applied sciences

MDPI

Article

Obstacle Avoidance in a Three-Dimensional Dynamic Environment Based on Fuzzy Dynamic Windows

Ce Xu [1,2], Zhenbang Xu [1,3,*] and Mingyi Xia [1]

[1] CAS Key Laboratory of On-Orbit Manufacturing and Integration for Space Optics System, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China; xuce16@mails.ucas.ac.cn (C.X.); xiamingyi@ciomp.ac.cn (M.X.)
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Center of Materials Science and Optoelectronics Engineering, University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: xuzhenbang@ciomp.ac.cn

Abstract: This paper presents a real-time path planning approach for controlling the motion of space-based robots. The algorithm can plan three-dimensional trajectories for agents in a complex environment which includes numerous static and dynamic obstacles, path constraints, and/or performance constraints. This approach is extended based on the dynamic window approach (DWA). As the classic reactive method for obstacle avoidance, DWA uses an optimized function to select the best motion command. The original DWA optimization function consists of three weight terms. Changing the weights of these terms will change the behavior of the algorithm. In this paper, to improve the evaluation ability of the optimization function and the robot's ability to adapt to the environment, a new optimization function is designed and combined with fuzzy logic to adjust the weights of each parameter of the optimization function. Given that DWA has the defect of local minima, which makes the robot hard to escape U-shaped obstacles, a dual dynamic window method and local goals are adopted in this article to help the robot escape local minima. By comparison, the proposed method is superior to traditional DWA and fuzzy DWA (F_DWA) in terms of computational efficiency, smoothness and security.

Keywords: obstacle avoidance; space-based robots; dynamic window approach; fuzzy logic; path planning

Citation: Xu, C.; Xu, Z.; Xia, M. Obstacle Avoidance in a Three-Dimensional Dynamic Environment Based on Fuzzy Dynamic Windows. *Appl. Sci.* **2021**, *11*, 504. https://doi.org/10.3390/app11020504

Received: 25 November 2020
Accepted: 1 January 2021
Published: 6 January 2021

1. Introduction

It is expected that space autonomous robotics will be used to complete complex and dangerous tasks in space as space technology develops [1]. As a robot completes a mission, it needs to plan a safe trajectory from a starting point to a target point. Collisions and path optimization are the main issues relating to the planning of the trajectory (i.e., navigation). Currently, navigation algorithms fall into two categories: global planning algorithms and local navigation algorithms. Previous studies have also combined the above two types of algorithms to achieve better navigation, but such a combination depends on the robot's autonomy and environmental factors [2].

Some notable global navigation algorithms avoid the collision problem by building a global environment map: e.g., A* [3], D* [4], FD* [5], and RRT [6]. When planning a path on a map, the optimization goal is usually the shortest path or the lowest energy use. However, considering that in the planning process, the speed, and accuracy of graph-based search methods depend on the granularity of the search space, those approaches are not suitable for real-time application.

To improve real-time performance, some studies have proposed local navigation algorithms. Currently, the simplest local navigation algorithms are bug-based algorithms, e.g., bug1, bug2 [7], and tangent bug [8]. When the robot encounters an obstacle, the robot

will move along the boundary of the obstacle until it avoids the obstacle. The classical reactive navigation methods include the VFH (vector field histogram) [9] and its extended versions VFH+ [10] and VFH* [11], which are all based on the virtual vector field method. The moving direction of histogram methods is given by dividing the surrounding environment into different angular sectors of the polar histogram and the next direction of the robot's movement is obtained according to the proximity of an obstacle. Another classical reactive navigation method is the velocity space approach. This approach uses the evaluation function such as safety and smoothness to calculate the next movement direction. Some well-known velocity space algorithms are dynamic-window approach (DWA) [12,13] and curvature velocity method (CVM) [14]. As for the nonlinear problem of moving obstacles, Seder, M. and Petrovic, I. [2] proposed an extension of the DWA algorithm (i.e., the time-varying dynamic-window algorithm), where the moving obstacles are modeled as moving cells in a gridded map. Considering the time factor in the process of obstacle avoidance, Molinos, E. J. et al. proposed DW4DO and DW4DOT methods based on CVM (curvature velocity method) to plan a safer and longer obstacle avoidance path [15], and applied the methods in a dynamic environment. However, the main drawback of the local navigation algorithms is the local minima, especially in the vase of U-shaped obstacles. One approach of avoiding the local minima problem is to select a local target of the robot's surroundings and calculate the direction of the next step. Lane-Curvature Method (LCM) [16] and Beam-Curvature Methods (BCM) [17] algorithms are representatives of this approach and are based on the CVM algorithm. The LCM algorithm obtains the local target by dividing the channel, whereas the BCM obtains the local target by dividing the sector. Moon, J. et al. proposed a hybrid dynamic window approach that can avoid the agent falling into the local minima, which is superior to DWA [18]. Yu, X. Y. et al. proposed a dynamic window method with virtual goal in dynamic environments, which can make the robot escape the concave trap [19]. In addition to these algorithms, there are evolutionary algorithms that are based on neural networks or fuzzy logic [20–22]. Tarokh, M. proposed a hybrid intelligent approach that combines fuzzy logic and genetic algorithm to enable robots to quickly find an optimal path that avoids rough areas, which effectively solves the path planning of highly mobile robots in rough environments [23]. Chen, L. et al. proposed a conditional deep Q-network with fuzzy logic, which aims to handle the issues of directional planning in end-to-end autonomous driving systems and the independence of different motion commands. This approach has better learning performance and driving stability performance, but it does not take into account obstacles such as cars and pedestrians [24].

