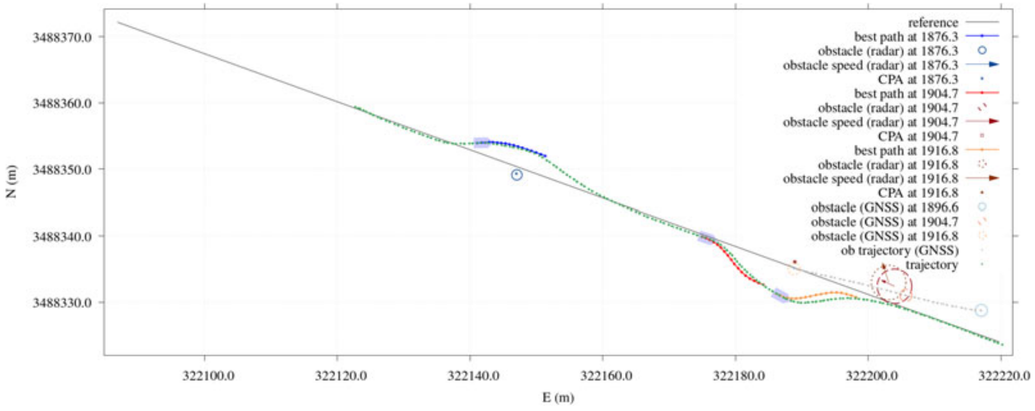

Figure 17. *Planar trajectory in Scenario V. The GNSS-estimated and radar-estimated positions of obstacles are marked with circles. The obstacle velocity measured by radar are marked with arrows.*

ship might have affected the state estimation errors of obstacles. Therefore, we introduced the shorter-term memory for these imprecise tracked obstacles. This means that the given algorithm might have reduced the uncertainty in the situational awareness and increased robustness to noise with respect to radar detection. With the successful collision checking, the own ship managed to avoid the CPA of the moving obstacle. Figure 18 illustrates that the pure-pursuit and feedback controller could allow the ASV to follow the desired speed and heading in a smooth way.

4.7. Repeatability test and failure analysis

More than 100 cases were performed on each scenario. Figure 19 shows the lowest, highest and mean values of the minimum distance between the own ship and obstacles during the collision avoidance test of each scenario. The success rate of collision avoidance of each scenario is also illustrated in Figure 19. It demonstrates that the success rate of Scenario I is 100% when the initial distance between the own

