robotics

MDPI

Article

Research on Trajectory Tracking and Obstacle Avoidance of Nonholonomic Mobile Robots in a Dynamic Environment

Kai Zhang, Ruizhen Gao * and Jingjun Zhang

College of Mechanical and Equipment Engineering, Hebei University of Engineering, Handan 056000, China; zhangkaideemail521@163.com (K.Z.); zhangjingjun@hebeu.edu.cn (J.Z.)

* Correspondence: gaoruizhen@hebeu.edu.cn

Received: 26 July 2020; Accepted: 3 September 2020; Published: 18 September 2020

Abstract: This paper presents an obstacle-avoidance trajectory tracking method based on a nonlinear model prediction, with a dynamic environment considered in the trajectory tracking of nonholonomic mobile robots for obstacle avoidance. In this method, collision avoidance is embedded into the trajectory tracking control problem as a nonlinear constraint of the position state, which changes with time to solve the obstacle-avoidance problem in dynamic environments. The CasADi toolkit was used in MATLAB to generate a real-time, efficient C++ code with inequality constraints to avoid collisions. Trajectory tracking and obstacle avoidance in dynamic and static environments are trialed using MATLAB and CasADi simulations, and the effectiveness of the proposed control algorithm is verified.

Keywords: nonholonomic mobile robot; model predictive control; trajectory tracking; dynamic obstacle avoidance; CasADi

1. Introduction

With the continuous development of robot technology, various robots are being increasingly used in complex working environments. In unknown working environments, mobile robots should be able to sense the surrounding environment, avoid obstacles, and have the ability to automatically navigate, with obstacles divided into dynamic obstacles and static obstacles.

Previous obstacle avoidance control methods based on nonholonomic robots were proposed, with the purpose of these methods being to enable mobile robots to avoid obstacles and utilize faster and better trajectory tracking. The proposed path-planning algorithms described in the literature [1–4] aim toward the static environment and do not consider the dynamic environment. Where the dynamic environment is considered [5], applications of this research demonstrate poor real-time performance by using support vector machines for collision avoidance problems; therefore, a need emerged for the development of a new algorithm.

Other collision avoidance algorithms exist, including fuzzy control [6–9], but the simple fuzzy processing of information leads to a reduction in the control accuracy of the system and deterioration of dynamic quality, with the design of fuzzy control lacking systematicity and unable to define the control goal. There is also the artificial potential field method [10–13], but this has a trap area and a path cannot be identified in a group of similar obstacles. It oscillates before encountering an obstacle, the robot swings in the narrow passage, and the target near the obstacle is unreachable. Another algorithm is the sliding mode control [14–17], but when the state trajectory reaches the sliding mode surface, it is difficult to strictly slide along the surface to the equilibrium point. Instead, it travels back and forth on both sides to approach the equilibrium point, resulting in oscillation.

Model predictive control [18–21] is one of the most commonly used advanced control technologies in the industry. Due to the use of discrete convolutions and models described by nonminimization, there is a large amount of information redundancy, which is conducive to improving the robustness of the system. A rolling optimization strategy can be adopted, whereby the optimization calculation is repeated online and implemented on a rolling basis, so that the uncertainties caused by model mismatches and external disturbances are compensated in time to obtain better dynamic control performance. In the literature [22,23], the motion equation of this robot utilized a linearized version of model predictive control in the trajectory tracking problem. In another work [24], an explicit nonlinear motion model of a robot was used and a nonlinear model predictive control was used for tracking problems. Ostafew et al. [25] proposed a learning-based, nonlinear, model predictive control algorithm to reduce the path-tracking error of repeated traversals along the reference path. Mehrez et al. [26] proposed an open-source toolkit to implement point stabilization and trajectory tracking of nonholonomic mobile robots. The toolkit and the software of the wheeled mobile robot research platform were combined using C++ code to verify the applicability of the control issues. Heonyoung et al. [27] proposed an obstacle-avoidance strategy based on model predictive control and optimized the optimization problem using gradient descent method, using an algorithm able to verify the tracking ability and obstacle-avoidance ability of the mobile robot but that does not consider the presence of a dynamic environment. Park et al. [28] proposed an obstacle-avoidance scheme for autonomous vehicles, where a separate controller was designed to track the generated trajectory using a parallax-based method to incorporate local obstacle information into a performance index, resulting in verification of the feasibility of the scheme. Jie et al. [29] proposed a collision-avoidance method that constructed the 3D, virtual, dangerous potential field as the superposition of the trigonometric function of the road and the exponential function of the obstacle; these authors formulated the tracking task as a multiconstraint model predictive control problem and proved the effectiveness of the scheme.

The main contributions of this paper are the use of a discrete-time system model based on MPC to achieve trajectory tracking control and to avoid dynamic obstacle collisions. Firstly, cost function is composed by minimizing state error and control input. Secondly, collision avoidance is embedded into the trajectory tracking control problem as a nonlinear constraint of the position state, which changes with time to solve the obstacle-avoidance problem in dynamic environments. Finally, the effectiveness of the proposed control method is verified by simulation.

This paper is organized as follows. Section 2 describes the dynamic model of the two-wheeled differential robot. Section 3 briefly introduces the model predictive control method. Based on this method, the controller is designed and the rules for dynamic obstacle avoidance are set. The cost function is optimized using the direct, single-shooting method. Section 4 is simulated with MATLAB and the code is optimized using the CasADi toolkit. Section 5 concludes the paper.

2. Materials and Methods

2.1. Two-Wheeled Differential Drive Robot

The mechanical structure of the two-wheeled differential robot is shown in Figure 1.

A two-wheeled differential robot is a typical nonlinear system. It is easy to turn and has a small turning radius. Both wheels are on the same common axis. The speed of each wheel is controlled by a separate motor. The speed of the left wheel is defined as v_l and the angular velocity is defined as w_l. The speed of the right wheel is defined as v_r and the angular velocity is defined as w_r. Assuming that the robot is purely rolling without sliding, the relationship is as follows:

$$v_l = \mathrm{r}w_l, \quad v_r = \mathrm{r}w_r \tag{1}$$

where r is the radius of the two-wheeled differential robot wheel.

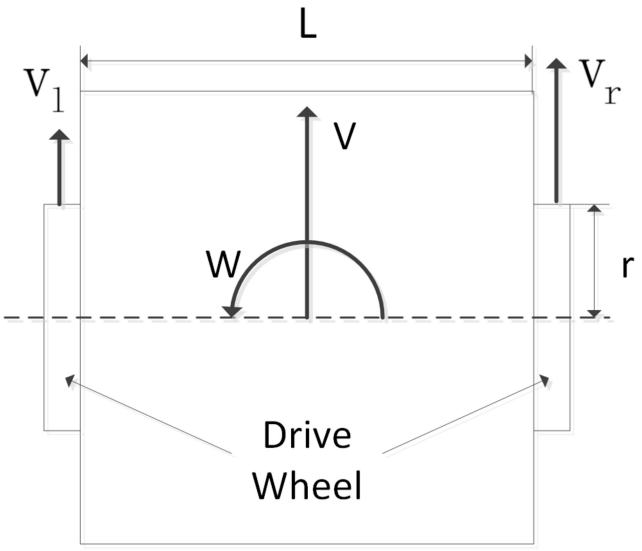


Figure 1. Top view of a typical two-wheeled differential robot.

The speed and angular velocity of the two-wheeled differential robot can be calculated from the speed and angular velocity of the two wheels, as shown in the following Formula (2):

$$v = \frac{v_l + v_r}{2} = \mathrm{r}\frac{w_l + w_r}{2}, \mathrm{w} = \frac{v_r - v_l}{L} = \mathrm{r}\frac{w_r - w_l}{L} \tag{2}$$

where L is the wheelbase of the two-wheeled differential robot, and the relationship between the control input u and the linear and angular velocities of the two drive wheels is as follows:

$$\mathrm{u} = \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \frac{v_l + v_r}{2} \\ \frac{v_r - v_l}{L} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} w_r \\ w_l \end{bmatrix} \tag{3}$$

The turning radius R of the two-wheeled differential robot can be calculated as follows:

$$\mathrm{R} = \frac{1}{2}\frac{(w_l + w_r)}{(w_r - w_l)} \tag{4}$$

When $w_l = w_r$, the turning radius is infinite, which means that the robot moves in a straight line.