Whether it is a graph-based search method or a local navigation algorithm, they mostly aim at a flat environment. For path planning in a space environment, popular approaches are mainly to integrate obstacles, speeds, and path lengths into constraints and generate trajectories by curve parameterization [25,26]. Most of the numerical methods use path point parameterization to define splines, polynomials, and so on. However, for nonlinear problems, these parameterization algorithms require a lot of computing resources. Without further assumptions or simplifications, these methods are not suitable for solving the problem of dynamic obstacles. Then, based on a local parameterized guidance vector field, Marchidan, A. et al. proposed a method that can generate a constant velocity vector field by the decomposition of unmanned aerial vehicle (UAV) kinematics into normal and tangential components with respect to the obstacle boundary [27]. This method can effectively avoid static obstacles and dynamic obstacles. However, it also has one limitation; that is, it assumes that agents are at a constant height and hence simplifies the three-dimensional (3D) problem to a two-dimensional (2D) problem. Due to the different power modes provided by the thrusters, the problems of the UAV and the space agent in obstacle avoidance and trajectory planning are not completely the same. However, in terms of multi-objective path planning [28,29], collaborative path planning [30,31] and cluster motion [32], they have similar path planning problems. Based on the Legendre polynomial method, Chamitoff, G. E. et al. proposed the ASTRO (Admissible subspace trajectory

optimizer) method which transforms the complex cost function into a simple convex form and applied it to cope with the problem of optimizing real-time three-dimensional (3D) trajectory [33]. Nonetheless, when the constraints are complex, the numerical method still has the problem of local minima in the optimization process.

To address the navigation problem in the three-dimensional environment, this paper proposes a fuzzy dynamic window approach (DF_DWA). According to the traditional DWA, it is believed that the values of 0.8, 0.1, and 0.1 for the parameters of azimuth, obstacle clearance, and speed, respectively can provide good results in some situations [2]. Nevertheless, as the environment becomes more complex, this set of weights is not suitable in all situations. Different working environments using the same weight coefficient will result in obstacle avoidance failure or make the machine stop working [15]. Zhang, H. et al. suggested using fuzzy logic to update the weight of DWA objective function by analyzing the distribution of eight obstacles [34]. Abubakr, O. A. et al. simplified the distribution of obstacles to three situations, thereby reducing the number of fuzzy logic rules, and improving computational efficiency [35]. However, the two methods mentioned update the weight of the objective function only by the distribution of obstacles, without considering the distance of the obstacles. As a result, the robot can easily fall into U-shaped obstacles or the local minima [36,37]. Moreover, the existing objective function does not consider the pointing problem when the agent approaches the target. In this article, by adding the term of the distance target, a new objective function is established to eliminate the influence of angle orientation. The weight parameters are adjusted by fuzzy logic so that the space robot can adapt to environmental changes. The dual dynamic window method and local goal method are used to avoid the agent trapping in the local minima and U-shaped obstacles.

The paper is organized as follows: Section 2 describes the collision-avoidance problem. Section 3 designs fuzzy rules and describes the dynamic-window approaches. The effectiveness of the method is analyzed by simulation in Section 4. Finally, conclusions are drawn from the results of the study in Section 5.

2. Formulation of the Collision-Avoidance Problem

In this paper, we mainly focus on the 3D real-time trajectory planning of the kind of space-based robots, such as SPHERES [38] and astronaut assistant robots [39]. The agent can work in a space station or specific working range in which the flying distance is much shorter than the radius of the orbit. In addition, the algorithm can also be extended to other types of agent 3D trajectory planning. This section provides the mathematical model of the motion and the attitude of the spatial agent. In a 3D environment, the attitude of the agent is represented by the pitch angle θ, yaw angle ψ, and roll angle ϕ, as shown in Figure 1. Under reference coordinate system (X_f, Y_f, Z_f), the equation of the agent's motion is written as:

$$\begin{bmatrix} X_{f(t+1)} \\ Y_{f(t+1)} \\ Z_{f(t+1)} \end{bmatrix} = \begin{bmatrix} X_{f(t)} \\ Y_{f(t)} \\ Z_{f(t)} \end{bmatrix} + \Delta t * \boldsymbol{R}_b^f \begin{bmatrix} v_x^b \\ v_y^b \\ v_z^b \end{bmatrix} \tag{1}$$

where the Δt is the sampling time. The conversion between the reference coordinate system $oX_fY_fZ_f$ and the body coordinate system $oX_bY_bZ_b$ is in the order of 321. Therefore, the transformation matrix from the body coordinate system to the reference coordinate system is:

$$\boldsymbol{R}_b^f = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2}$$

where c and s are the cosine and sine, respectively. The relationship between the angular velocity component and Euler angle is:

$$\boldsymbol{R}(\boldsymbol{\xi}) = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & s_\phi c_\theta \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \tag{3}$$

$$\dot{\boldsymbol{\xi}} = \boldsymbol{R}(\boldsymbol{\xi})^{-1}\boldsymbol{\omega} \tag{4}$$

where $\boldsymbol{\xi} = [\phi, \theta, \psi]^{\mathrm{T}}$, $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^{\mathrm{T}}$, ω_x, ω_y and ω_z are the components of the angular velocity $\boldsymbol{\omega}$ in the axis of the body coordinate. The equation of the attitude of the agent is written as:

$$\begin{bmatrix} \phi_{(t+1)} \\ \theta_{(t+1)} \\ \psi_{(t+1)} \end{bmatrix} = \begin{bmatrix} \phi_{(t)} \\ \theta_{(t)} \\ \psi_{(t)} \end{bmatrix} + \dot{\boldsymbol{\xi}} * \Delta t \tag{5}$$

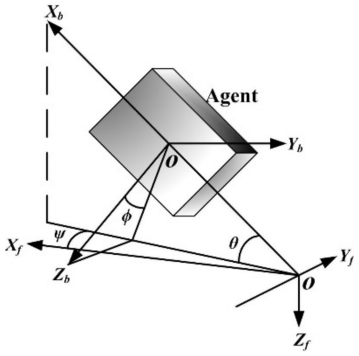


Figure 1. Agent coordinate system.