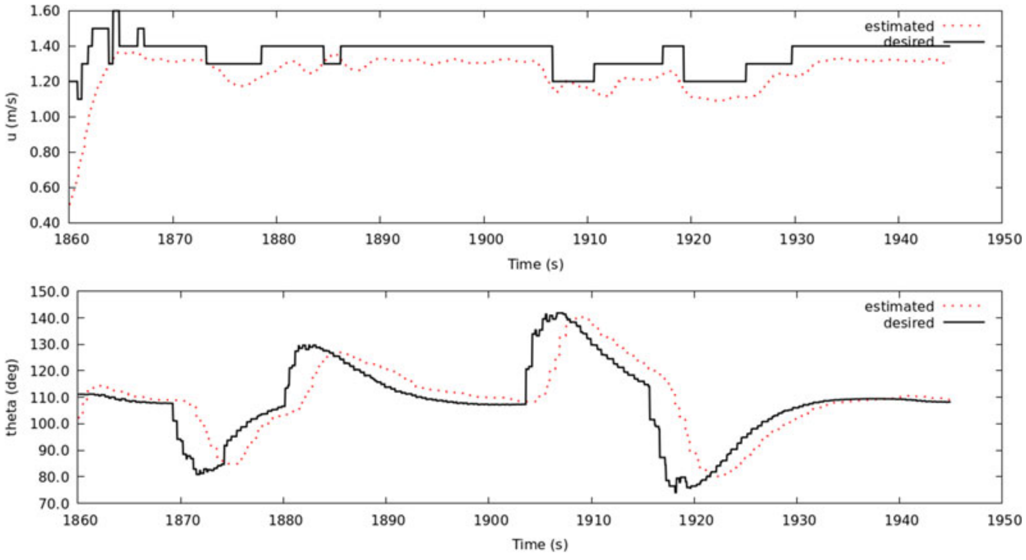

Figure 18. *Time series of estimated and desired state in Scenario V.*

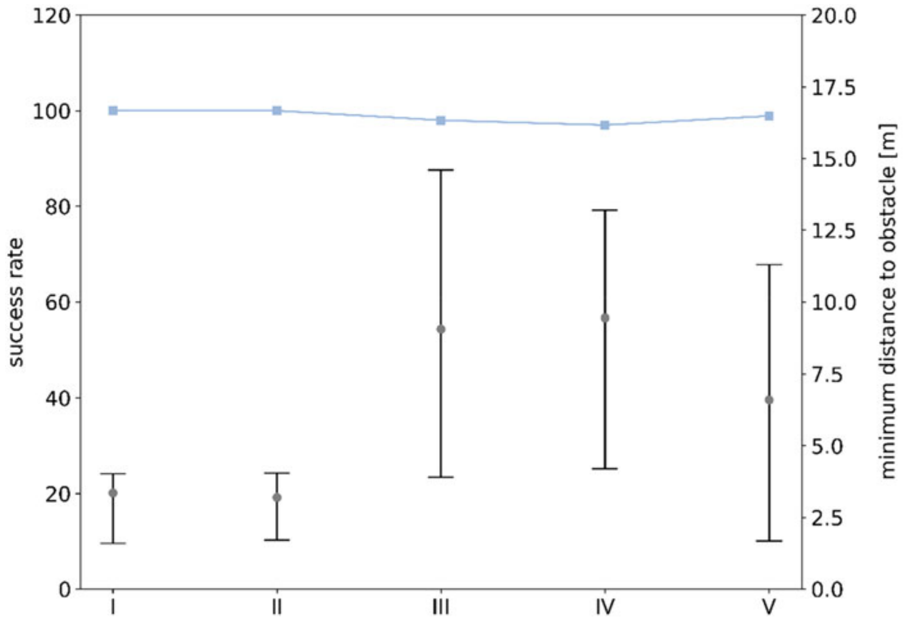

Figure 19. *The success rate of collision avoidance and minimum distance between own ship and obstacle in each scenario.*

ship and pontoon is larger than 15 m. In Scenario II, none of failure test is observed. However, due to the limit of measurement accuracy of mmWave radar, the static floating pontoons will be classified as moving targets at some moments. It is difficult to determine the portside or starboard manoeuvre, which will increase the risk of collision.

The success rate of Scenarios III, IV and V is about 98%, 97% and 99%, respectively. The reason of the failure in collision avoidance of Scenario III is due to the fact that the moving target is classified as a static one at first, allowing the own ship to perform starboard manoeuvre; however, as the own ship approaches the moving target, such a target is classified as crossing-from-portside one, leading to portside manoeuvre of the own ship; the inconsistency in the collision avoidance manoeuvre increases the risk of collision. For the same reason, it fails to avoid a collision in Scenarios IV and V.

As shown in Figure 19, the minimum distance in scenario V is smaller than the other scenarios. The reason is that the minimum distance between the own ship and obstacles are mainly determined by the lateral distance along the reference line. In Scenarios III and IV, the relative lateral speed between the own ship and moving target is large enough to maintain a safe distance. Meanwhile, the lateral speed of moving target is very small, leading to the smaller relative lateral speed and distance in Scenario V.

5. Conclusion

The contribution of this paper is a radar-based trajectory-generation algorithm which allows for collision avoidance. Such a method can easily accommodate the passenger comfort, physical constraints and manoeuvre smoothness. The experimental results demonstrate that it realises the short-term objectives, such as velocity keeping and collision avoidance. Future work includes combining GNSS with the inertial measurement unit to enable high-frequency and high-quality measurements on the motion of the own ship and obstacles. This might achieve the increased robustness of collision avoidance, especially for the dynamic obstacles. Extensive experimental study should be conducted on the effect of parameters on the performance of the given path-planning algorithm, which will develop an adaptive path-planning method.