2.2. Kinematics of Mobile Robots

As shown in Figure 2, the state vector of the two-wheeled differential robot is defined as

$$\mathbf{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5}$$

where x and y are the center positions of the current wheel axis of the robot and θ is the orientation of the robot in the current two-dimensional space (global coordinate frame). The two-wheeled differential robot has no lateral slip, nonlinear constraints, as follows:

$$\dot{x}\cos\,\theta - \dot{y}\sin\,\theta = 0 \tag{6}$$

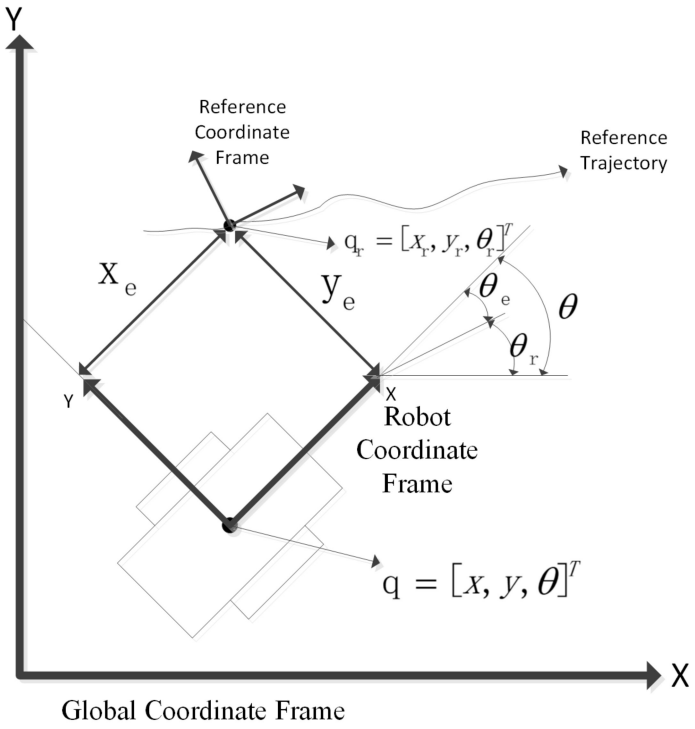


Figure 2. Global coordinate frame, robot coordinate frame, and reference coordinate frame of the mobile robot.

The motion model (7) of the mobile robot can be derived from Formulae (1) to (6). This equation shows the relationship between the state of the robot and its own linear and angular velocity [30].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{7}$$

Discretizing the above formula gives a discrete-time formula, i.e.,

$$x_{k+1} = x_k + \Delta T \cos\theta_k v_k y_{k+1} = y_k + \Delta T \sin\theta_k v_k \theta_{k+1} = \theta_k + \Delta T w_k \tag{8}$$

3. Controller Design

3.1. Nonlinear Model Control Prediction

In the case of discrete time, the dynamic model can be defined by the nonlinear state time, i.e.,

$$x\,(k+1) = \mathrm{f}(x(k), u(k)) \tag{9}$$

where $\mathrm{f}(x(k), u(k)) \in R^n$ is a nonlinear continuous function, $x(k) \in R^n$ is the system state vector, and $u(k) \in R^m$ is the control input vector of the system. Generally speaking, the MPC scheme involves using a nominal model to predict the future state within a limited range of predictions for the system and the last available measurement or accurate estimation of the state to obtain the best control action within the control range (control vector). The control range is less than or equal to the prediction range. This control vector guides the system state to follow the time-varying reference generated by the model of the form (7). Then, the first element of the control vector is applied and the entire process is repeated with the next sample.

When the research object is a planar, nonholonomic, two-wheeled mobile robot, the cost function for the state error and control input is as follows:

$$J_N(x, \mathrm{u}) = \sum_{k=0}^{N-1} e(x(k), u(k)), \tag{10}$$

where $e(x(k), u(k)) = (\mathrm{q} - q_r)^T\ \mathrm{Q}\ (q_c - q_r) + (u - u_r)^T\ \mathrm{R}\ (u - u_r)$ represents the running cost, q_r=$[x_r\ y_r\ \theta_r]^T$ is the robot reference pose, $u_r = [v_r\ w_r]^T$ is the reference control input, $\mathrm{Q} \in R^{3\times3}$ and $\mathrm{R} \in R^{2\times2}$ are the state and control input positive definite weight matrices, respectively, and N is the number of prediction steps. Therefore, at time t, the real-time open-loop optimization problem of the nonlinear model predictive controller can be summarized as follows:

$$\min_u J_N(x_0, \mathrm{u}) = \sum_{k=0}^{N-1} e(x(k), u(k)) \tag{11}$$

subject to $x(k + 1) = \mathrm{f}(x(k), u(k))$,

$x(0) = x_0$,
$u(k) \in$ U,$\forall$k $\in$ [0, N−1], and
$x(k) \in$ X,$\forall$k $\in$ [0, N],

where U is the control space set and X is the state space set.

3.2. Obstacle Avoidance

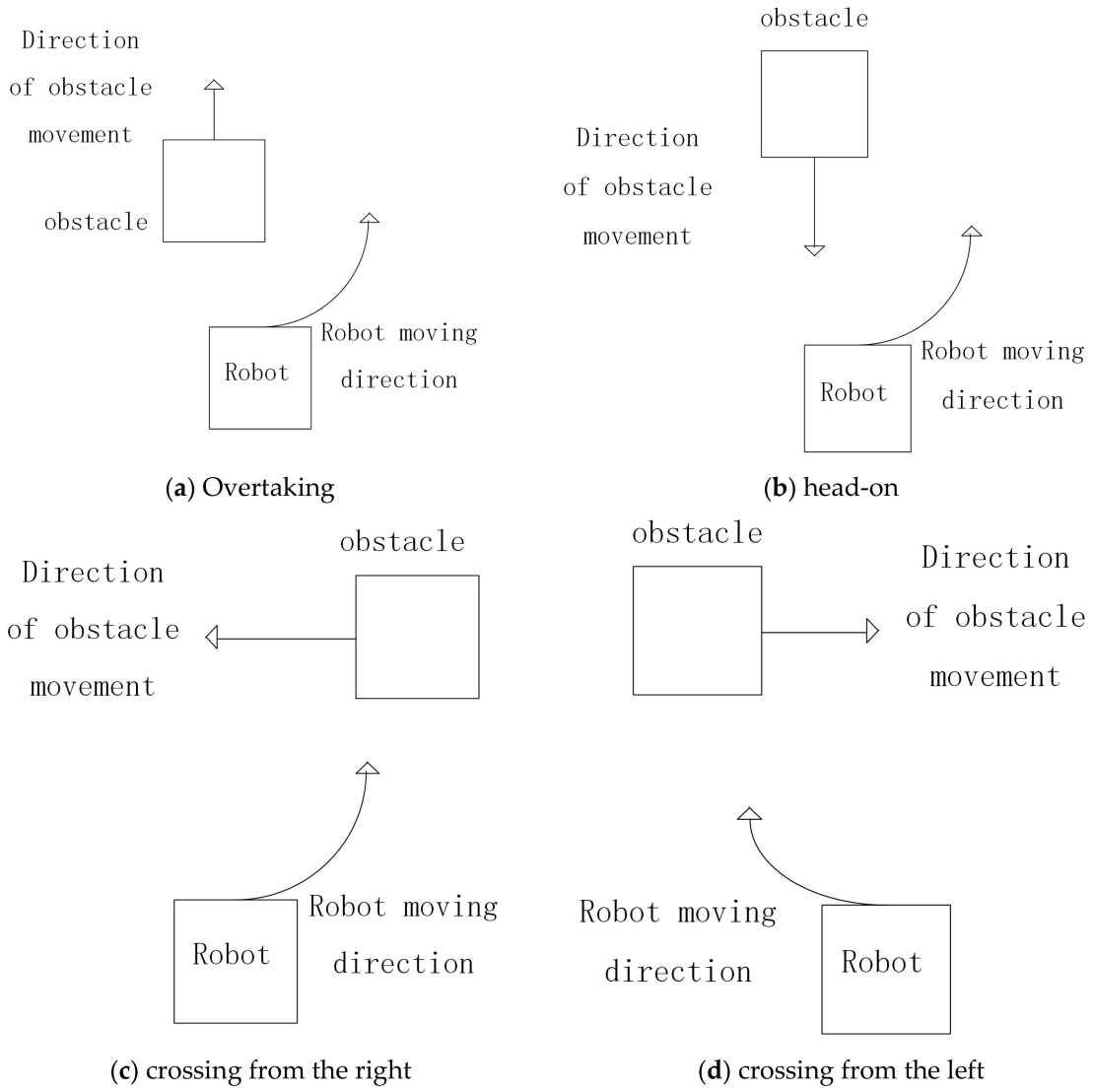
In general, the collision avoidance problem is treated as a planning problem independent of the controller. Due to the unfavorable ground environment, the controller may not be able to solve the problem. In order to overcome this problem, the characteristics of the nonlinear model can be used to predict the control and integrate state constraints. Assuming the obstacle is a square shape, the collision constraint is transformed into a state constraint and integrated into the controller design in the form

$$(x(t) - x_o(t))^2 + (y(t) - y_o(t))^2 \geq (r_r + r_o + r)^2 \tag{12}$$

where $x(t)$ and $y(t)$ are the predicted position of the robot in the prediction range, $x_o(t)$ and $y_o(t)$ are the centers of the obstacle in the prediction range, r_r is the safety distance from the edge of the robot to the edge of the obstacle, r_o is the radius of the circular envelop for the square obstacle, and r is the radius of the robot.

When the obstacle is of arbitrary shape, a circle can be created to enclose the obstacle. The center position of the obstacle can be determined by this circle and the center position coordinates of the obstacle can be obtained.

Several situations when the mobile robot avoids obstacles are shown in Figure 3.