The trajectory of the agent is generated by Equations (1) and (5). Due to the limitation of the sensor, the activity range that the agent can perceive in a certain time is limited; for example, the effective detection distance of a vehicle-mounted lidar is within 6–8 m. Taking the geometric center of the agent as the origin, the range can be obtained by:

$$S(t) = \left\{ \boldsymbol{p} \in \mathbb{R}^3 : \|\boldsymbol{p}(t) - \boldsymbol{p}\| \leq \mathrm{r}_s \right\} \tag{6}$$

where $\boldsymbol{P} = [x, y, z]$ is the position of the agent, $\boldsymbol{P}(t)$ is the point within the detection range of the agent. r_s is the detection radius of the sensor. Within this range, the space occupied by obstacles can be expressed as:

$$(t) = \left\{ \boldsymbol{p} \in \mathbb{R}^3 : \|\boldsymbol{p}(t) - \boldsymbol{p}_o(t)\| \leq \mathrm{r}_o \right\} \tag{7}$$

where $\boldsymbol{P}_o(t) = [x_o(t), y_o(t), z_o(t)]$ is the position of the obstacles, $\mathrm{r_o}$ is the radius of the obstacles. In this paper, obstacles and agents are assumed to be spherical with the radius of $R_{obstacle}$ and R_{agent}. Users can also refer to the reference [40] to configure agents and obstacles by different boundary constraints. In addition, the obstacle information can be obtained by the sensors [41]. All the admissible positions of the agent can be expressed as:

$$\boldsymbol{\Omega}(t) = \boldsymbol{S}(t) \cap (\boldsymbol{o}(t) + \boldsymbol{S}_b(t)) \tag{8}$$

where the $\boldsymbol{S}_b(t)$ is an added safety zone that accounts for measurement and process uncertainty. Because of the limitation of sensors, new obstacles will appear in the search area, and the environment within the search range is changing, which requires the algorithm itself to be able to adapt to the change. In addition, some obstacles may also be dynamic, so the admissible range of the agent is a time-varying area. In response to these problems, we propose a fuzzy dynamic window method, which will be discussed in Section 3.

3. Proposed Dynamic Window Approach

DWA is one of the commonly used methods for local obstacle avoidance. During an iteration, the algorithm can calculate all feasible speed groups for the next iteration. The optimal motion combination is then selected by the evaluation function. The algorithm adopted in the present paper differs from the original algorithm as described below:

(1) DWA is extended to a 3D dynamic environment.
(2) New evaluation items are added to the original evaluation function and therefore a new evaluation function is established. Fuzzy logic is also introduced to adjust the weight of each evaluation item according to the working conditions. The distance from the agent to the goal and the distance between the agent and the obstacle are the basis for assessing the adjustment.
(3) Two different velocity windows are evaluated at the same time. To ensure the safety of the trajectory, the maximum velocity window is adopted to calculate the braking distance and the distance to the obstacle.
(4) Local goals are used to avoid large obstacles or U-shaped obstacles.

Figure 2 summarizes the algorithm workflow.

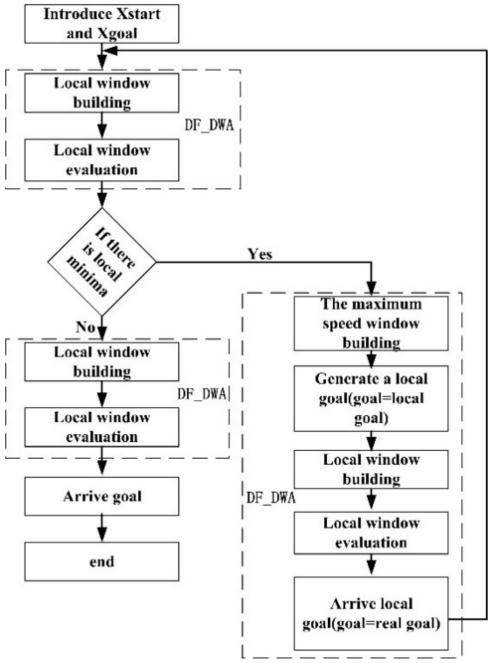
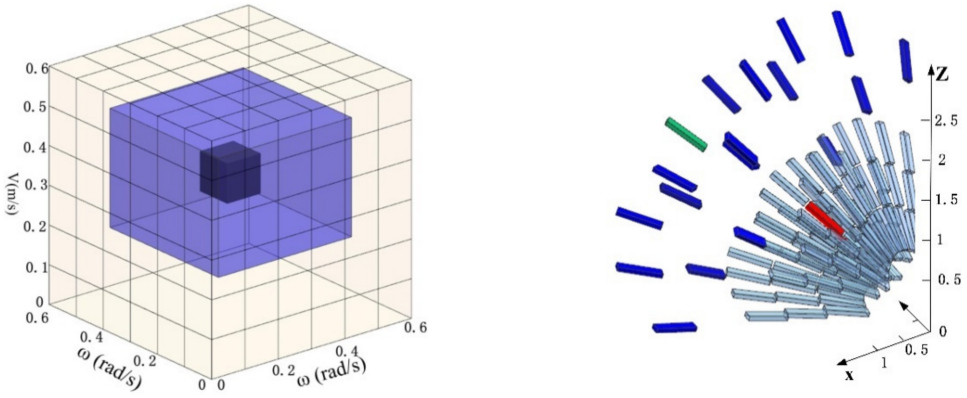


Figure 2. The DF_DWA flowchart.

3.1. The Two Velocity Windows

Due to the acceleration and velocity limitations, the agent cannot move as required. Meanwhile, the high rate iteration of the algorithm or obstacles may cause the agent speed to be extremely slow, which may cause the speed group to be ignored and run errors. Therefore, two-level velocity windows are used to deal with this problem, as shown in Figure 3.