Acknowledgements. This paper is partly supported by the National Science Foundation of China (61473183, U1509211, 61627810), and National Key R&D Program of China (2017, YFE0128500).

References

Bishop, G. and Welch, G. (2001). An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, **8**(27599–23175), 41.

Blaich, M., Rosenfelder, M., Schuster, M., Bittel, O. and Reuter, J. (2012a). Fast Grid Based Collision Avoidance for Vessels Using A_* Search Algorithm. *Presented at the 2012 17th International Conference on Methods & Models in Automation & Robotics (MMAR)*, IEEE, pp. 385–390.

Blaich, M., Rosenfelder, M., Schuster, M., Bittel, O. and Reuter, J. (2012b). Extended grid based collision avoidance considering COLREGs for vessels. *IFAC Proceedings Volumes*, **45**(27), 416–421.

Cockcroft, A. N. and Lameijer, J. N. F. (2003). *Guide to the Collision Avoidance Rules*. Oxford: Elsevier.

Dolgov, D., Thrun, S., Montemerlo, M. and Diebel, J. (2008). Practical search techniques in path planning for autonomous driving. *Ann Arbor*, **1001**(48105), 18–80.

Eriksen, B. H., Breivik, M., Wilthil, E. F., Flåten, A. L. and Brekke, E. F. (2019). The branching-course model predictive control algorithm for maritime collision avoidance. *Journal of Field Robotics*, **36**(7), 1222–1249.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Chichester: John Wiley & Sons.

Frenet, F. (1852). Sur les courbes à double courbure. *Journal de mathématiques pures et appliquées*, **17**, 437–447.

González, D., Pérez, J., Milanés, V. and Nashashibi, F. (2015). A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, **17**(4), 1135–1145.

Howard, T. M. and Kelly, A. (2007). Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, **26**(2), 141–166.

Kühnel, W. (2015). *Differential Geometry*, vol. **77**. Stuttgart: American Mathematical Soc.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D. and Huntsberger, T. L. (2013). Safe maritime autonomous navigation with COLREGS, using velocity obstacles. *IEEE Journal of Oceanic Engineering*, **39**(1), 110–119.

Paden, B., Čáp, M., Yong, S. Z., Yershov, D. and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, **1**(1), 33–55.

Reif, J. and Wang, H. (1998). The Complexity of the two Dimensional Curvature-Constrained Shortest-Path Problem. *Presented at the Third International Workshop on Algorithmic Foundations of Robotics*. pp. 49–57.

Schuster, M., Blaich, M. and Reuter, J. (2014). Collision avoidance for vessels using a low-cost radar sensor. *IFAC Proceedings Volumes*, **47**(3), 9673–9678.

Takahashi, O. and Schilling, R. J. (1989). Motion planning in a plane using generalized Voronoi diagrams. *IEEE Transactions on Robotics and Automation*, **5**(2), 143–150.

Takahashi, A., Hongo, T., Ninomiya, Y. and Sugimoto, G. (1989). Local Path Planning and Motion Control for AGV in Positioning. *Presented at the Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems'.(IROS'89)'The Autonomous Mobile Robots and its Applications*, IEEE, pp. 392–397.

Werling, M., Ziegler, J., Kammel, S. and Thrun, S. (2010). Optimal Trajectory Generation for Dynamic Street Scenarios in A Frenet Frame. *Presented at the 2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 987–993.

Wilthil, E. F., Flåten, A. L., Brekke, E. F. and Breivik, M. (2018). Radar-based Maritime Collision Avoidance Using Dynamic Window. *Presented at the 2018 IEEE Aerospace Conference*, IEEE, pp. 1–9.

Cite this article: Hu Z, Yang Z, Liu X, Zhang W (2023). Radar-based path planning of autonomous surface vehicle with static and dynamic obstacles in a Frenet Frame. *The Journal of Navigation* **76**: 4–5, 487–505. https://doi.org/10.1017/S037346332300019X