(**a**) Overtaking (**b**) head-on

(**c**) crossing from the right (**d**) crossing from the left

Figure 3. Schematic diagram of obstacle avoidance for nonholonomic robots.

3.3. Direct, Single-Shooting Method

In order to solve the optimal control problem of the nonlinear model prediction, a direct, single-shooting method was adopted to convert the optimal control problem into a nonlinear programming problem. The algorithm originated from solving the boundary value problem of differential equations, with the basic idea of this algorithm being based on the Runge–Kutta method. The time interval is uniformly dispersed into smaller N intervals, then the initial value problem is solved in each small interval and added to the solution process. Some matching conditions then form a solution over the entire range. All Runge–Kutta variants include first-order and second-order differential algorithms. In order to make the initial state of the state trajectory and the sensitivity of the control input more convenient to calculate, after conversion to algebraic differential equations, calculations are performed using other integrators. Then, the problem constraint is estimated at each time node and the control vector parameters are changed to piecewise constants. The result of this method is a least-squares, nonlinear programming with a fixed dimension, which can be solved using the Gauss–Newton method.

Based on the above, the nonlinear model prediction scheme is as follows:

1. Set the time parameter t, the number of prediction steps N, the time interval T, and the weight matrices Q and R;
2. Get the initial state x_0;

3. Obtain the obstacle poses x_o and y_o;
4. Solve the optimization problem (11) and get the optimal input control vector and prediction state;
5. Wait for the next time interval and set the next time parameter, then repeat from Step 2.

3.4. CasADi Toolkit

CasADi is an open-source software toolkit for numerical optimization written in C ++. It is a general tool with great flexibility and can be used to model and solve numerical optimization problems. Regarding the problem of differential equation constraints in particular, that is, the optimal control problem, compared with other similar frameworks, CasADi can be more easily increased to a higher dimension, thereby increasing the richness of the differentiable operation set. CasADi realizes the forward and reverse modes of automatic differentiation (ad); therefore, it is very suitable for nonlinear model predictive control programming. The following experiments use the basic principles of CasADi.

4. Results

This experiment validates the proposed simulation based on model predictive control to avoid obstacles. The obstacle-avoidance simulation of static and dynamic obstacles with three different trajectories using the same prediction steps is utilized by a differential mobile robot. The three trajectories are circular, figure-of-eight and linear. The simulation uses MATLAB2017b version and the CasADi toolkit.

When the obstacle is a square with a side length of 0.3 m and the constraint is set from the center point of the obstacle to the vertex of the square as a circle, then

$$x_o - r_o - r_r < \mathrm{x} < x_o + r_o + r_r \; y_o - r_o - r_r < \mathrm{y} < y_o + r_o + r_r \tag{13}$$

Therefore, the control input constraints of the robot are

$$-v_{max} \le v \le v_{max} \; -w_{max} \le w \le w_{max} \tag{14}$$

4.1. Circular Trajectory

When the starting position of the robot is set to [0.5; 0.5; 0], the reference trajectory is as follows:

$$\begin{aligned} \mathrm{xr} &= 1 + 2 * \sin(0.5 * \mathrm{t}), \\ \mathrm{yr} &= -1 + 2 * \cos(0.5 * \mathrm{t}). \end{aligned} \tag{15}$$

Therefore, the control inputs of the robot are as follows:

$$v_{max} = 1.5 \text{ m/s and } w_{max} = 3 \text{ rad/s} \tag{16}$$

When the sample time interval to T is set at 0.08 s and the number of prediction steps is N = 10, the weight matrices Q and R are set as follows:

$$\mathrm{Q} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \text{and } \mathrm{R} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.05 \end{bmatrix}. \tag{17}$$

4.1.1. Static Obstacle Avoidance

When the obstacle coordinates are set to [2.5, −1.5] and the coordinate points are located at the lower left corner of the square obstacle, the obstacle-avoidance effect is as shown in Figure 4.

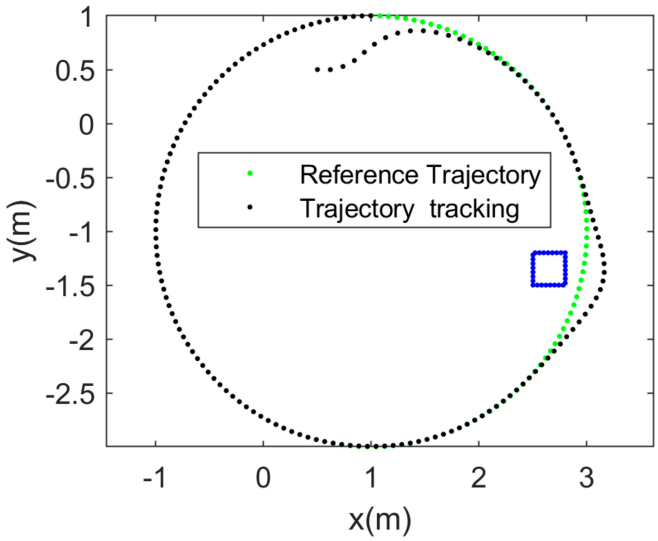
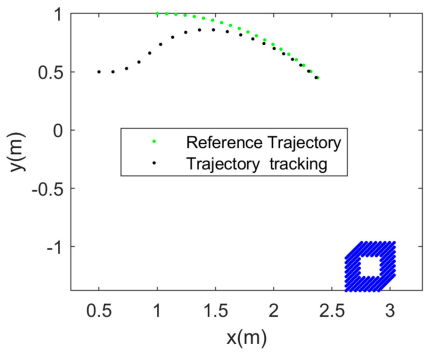


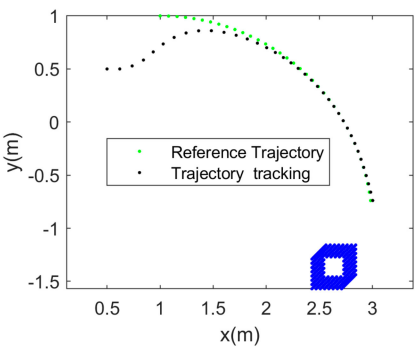
Figure 4. Static obstacle-avoidance effect diagram of circular trajectory tracking.

4.1.2. Dynamic Obstacle Avoidance

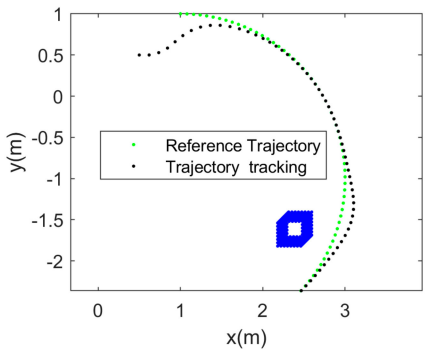
When the obstacle coordinates are set to [2.95, −1.05], the coordinate points are located at the lower left corner of the square obstacle, the moving direction of the obstacle is $\pi/4$, and the speed is 0.2 m/s, the obstacles at the position and the obstacles at the predicted 10 step positions are shown in Figure 5. Figure 5a shows the trajectory tracking at the beginning, Figure 5b shows the robot avoiding obstacles when it encounters moving obstacles, Figure 5c shows the robot tracking to the reference track after obstacle avoidance, and Figure 5d shows the robot moving to the final position.



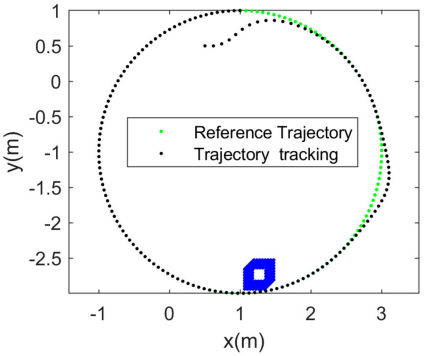
(**a**) Robot and obstacle position at 1.6 s



(**b**) Robot and obstacle position at 2.96 s



(**c**) Robot and obstacle position at 4.72 s



(**d**) Robot and obstacle position at 13.36 s

Figure 5. Dynamic obstacle-avoidance effect of circular trajectory tracking.

4.1.3. Experimental Results

By observing Figures 4–11, it can be seen that the algorithm allows the robot to effectively avoid static and dynamic obstacles. At 1.6 s, the robot completes trajectory tracking. At 2.96 s, the velocity, angular velocity, and direction of the robot for dynamic obstacle avoidance starts to diverge from the static obstacle avoidance, and the robot begins to avoid obstacles dynamically. At 4.72 s, the robot completes the dynamic obstacle avoidance. At 6.4 s, the robot's velocity, angular velocity, and direction angle stabilize, and the errors in each direction approach zero.