Figure 3a shows the two-level velocity windows. The black cube represents the agent and the blue cube represents the 'local window'. The local window is the reachable velocity by the agent in the next iteration, which is limited by the agent acceleration and maximum velocity. The light cube represents the window of the maximum speed that the agent can reach. The maximum speed window can ignore the kinematic constraints of the agent. In Figure 3b, according to the elevation and azimuth angles, the agent's motion is divided into different motion planes to form a 3D fan-shaped area. The dense part of the 3D sector represents that the agent can reach predicted positions if the agent maintains its velocity for 3 s. The dark-blue area represents the positions where the agent can reach at the maximum speed. The red rectangular prism represents the position of the agent selected by the evaluation function.



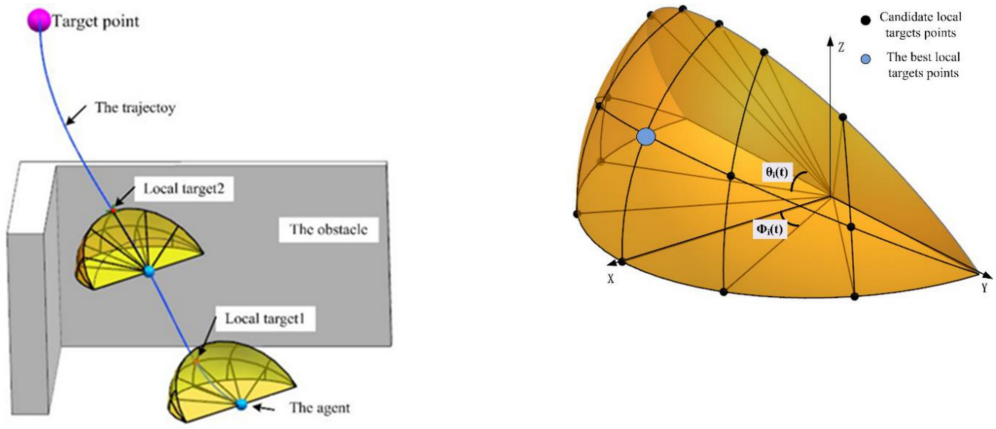
(**a**) Velocity space (**b**) Cartesian space

Figure 3. Two-level (**a**) velocity space and (**b**) Cartesian space.

By setting two-level velocity windows, when the agent iteration speed is extremely slow, the best trajectory between the maximum speed is first determined. Then the agent moves to near the optimal position in the next iteration. Based on this, we can further analyze how the agent determines the best local goal when it encounters a large obstacle or a U-shaped obstacle.

3.2. The Candidate Local Goal

As shown in Figure 4, the candidates for local goal points are generated in a safe fan-shaped area where the agent can fly, and the optimal value of the candidate local goal point is then detected.



(**a**) (**b**)

Figure 4. (**a**) Local goals in trajectory planning and (**b**) the position of the local goals.

The candidates for local goal points are defined by:

$$\boldsymbol{goal}_{local}(i) = (\boldsymbol{X}(t), \phi_i(t), \theta_i(t)) + R_s * (\boldsymbol{\eta}, j\frac{\pi}{mR_s}, k\frac{\pi}{mR_s}) \quad (9)$$

where $\boldsymbol{goal}_{local}(i)$ is the candidates for local points, $\phi_i(t)$ is the azimuth of the candidates for local points. $\theta_i(t)$ is the elevation of the candidates for local points, $X(t)$ is the position of the agent, and R_S is the safe separation of the agent and obstacles. j, k and m are the index and number of the candidate local goals. The larger the value of m is, the more local goal points can be selected, and the better the optimization result of the local goal can get. R_S is defined by:

$$R_s = R_r + r_0 + R_{braking}(t) \quad (10)$$

$$R_{braking}(t) = \frac{{v_t}^2}{a_{\max}} \tag{11}$$

$$\boldsymbol{\eta} = \begin{bmatrix} \cos(\phi_i)cos(\theta_i) \\ \sin(\phi_i)cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix} \tag{12}$$

where R_r is the agent radius, $R_{braking}$ is the braking distance of the agent, v_t is the current velocity of the agent, and $a_{\max}$ is the maximum acceleration that the agent can achieve.

3.2.1. Distance-to-the-Virtual Goal Term

This evaluation term is used to select the candidate virtual local goal point closest to the real goal.

$$dist1 = 1 - \frac{dist(\boldsymbol{P}_{goallocal} \rightarrow \boldsymbol{P}_{goal})}{d_{1\max}} \tag{13}$$

where $\boldsymbol{P}_{goal}$ is the position of the real goal, $\boldsymbol{P}_{goallocal}$ is the position of the virtual local goal, $d_{1\max}$ is the distance between the real goal and the current position of the agent.

$$R_{s1} = R_r + R_{braking}(t) \tag{14}$$

A safe area is established between the agent and the local target, and the safe area is defined as a cuboid area, as shown in Figure 5. If there exist obstacles in the safe area between the agent and the candidate local goals, the candidate local goal in the cuboid area will be deleted from $\boldsymbol{goal}_{local}(i)$.

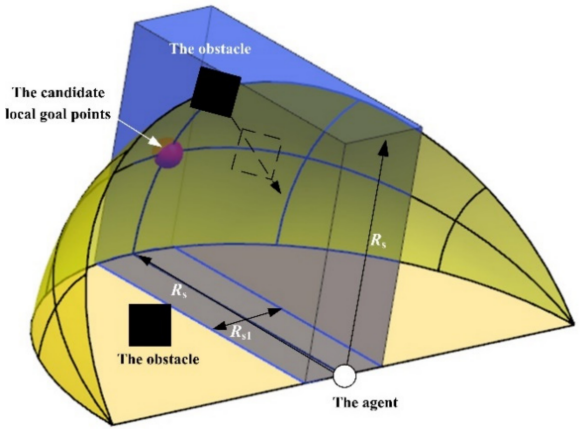


Figure 5. The selection of the candidate local goals.