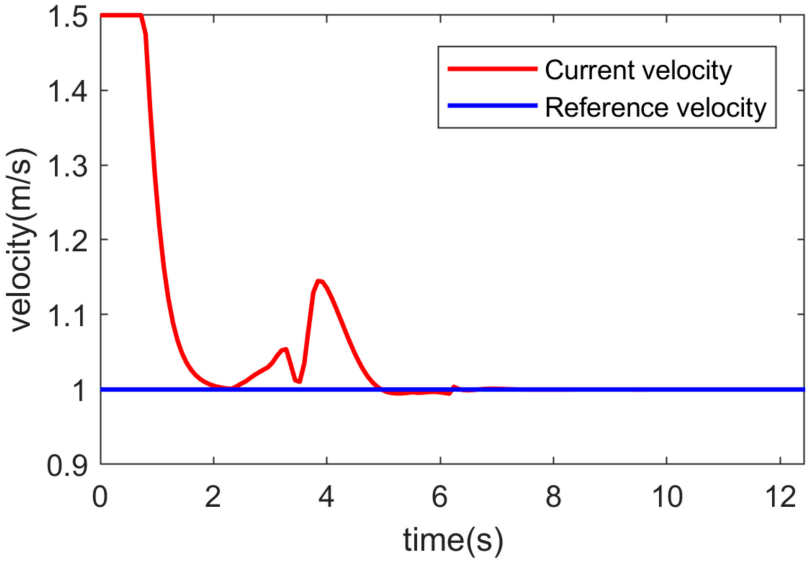


Figure 6. Variation in velocity and reference velocity of a circular trajectory.

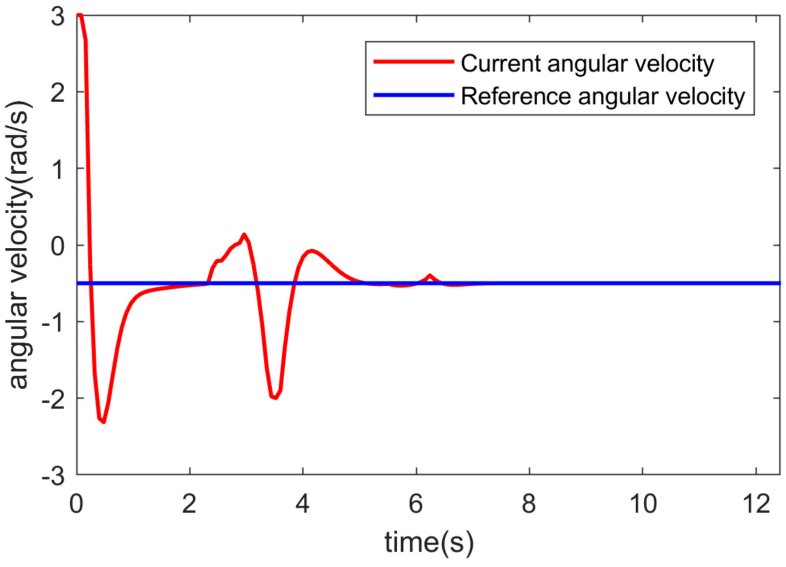


Figure 7. Variation in angular velocity and reference angular velocity of a circular trajectory.

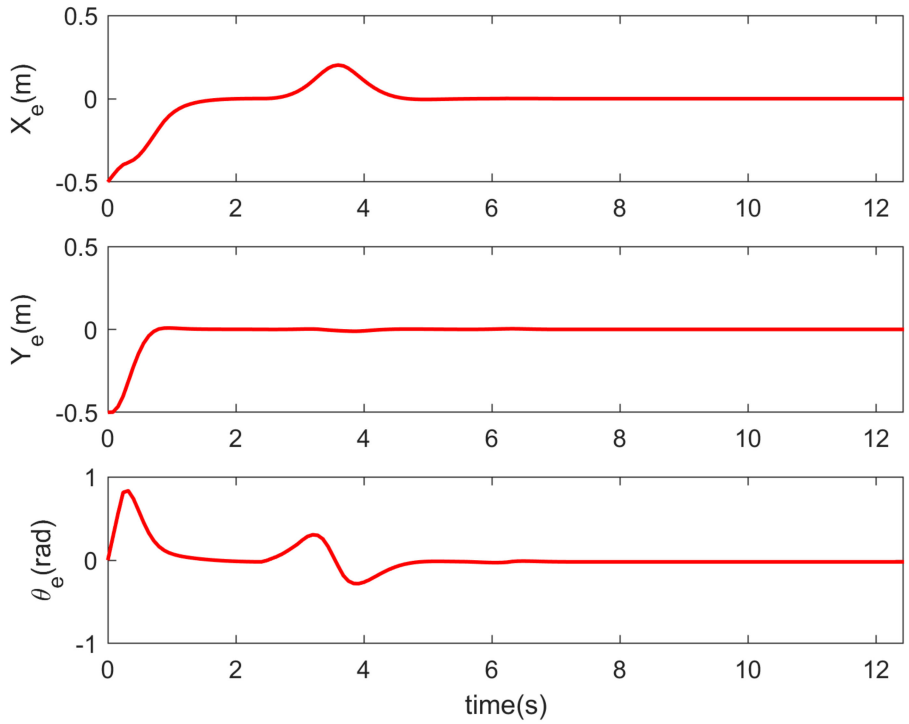


Figure 8. X-direction, Y-direction, and θ-direction error changes of a circular trajectory.

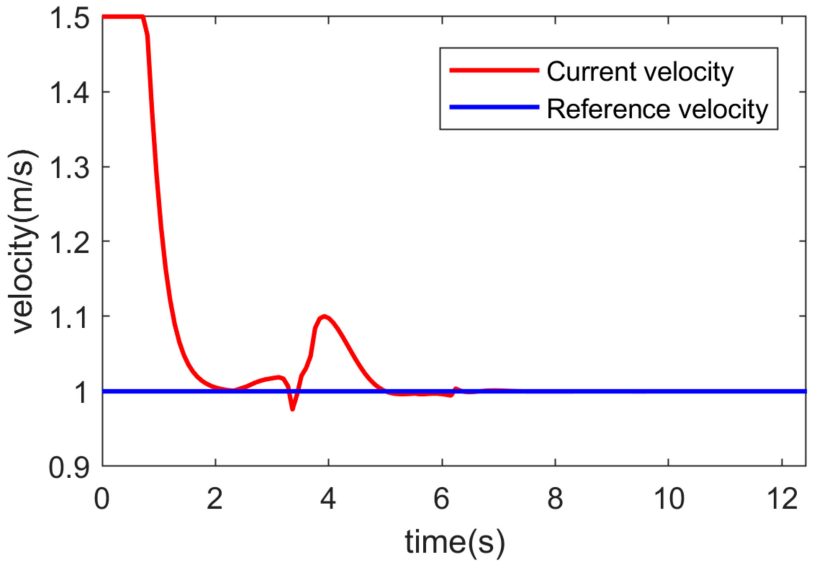


Figure 9. Variation in velocity and reference velocity of a circular trajectory.

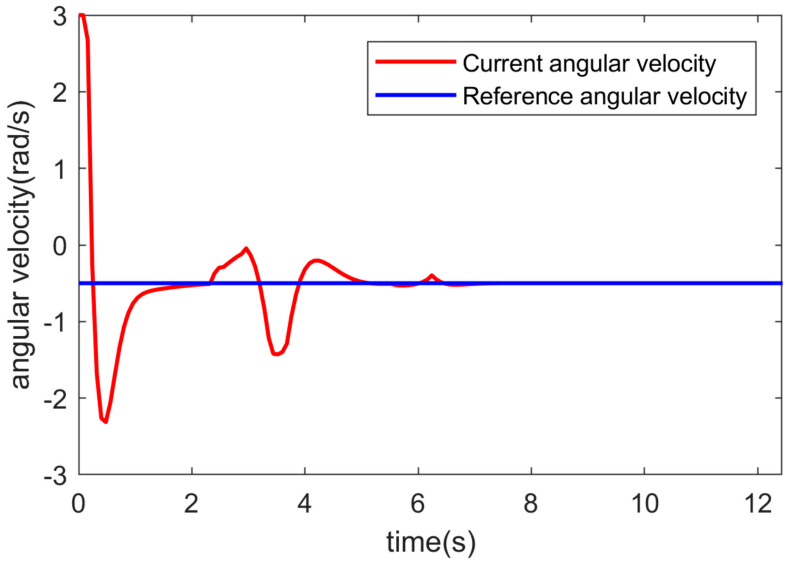


Figure 10. Variation in angular velocity and reference angular velocity of a circular trajectory.

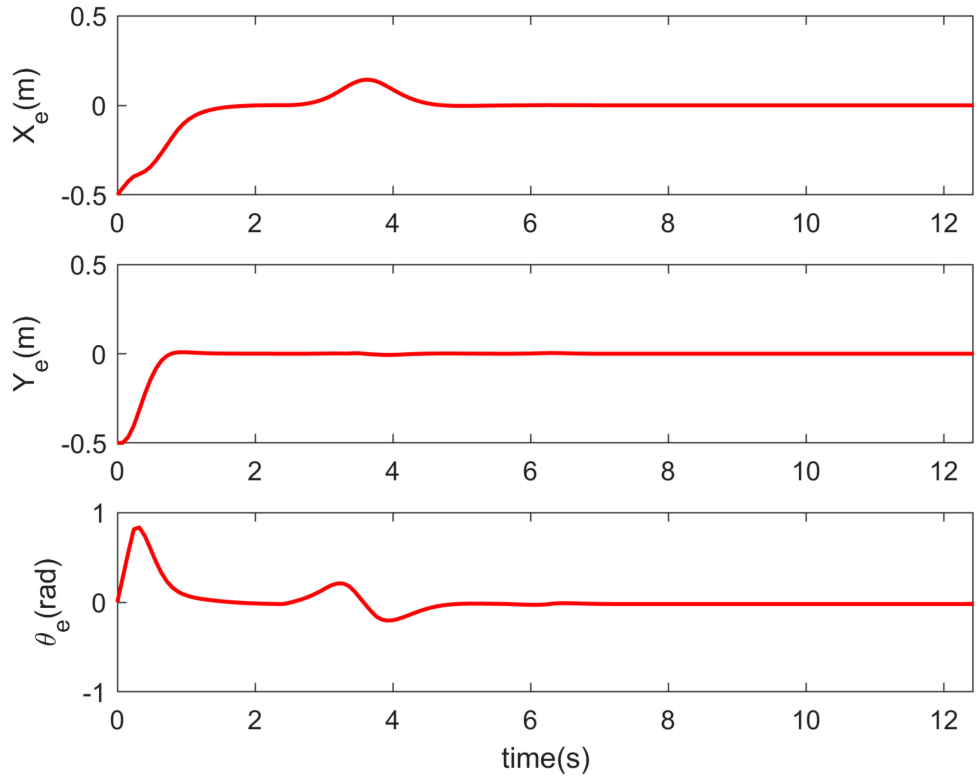


Figure 11. X-direction, Y-direction, and θ-direction error changes of a circular trajectory.

4.2. Figure-of-Eight Trajectories

When the starting position of the robot is set to [1; 2; 0], the reference trajectory is as follows:

$$\begin{aligned} xr &= 1 + 2 * \sin(0.3 * t), \\ yr &= -1 + 2 * \cos(0.15 * t) \end{aligned} \tag{18}$$

The control inputs of the robot are as follows:

$$v_{max} = 3\ \mathrm{m/s} \text{ and } w_{max} = 3\ \mathrm{rad/s}. \tag{19}$$

When the sample time interval is set to T = 0.08 s and the number of prediction steps is N = 10, the weight matrices Q and R are set as follows:

$$Q = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 0.2 \end{bmatrix} \text{and } R = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.05 \end{bmatrix}. \quad (20)$$

4.2.1. Static Obstacle Avoidance

When the obstacle coordinates are set to [2, −1.9] and the coordinate points are located at the lower left corner of the square obstacle, the obstacle-avoidance effect is as shown in Figure 12.

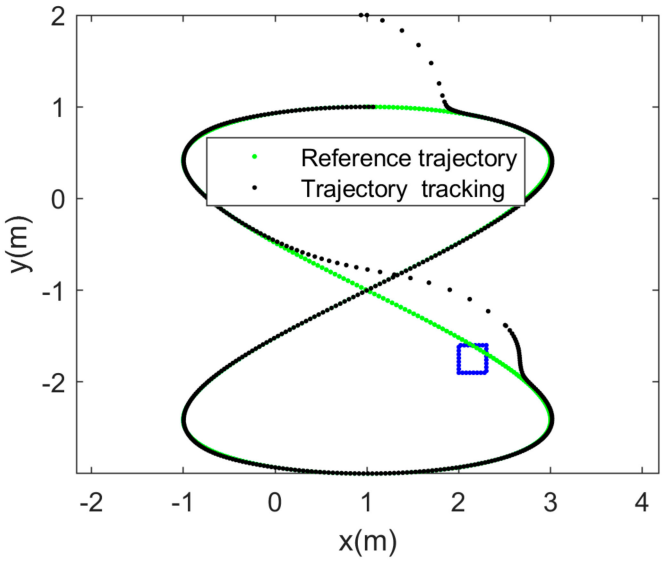
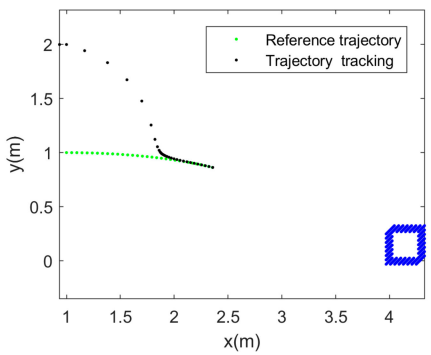


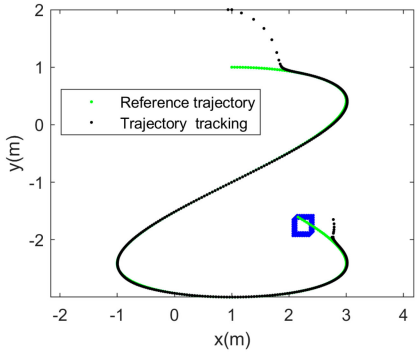
Figure 12. Static obstacle-avoidance effect diagram of figure-of-eight-shaped trajectory tracking.

4.2.2. Dynamic Obstacle Avoidance

When the obstacle coordinates are to [4.2, 0.2], the coordinate points are located at the lower left corner of the square obstacle, the moving direction of the obstacle is π/4, and the speed is 0.1 m/s, the dynamic obstacle avoidance is shown in Figure 13. Figure 13a shows the trajectory tracking at the beginning, Figure 13b shows the robot avoiding obstacles when it encounters moving obstacles, Figure 13c shows the robot tracking to the reference track after obstacle avoidance, and Figure 13d shows the robot moving to the final position.

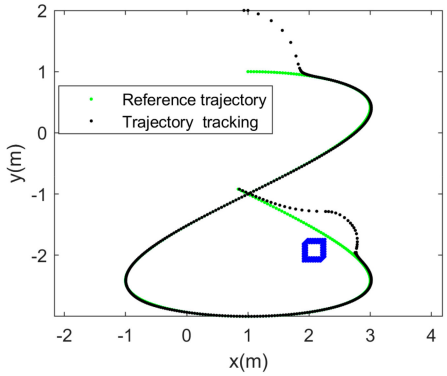


(**a**) Robot and obstacle position at 2.56 s

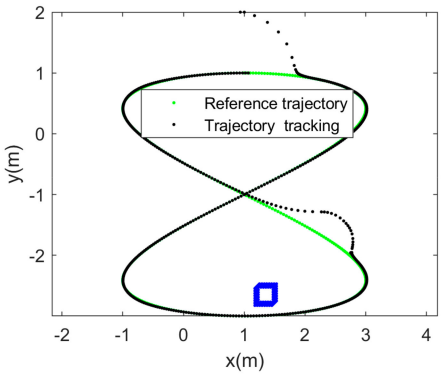


(**b**) Robot and obstacle position at 29.44 s

Figure 13. *Cont.*



(**c**) Robot and obstacle position at 31.76 s



(**d**) Robot and obstacle position at 42 s

Figure 13. Dynamic obstacle-avoidance effect diagram of figure-of-eight-shaped trajectory tracking.

4.2.3. Experimental Results

As seen in Figures 12–19, the algorithm enables the robot to effectively avoid static and dynamic obstacles. At 2.28 s, the robot completes trajectory tracking. At 27.52 s, the velocity, angular velocity, and direction of the robot for dynamic obstacle avoidance diverge from the static obstacle avoidance and the robot starts to avoid obstacles dynamically. At 31.52 s, the robot completes the dynamic obstacle avoidance. At the same time, the velocity, angular velocity, and direction angle of the robot stabilize, with the error in each direction approaching zero.

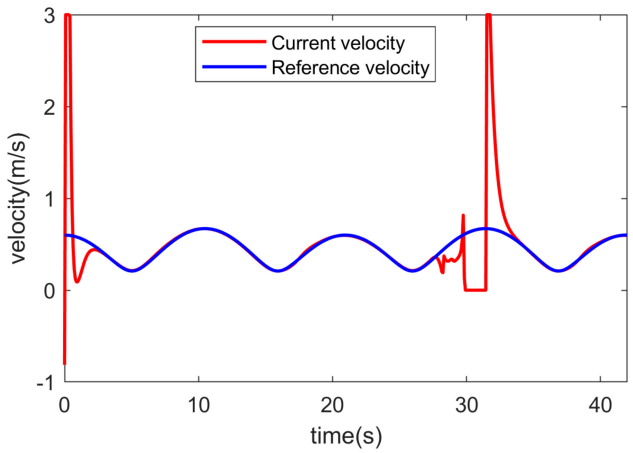


Figure 14. In velocity and reference velocity of a figure-of-eight-shaped trajectory.