3.2.2. Orientation-to-the-Local Goal Term

This evaluation term is used to select the candidate local goal point with the smallest angle between the real goal and the local goal points.

$$heading_{local} = 1 - \frac{arccos(\boldsymbol{P}_{goallocal} \rightarrow \boldsymbol{P}_{goal})}{\pi} \tag{15}$$

3.2.3. Distance-to-the-Obstacle Term

This evaluation term is used to select a local goal point far away from obstacles.

$$dist2 = \frac{dist(\boldsymbol{P}_{goallocal} \rightarrow \boldsymbol{P}_{obstacle})}{\sum_{i=1}^{m} dist(\boldsymbol{P}_{goallocal} \rightarrow \boldsymbol{P}_{obstacle})} \tag{16}$$

where $\boldsymbol{P}_{obstacle}$ is the position of obstacles detected by the agent, m is the number of obstacles.

The optimal local goal is selected by the evaluation function Equation (17):

$$g_{end} = \mu_{\text{dist}1} \bullet dist1 + \mu_{head1} \bullet heading_{local} + \mu_{\text{dist}2} \bullet dist2 \quad (17)$$

where μ_{dist1}, μ_{dist2} and μ_{head1} are the weights of the functions. The $goal_{local}$ (i) point corresponding to the larger set of results in g_{end} is used as the local goal.

3.3. The Evaluation Function for DF_DWA

Once the dynamic window is set, each set of speeds (linear and angular velocities) is evaluated to select the best combination in the iteration. This paper uses the speed, goal distance, goal orientation, and obstacle clearance as components of the evaluation function. The detailed content of the algorithm DF_DWA is given in Algorithm 1.

Algorithm 1. DF_DWA.

```
Procedure DF_DWA(goal, x_initial = [x0, y0, z0], parameter_initial, Kinematic, p_e, r0)
  i←1, ◊ iteration counter
    for I = 1:5000
      If norm(x(i)-goal)>p_e ◊ p_e is the allowable position error
        If norm(x(i)-obstacle(i))<= r0
        break;
        If (there are large obstacles or U-shaped obstacles) then
        g_end←(Equation (9), Equation (10), Equation (14), Equation (15), Equation (16));
        goal_r←goal && goal←V_goal;
          If norm(x(i)- goal)<= p_e;
          goal←goal_r;
          end if
          end if
    EvalParameter←evalfis([distance(x(i)-goal), distance(x(i)-obstacle(i))]; ◊ Calculate parameters by fuzzy rule.
    [V_min (i), V_max (i), ω_min (i), ω_max (i)]←DynamicWindow(x(i), Kinematic);
    f (V_(t), ω_(t),distgoal(t), distobstcles(t), heading(yaw(t), pitch(t)), Evalparameter)←Equation (23);
    ind(i)←max(f); ◊ Find the velocity group with the highest evaluation function value
    [u(i)]←Kinematic(ind,1:3); ◊ Finding the optimal speed group
    x(i + 1)←P(x,u); ◊ Iteration
  else If norm(x(i)-goal)<p_e
    break;
    end if
  end
end Procedure
```

3.3.1. Speed Term

The speed term is used to evaluate whether the agent advances at the optimal speed. There are two scenarios in the process of an agent flying toward a goal. One is that the agent faces the goal while the other is that the agent faces away from the goal. If the goal is in front of the robot, the reward value of this term can be gotten as follow:

$$speed(v_t) = \frac{v_t}{v_{\max}} \quad (18)$$

When the agent faces away from the target, it only needs to rotate motion to save energy:

$$speed(v_t, \omega_t) = (\frac{v_t}{v_{\max}}) * \alpha + (\frac{\omega_t}{\omega_{\max}}) * \beta \quad (19)$$

where α and β are the proportional parameters of the linear velocity and angular velocity. The value of α and β is between 0 and 1. The angular velocity has two maximum values: (1) $v_t = 0$, $\omega_t = \omega_{\max}$; (2) $v_t = 0$, $\omega_t = -\omega_{\max}$. As the linear velocity increases, the rotation speed will decrease until $v_t = v_{\max}$, $\omega_t = 0$.

3.3.2. Distance-to-the-Goal Term

This term prizes the trajectory that makes the agent move towards the goal. The distance from the starting point to the goal is defined as the maximum distance $d_{\max}$. Also, the reward function is normalized between 0 and 1. Therefore,

$$distgoal(v_t, \omega_t) = 1 - \frac{d_{goal}(t)}{d_{\max}} \tag{20}$$

where d_{goal} is the distance from the position of the agent to the goal, and $d_{\max}$ is the distance from the starting point to the goal.

3.3.3. Orientation-to-the-Goal Term

This heading term prizes the curvature arcs that head the agent towards the goal. The direction angle between the end of the trajectory and the goal is compared with the azimuth of the agent (i.e., ϕ_{error}, θ_{error}). As shown in Figure 6, the ϕ_{error} and θ_{error} can be given by:

$$\phi_{error} = \phi' - \phi \tag{21}$$

$$\theta_{error} = \theta' - \theta \tag{22}$$

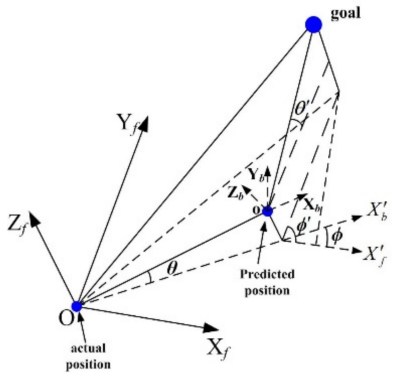


Figure 6. Angle θ_{error} and ϕ_{error} to the goal.

The reward function is given by 180-ϕ_{error}, 180-θ_{error} and normalized between 0 and 1.

$$heading_{pitch}(v_t, \omega_t) = 1 - \frac{\left|\arctan2(y_{goal} - y(t), x_{goal} - x(t)) - th_{pitch}(t)\right|}{\pi} \tag{23}$$

$$heading_{yaw}(v_t, \omega_t) = 1 - \left(\frac{\left|\arctan2(z_{goal} - z(t), sqrt((y_{goal} - y(t))^2 + (x_{goal} - x(t))^2))\right|}{\pi} - \left|th_{yaw}(t)\right| / \pi\right) \tag{24}$$

where (x_{goal}, y_{goal}, z_{goal}) is the coordinate of the goal point and (th_{pitch}, th_{yaw}) is the attitude angle of the agent.