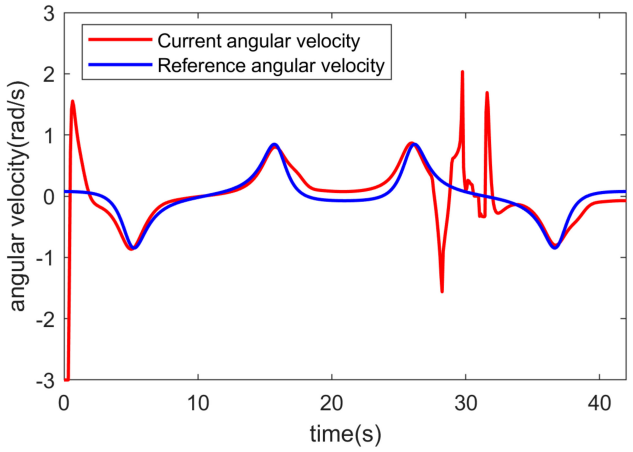


Figure 15. In angular velocity and reference angular velocity of a figure-of-eight-shaped trajectory.

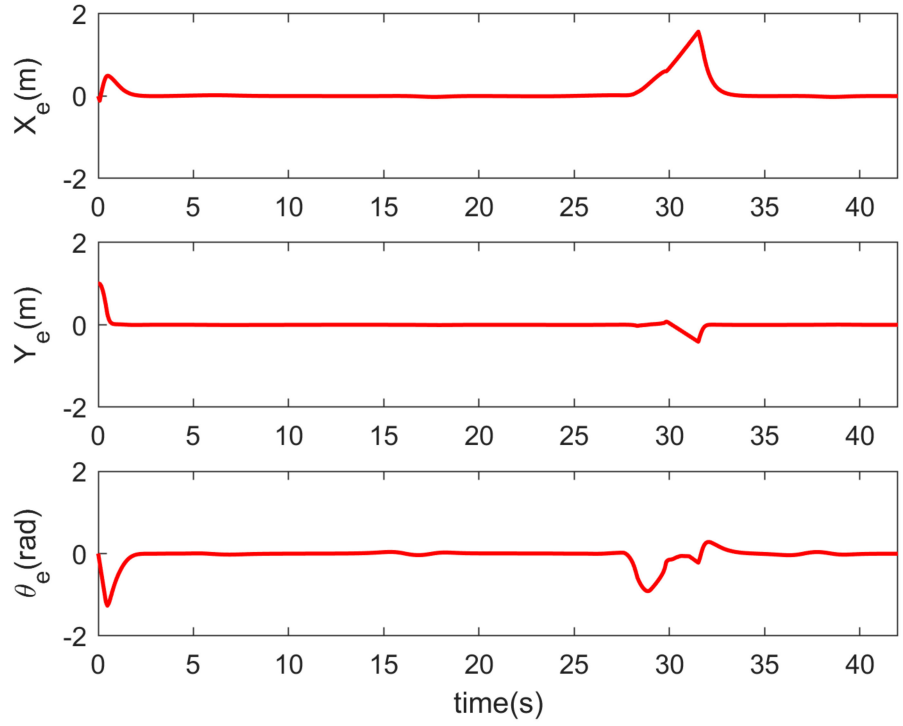


Figure 16. X-direction, Y-direction, and θ-direction error changes of a figure-of-eight-shaped trajectory.

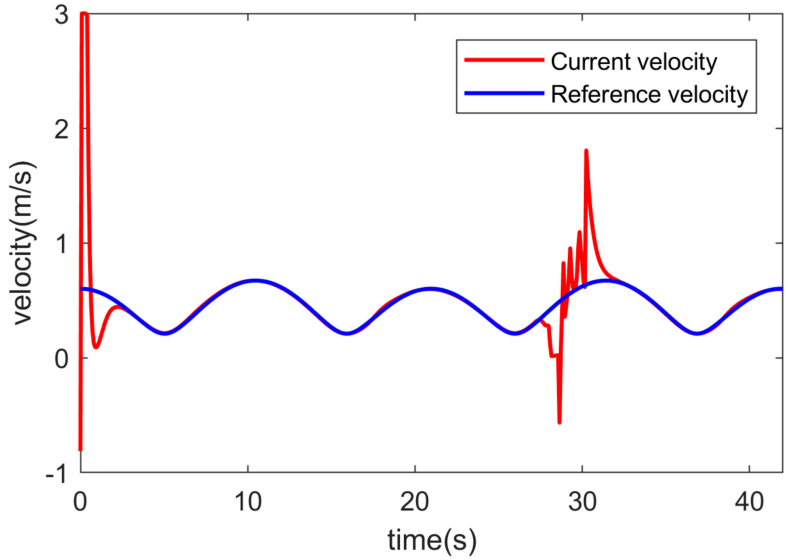


Figure 17. Variation in velocity and reference velocity of a figure-of-eight-shaped trajectory.

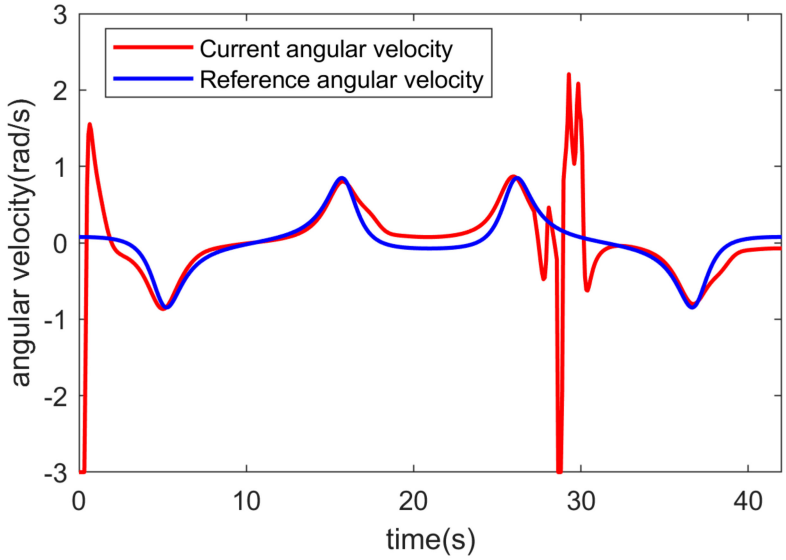


Figure 18. Variation in angular velocity and reference angular velocity of a figure-of-eight-shaped trajectory.

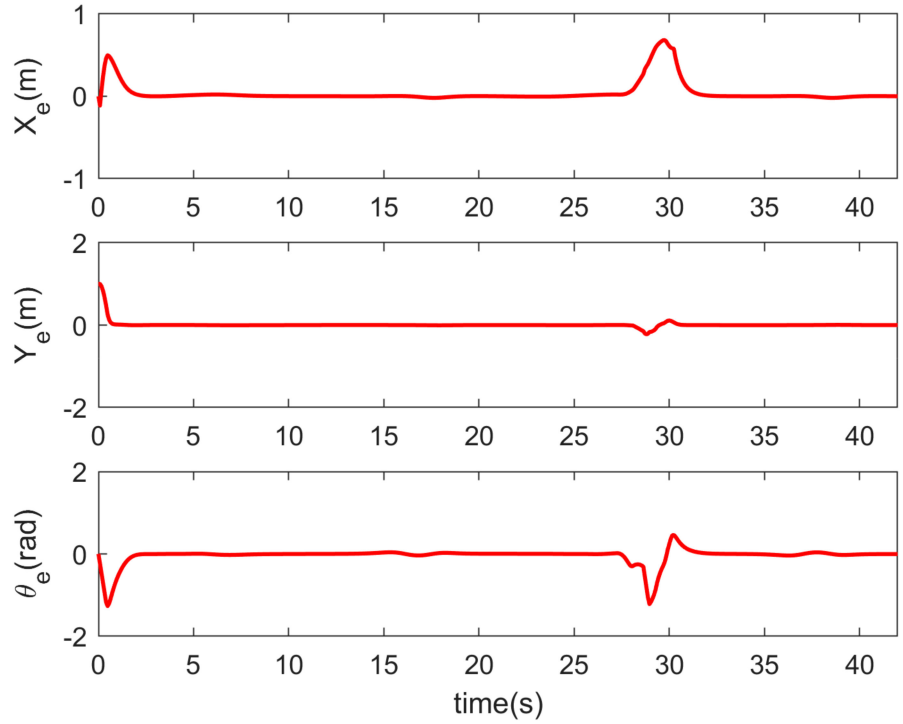


Figure 19. X-direction, Y-direction, and θ-direction error changes of a figure-of-eight-shaped trajectory.

4.3. Linear Trajectory

When the starting position of the robot is set to [0; 1; 0], the reference trajectory is as follows:

$$\begin{aligned} \text{xr} &= \text{t}, \\ \text{yr} &= 0. \end{aligned} \tag{21}$$

The control inputs of the robot are as follows:

$$v_{max} = 1.5 \text{ m/s and } w_{max} = 1.5 \text{ rad/s} \tag{22}$$

When the sample time interval is set to T = 0.08 s and the number of prediction steps is N = 10, the weight matrices Q and R are set as follows:

$$Q = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \text{and } R = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.05 \end{bmatrix} \quad (23)$$

4.3.1. Obstacle Avoidance

When the dynamic obstacle coordinates are set to [0.8, 1.2] and the coordinate points are located in the lower left corner of the dynamic square obstacle, the moving direction of the obstacle is 0, the speed is 1 m/s, and the acceleration is a uniform deceleration movement of -0.25 m/s^2; movement stops when the obstacle speed is 0 m/s. Figure 20 shows the robot avoiding obstacles when encountering moving obstacles, whereas Figure 21 shows the robot moving to the final position.

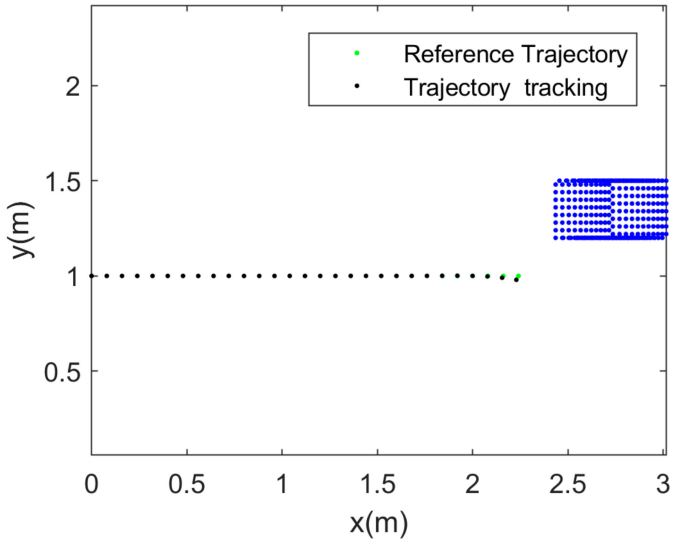


Figure 20. Robot and obstacle position at 2.32 s.

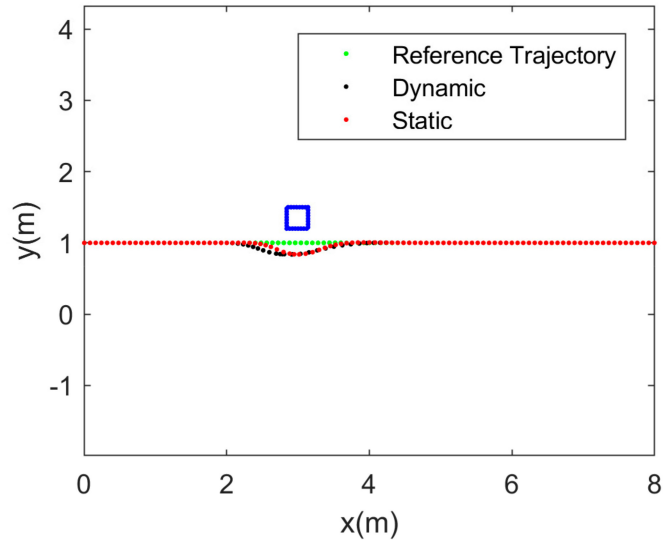


Figure 21. Robot and obstacle position at 8 s.

When the static obstacle coordinates are set to [2.8, 1.2] and the coordinate points are located in the lower left corner of the square obstacle, the obstacle avoidance effect is as shown Figure 21.

4.3.2. Experimental Results

As seen in Figures 20–24, the algorithm can enable the robot to effectively avoid static and dynamic obstacles. At 2.32 s, the speed, angular velocity, and direction of the robot used for dynamic obstacle avoidance diverge from static obstacle avoidance and the robot starts to avoid dynamic obstacles. At 4 s, the dynamic obstacle stops at the same position as the static obstacle. At 4.16 s, the obstacle avoidance is completed and the time, speed, angular velocity, and direction angle of the robot are considered to be stable, with the error in each direction close to zero.

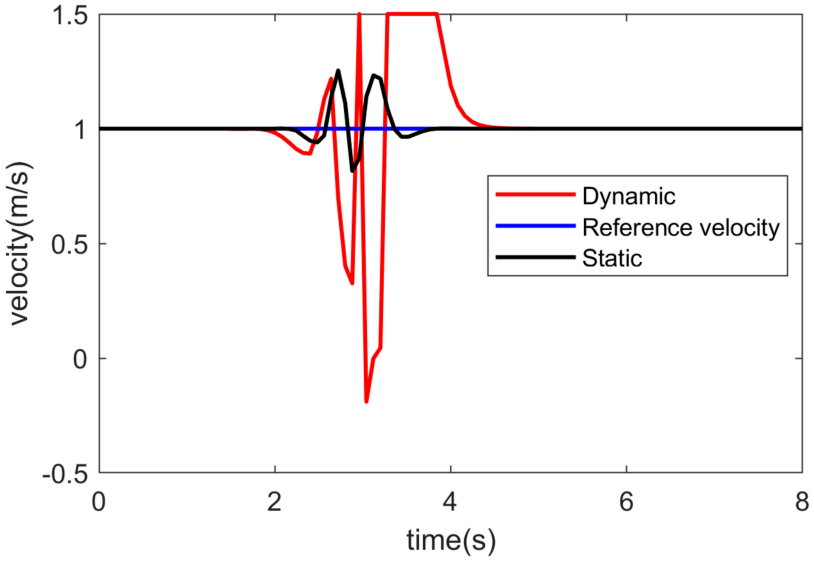


Figure 22. Variation in velocity and reference velocity of a linear trajectory.

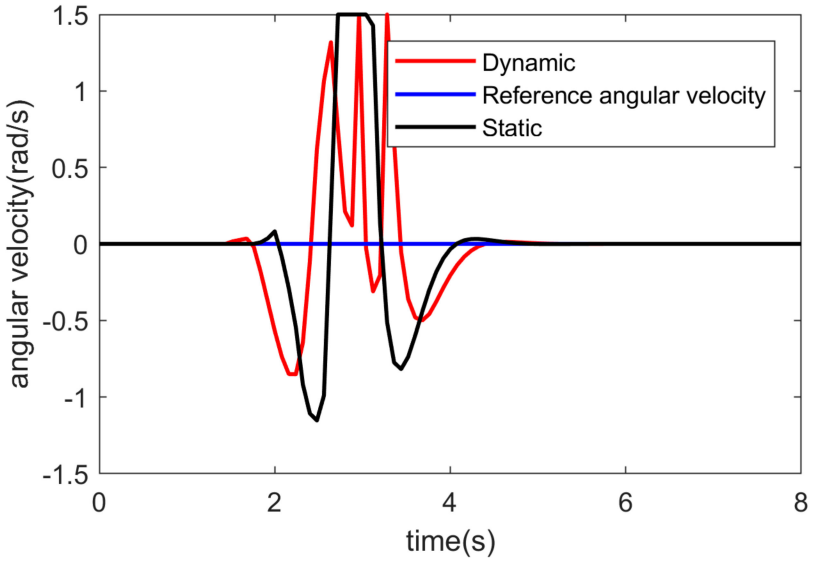


Figure 23. Variation in angular velocity and reference angular velocity of a linear trajectory.

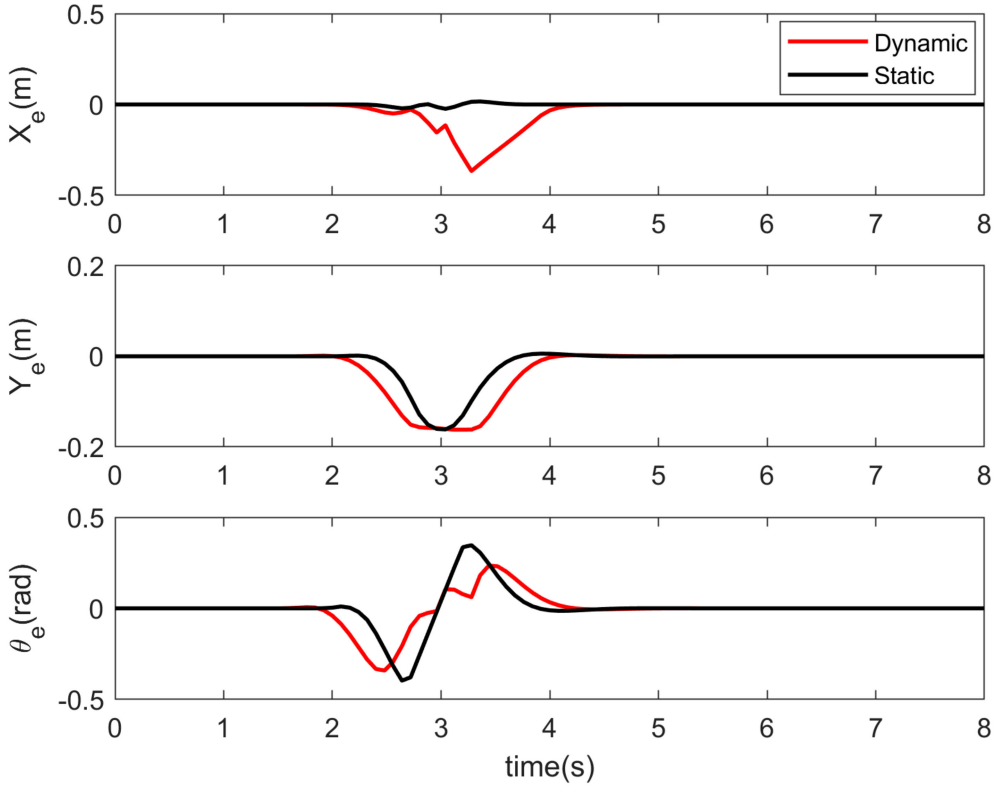


Figure 24. X-direction, Y-direction, and θ-direction error changes of a linear trajectory.