3.3.4. Distance-to-the-Obstacle Term

This term prizes that the agent travels far from the obstacles. The distance to obstacles includes the distance to static obstacles and the predicted distance to moving obstacles. If the value of this term is small, the agent will be close to obstacles. In contrast, if the value of this term is big, the agent will be far from obstacles.

$$dist_{obstacle}(i) = norm(x(t) - x_{obstacle}(t), y(t) - y_{obstacle}(t), z(t) - z_{obstacle}(t)) - r_0 \tag{25}$$

$$dist_{obstacle} = dist_{static} + dist_{dynamic} - r_0 \tag{26}$$

where ($x_{obstacle}$, $y_{obstacle}$, $z_{obstaclel}$) are the coordinates of obstacles. Distance to obstacles includes distance to static obstacles ($dist_{static}$) and distance to dynamic obstacles ($dist_{dynamic}$).

For the dynamic obstacle, the position and velocity direction of the dynamic obstacle are measured by the sensor in real time, and then the position that the dynamic obstacle can reach in each time interval (Δt) in the next prediction time is calculated as the predicted position of the dynamic obstacle. Then, among these positions, the shortest distance to the agent is used as the dynamic obstacle distance ($dist_{dynamic}$), as shown in Figure 7. By considering this distance, the agent plans the next motion command. In actual operation, the polynomial fitting algorithm [19] and extended Kalman filtering method [33] can be used to predict the trajectory of obstacles. In addition, choosing the shortest distance as the distance of the dynamic obstacles increases the threat posed by the dynamic obstacle to the agent. Therefore, fuzzy rules can increase the weight of distance to the obstacle item and the agent can avoid obstacle as quickly as possible.

$$dist_{dynamic} = \min\left(norm\left(x(t) - x_{preob}(t_i), y(t) - y_{preob}(t_i), z(t) - z_{preob}(t_i)\right)\right) \tag{27}$$

where (x_{preob}, y_{preob}, z_{preob}) are the predicted position of the dynamic obstacle.

$$dist_{obstacle}(i) = \frac{dist_{obstacle}(i)}{\sum_{i=1}^{n} dist_{obstacle}(i)} \tag{28}$$

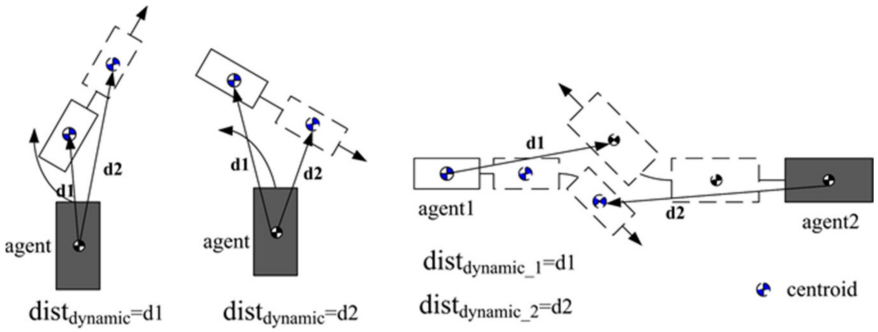


Figure 7. Perception prediction planning architecture.

The reward value of the iteration is calculated using Equation (29). Among all the speed groups being evaluated, the largest speed pair among the evaluation function values is selected as the motion instruction.

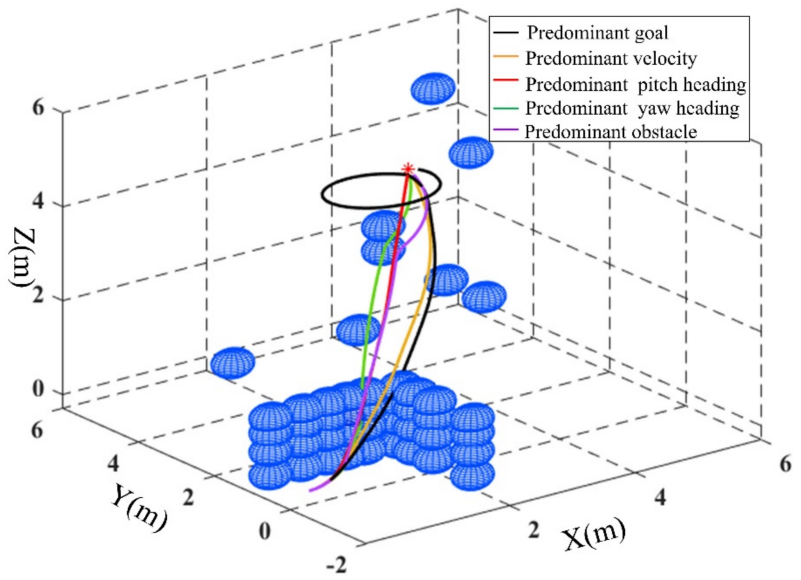
$$\begin{aligned} f(v_t, \omega_t) \quad &= \varepsilon_{speed} \bullet speed(v_t, \omega_t) + \varepsilon_{goal} \bullet distgoal(v_t, \omega_t) \\ &+ \varepsilon_{pitchheading} \bullet heading_{pitch}(v_t, \omega_t) + \varepsilon_{yawheading} \bullet heading_{yaw}(v_t, \omega_t) \\ &+ \varepsilon_{obstacle} \bullet dist_{obstacle}(i) \end{aligned} \tag{29}$$

where ε_{speed} is the reward value of speed, ε_{goal} is the reward value of the distance from the candidate points to the goal. $\varepsilon_{obstacle}$ is the reward value of the distance from the agent to the obstacles, and ($\varepsilon_{pitchheading}$, $\varepsilon_{yawheading}$) is the heading reward value.