5. Conclusions

A model-based predictive control scheme is proposed in this paper to solve the problem of robot dynamic obstacle avoidance. Collision avoidance as a nonlinear constraint condition of changes in position state with time is embedded in the trajectory tracking control problem. Through MATLAB simulation, the robot was shown to successfully avoid static obstacles and dynamic obstacles. However, the dynamic obstacle avoidance control scheme in this paper is only theoretical. Actual road traffic conditions, which are more complex, were not considered in this work. Therefore, the obstacle avoidance strategies need to be further improved. Moreover, when a vehicle is actually driving, the position of the moving obstacle cannot be accurately obtained; further research is needed to obtain accurate obstacle positions.

Author Contributions: Conceptualization, R.G. and J.Z.; methodology, K.Z.; software, K.Z.; validation, K.Z.; formal analysis, K.Z.; investigation, K.Z.; resources, K.Z.; data curation, K.Z.; writing—original draft preparation, K.Z.; writing—review and editing, R.G.; project administration, R.G.; funding acquisition, J.Z. and R.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Hebei Natural Science Foundation Project "Research and Development of Polishing Industrial Robot Perception System Based on Multi-sensor Information Fusion" (F2017402182) and Handan City Science and Technology Research and Development Plan Project: 1328103077-2, 1521109072-5.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ismail, A.L.T.; Alaa, S.; Mohammed, A.-W. A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment. *J. Comput. Sci.* **2008**, *4*, 341–344. [CrossRef]
2. Li, Q.; Zhang, C.; Han, C.; Xu, Y.; Yin, Y.; Zhang, W. Path planning based on fuzzy logic algorithm for mobile robots in static environment. In Proceedings of the 2013 25th Chinese Control and Decision Conference (CCDC), Guiyang, China, 18 July 2013; pp. 2866–2871.

3. Dao, T.; Pan, T.; Pan, J. A multi-objective optimal mobile robot path planning based on whale optimization algorithm. In Proceedings of the 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, China, 6–10 November 2016; pp. 337–342.
4. Švestka, P.; Overmars, M.H. Motion Planning for Carlike Robots Using a Probabilistic Learning Approach. *Int. J. Robot. Res.* **1997**, *16*, 119–143. [CrossRef]
5. Tian, J.; Gao, M.; Lu, E. Dynamic Collision Avoidance Path Planning for Mobile Robot Based on Multi-sensor Data Fusion by Support Vector Machine. In Proceedings of the 2007 International Conference on Mechatronics and Automation, Harbin, China, 5–9 August 2007; pp. 2779–2783.
6. Riid, A.; Pahhomov, D.; Rüstern, E. Car navigation and collision avoidance system with fuzzy logic. In Proceedings of the 2004 IEEE International Conference on Fuzzy Systems (IEEE Cat. No.04CH37542), Budapest, Hungary, 25–29 July 2004; pp. 1443–1448.
7. Sugeno, M. Fuzzy control: Principles, practice and perspectives. In Proceedings of the IEEE International Conference on Fuzzy Systems, San Diego, CA, USA, 8–12 March 1992; p. 109.
8. Tanaka, K. Model-based fuzzy control of a trailer type mobile robot. In Proceedings of the 1995 IEEE International Conference on Fuzzy Systems, Yokohama, Japan, 20–24 March 1995; pp. 65–70.
9. Song, K.-T.; Sheen, L.-H. Fuzzy-neuro control design for obstacle avoidance of a mobile robot. In Proceedings of the 1995 IEEE International Conference on Fuzzy Systems, Yokohama, Japan, 20–24 March 1995; pp. 71–76.
10. Tang, L.; Dian, S.; Gu, G.; Zhou, K.; Wang, S.; Feng, X. A novel potential field method for obstacle avoidance and path planning of mobile robot. In Proceedings of the 2010 3rd International Conference on Computer Science and Information Technology, Chengdu, China, 9–11 July 2010; pp. 633–637.
11. Zhang, N.; Zhang, Y.; Ma, C.; Wang, B. Path planning of six-DOF serial robots based on improved artificial potential field method. In Proceedings of the 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, China, 5–8 December 2017; pp. 617–621.
12. Liu, Z.; Jiang, T. Route planning based on improved artificial potential field method. In Proceedings of the 2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Wuhan, China, 16–19 June 2017; pp. 196–199.
13. Azzabi, A.; Nouri, K. Path planning for autonomous mobile robot using the Potential Field method. In Proceedings of the 2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET), Hammamet, Tunisia, 14–17 January 2017; pp. 389–394.
14. Utkin, V.I.; Drakunov, S.V.; Hashimoto, H.; Harashima, F. Robot path obstacle avoidance control via sliding mode approach. In Proceedings of the IROS '91: IEEE/RSJ International Workshop on Intelligent Robots and Systems '91, Osaka, Japan, 3–5 November 1991; pp. 1287–1290.
15. PSen, P.T.H.; Minh, N.Q.; Anh, D.T.T.; Minh, P.X. A New Tracking Control Algorithm for a Wheeled Mobile Robot Based on Backstepping and Hierarchical Sliding Mode Techniques. In Proceedings of the 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, 16–18 January 2019; pp. 25–28.
16. Li, Y.-D.; Wang, Z.; Zhu, L. Adaptive neural network PID sliding mode dynamic control of nonholonomic mobile robot. In Proceedings of the 2010 IEEE International Conference on Information and Automation, Harbin, China, 20–23 June 2010; pp. 753–757.
17. Boukadida, W.; Bkekri, R.; Benamor, A.; Messaoud, H. Trajectory tracking of robotic manipulators using optimal sliding mode control. In Proceedings of the 2017 International Conference on Control, Automation and Diagnosis (ICCAD), Hammamet, Tunisia, 19–21 January 2017; pp. 545–550.
18. Ferreau, H.J.; Almér, S.; Peyrl, H.; Jerez, J.L.; Domahidi, A. Survey of industrial applications of embedded model predictive control. In Proceedings of the 2016 European Control Conference (ECC), Aalborg, Denmark, 29 June–1 July 2016; p. 601.
19. Qin, S.J.; Badgwell, T.A. A survey of industrial model predictive control technology. *Control. Eng. Pr.* **2003**, *11*, 733–764. [CrossRef]
20. Li, Z.; Yang, C.; Su, C.-Y.; Deng, J.; Zhang, W. Vision-Based Model Predictive Control for Steering of a Nonholonomic Mobile Robot. *IEEE Trans. Control Syst. Technol.* **2016**, *24*, 553–564. [CrossRef]
21. Van Essen, H.; Nijmeijer, H. Non-linear model predictive control for constrained mobile robots. In Proceedings of the 2001 European Control Conference (ECC), Porto, Portugal, 4–7 September 2001; pp. 1157–1162.

22. Falcone, P.; Tufo, M.; Borrelli, F.; Asgari, J.; Tseng, H.E. A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. In Proceedings of the 2007 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 12–14 December 2007; pp. 2980–2985.
23. Klančar, G.; Škrjanc, I. Tracking-error model-based predictive control for mobile robots in real time. *Robot. Auton. Syst.* **2007**, *55*, 460–469. [CrossRef]
24. Gu, D.; Hu, H. Receding horizon tracking control of wheeled mobile robots. *IEEE Trans. Control Syst. Technol.* **2006**, *14*, 743–749.
25. Ostafew, C.J.; Schoellig, A.P.; Barfoot, T.D. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 4029–4036.
26. Mehrez, M.W.; Mann, G.K.I.; Gosine, R.G. Stabilizing NMPC of wheeled mobile robots using open-source real-time software. In Proceedings of the 2013 16th International Conference on Advanced Robotics (ICAR), Montevideo, Uruguay, 25–29 November 2013; pp. 1–6.
27. Lim, H.; Kang, Y.; Kim, C.; Kim, J.; You, B.-J. Nonlinear Model Predictive Controller Design with Obstacle Avoidance for a Mobile Robot. In Proceedings of the 2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications, Beijing, China, 12–15 October 2008; pp. 494–499.
28. Park, J.M.; Kim, D.-W.; Yoon, Y.; Kim, H.J.; Yi, K.-S. Obstacle avoidance of autonomous vehicles based on model predictive control. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2009**, *223*, 1499–1516. [CrossRef]
29. Ji, J.; Khajepour, A.; Melek, W.W.; Huang, Y. Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control with Multiconstraints. *IEEE Trans. Veh. Technol.* **2017**, *66*, 952–964. [CrossRef]
30. Kanayama, Y.; Kimura, Y.; Miyazaki, F.; Noguchi, T. A stable tracking control method for an autonomous mobile robot. In Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 13–18 May 1990; pp. 384–389.