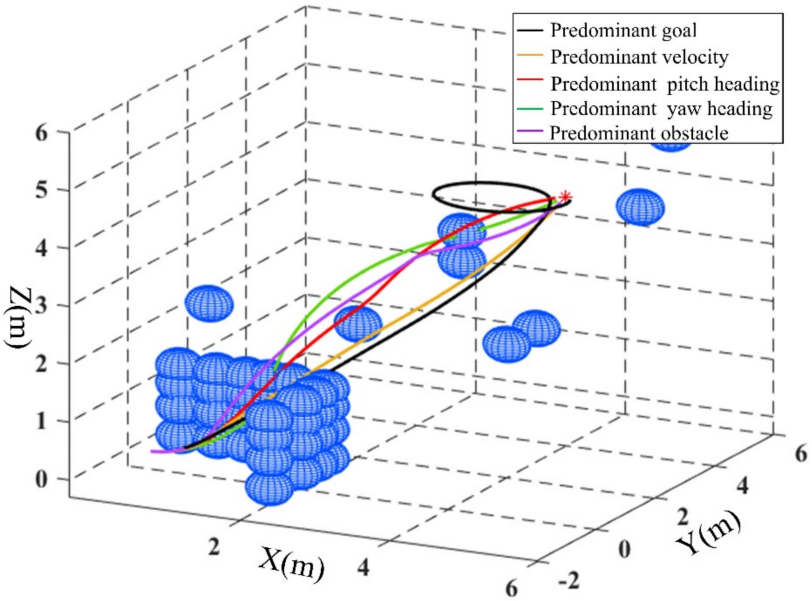
3.4. The Fuzzy Rule for the DF_DWA

According to the evaluation function (26), the agent selects the best speed group in the predicted trajectory as the action command. The reward value in the evaluation function determines the weight of each evaluation term in the process of predicting the trajectory. Therefore, with different weight settings, the selected trajectories are also different. In the next paragraphs, we will discuss the impact of the weight settings on trajectory generation and obstacle avoidance, in which each reward value of the evaluation function will be set to zero or predominant parameter.

To test the influence of the reward value weight in the evaluation function on the trajectory generation, we set scenarios as shown in Figures 8 and 9, and the test data is given in Table 1. The starting point of the agent is (0,0,0) and the goal point of the agent is (4,4,4). α is the number of iterations in each case. $\overline{v}$ and $\overline{\omega}$ are the average linear velocity and the average angular velocity of the agent, respectively. Meanwhile, the variance of linear velocity ($\sigma_v{}^2$) and angular velocity ($\sigma_\omega{}^2$) is used to evaluate the smoothness of the trajectory. In the test, the speed of the agent is limited to 2 m/s (linear velocity) and 60 degree/s (angular velocity). The acceleration is limited to 0.5 m/s^2 (linear velocity) and 90 degree/s^2 (angular velocity).



(**a**)



(**b**)

Figure 8. The path planning results with each reward value of the evaluation function being set to predominant weight: (**a**) front view (**b**) side view.

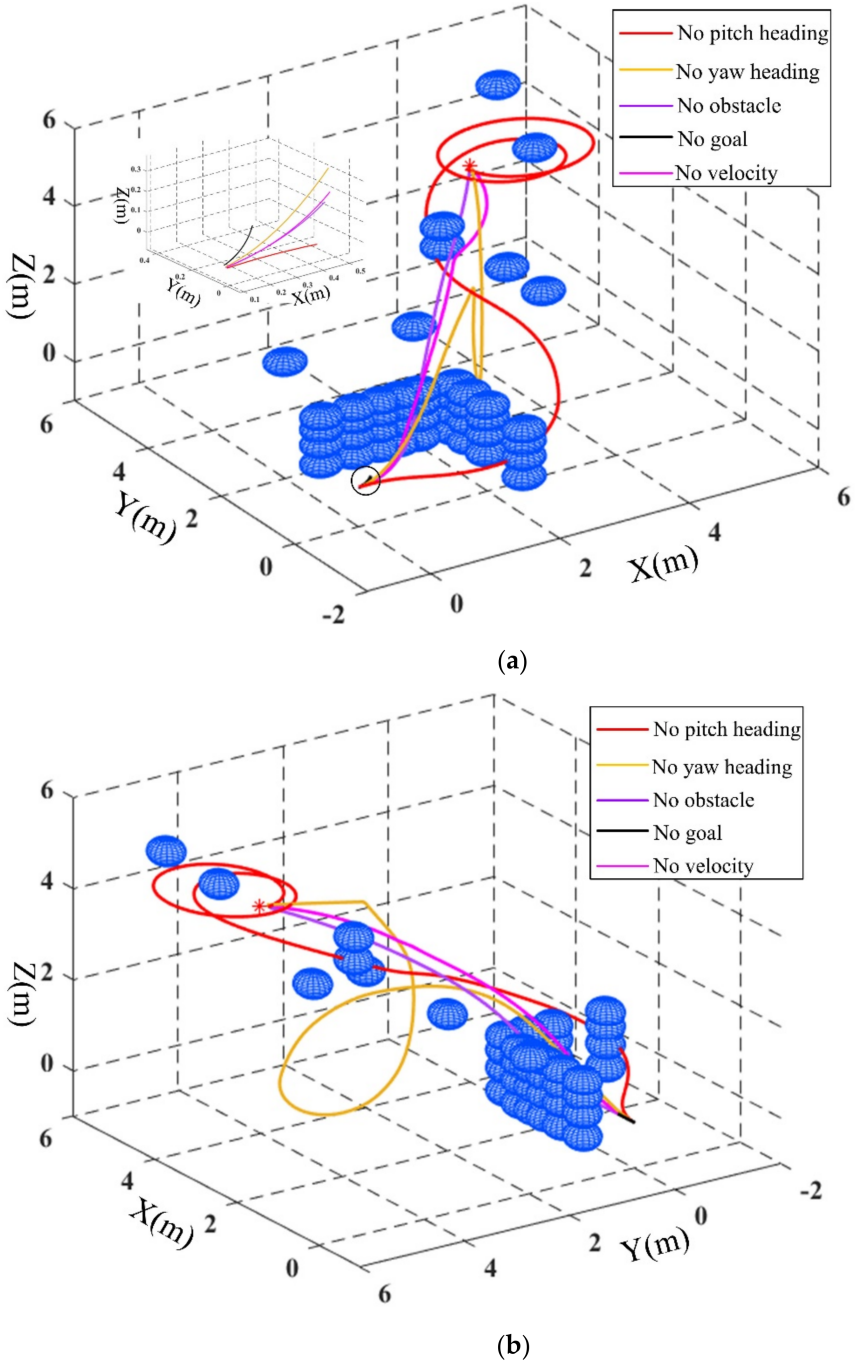


Figure 9. The path planning results with each reward value of the evaluation function being set to zero: (**a**) front view (**b**) side view.

It can be seen from Figure 8 that when $\varepsilon_{obstalce}$ is set as the predominant weight, the planned trajectory can avoid obstacles. If other terms are set as the predominant weight, the trajectory can cross through obstacles, resulting in failure of obstacle avoidance. Meanwhile, Figure 9 shows that if the value of $\varepsilon_{obstalce}$ is set to zero, the trajectory cannot avoid obstacles. When ε_{speed} is set as the predominant weight, it can be observed from Table 1 that the number of iterations is the least. If ε_{speed} is set to zero, although the agent can safely reach the target position, the number of iterations doubles. It can be observed from Figures 8 and 9 that the same situation occurs when $\varepsilon_{heading}$ is set to zero or ε_{goal} is set as the predominant weight. In other words, the trajectory reaches the goal after a full circle, which increases the number of iterations and causes the trajectory to be not smooth enough.

It can also be analyzed from Figure 9 that when there is no $\varepsilon_{pitchheading}$, the trajectory has a horizontal rotation, and when $\varepsilon_{yawheading}$ is absent, there is a vertical rotation. Therefore, the weight of these two evaluation items should not deviate too much when being set. In addition, Figure 9a shows that if ε_{goal} is zero, the agent will fall into local minima.

Table 1. The results of the parameter test.

Configuration	α	$\overline{v}$	$\overline{\omega}_{pitch}$	$\overline{\omega}_{yaw}$	σ_v^2	$\sigma_{\omega_{pithch}}^2$	$\sigma_{\omega_{yaw}}^2$
No $\varepsilon_{pitchheading}$	282	0.7204	0.0034	−0.386	0.0091	0.0029	02630
Predominant $\varepsilon_{pitchheading}$	197	0.3621	0.0039	0.0495	0.056	0.0231	0.0040
No $\varepsilon_{yawheading}$	219	0.7165	−0.1571	0.0515	0.0107	0.2323	0.0048
Predominant $\varepsilon_{yawheading}$	462	0.16	0.006	0.0244	0.0258	0.0123	0.0073
No ε_{goal}	100	0.0318	0.0602	0.0779	7.12×10^{-4}	0.003	0.0047
Predominant ε_{goal}	354	0.6429	0.0282	0.4408	0.0282	0.0240	0.1436
No $\varepsilon_{obstacle}$	234	0.3038	0.0142	0.0417	0.0372	0.0147	0.0048
Predominant $\varepsilon_{obstacle}$	187	0.2055	0.0095	0.0415	0.0438	0.0109	0.0093
No ε_{speed}	340	0.2139	0.0097	0.0429	0.0446	0.0114	0.0091
Predominant ε_{speed}	169	0.4195	0.0293	0.0817	0.0723	0.0178	0.0039

The above analysis shows that, in the evaluation function, a single or constant combination of reward values cannot guarantee the safety of the agent's trajectory. When the environment changes, the agent cannot make changes to adapt to the environment. Therefore, based on the traditional DWA, this paper introduces fuzzy rules to adjust the reward value.

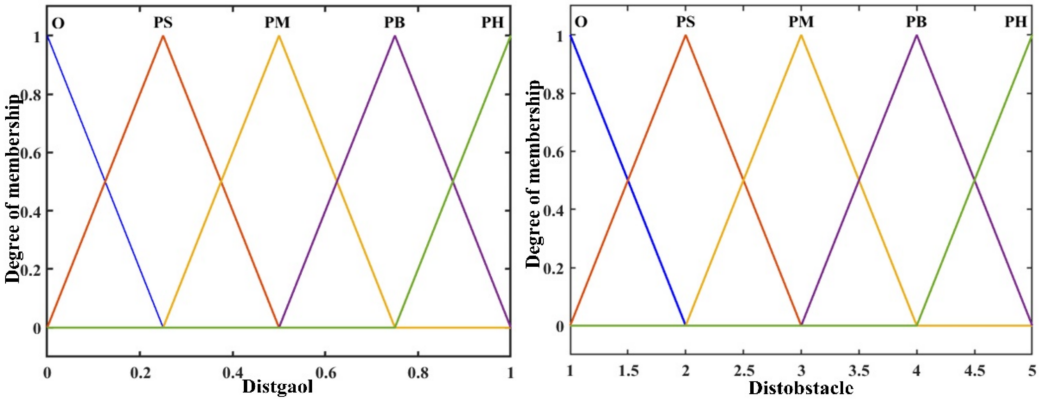
When setting a fuzzy rule to adjust the weights of parameters, the distance to the obstacle and the distance to the target are used as the inputs of the fuzzy rule, and the evaluation coefficients of the speed, distance to the goal, distance to the obstacle, and the direction to the goal are used as outputs. The fuzzy rule is shown in Figures 10 and 11.

(1) When the distance to the goal and the distance to the obstacles are long (i.e., longer than 5 agent radii), the agent does not need to avoid obstacles first. The agent should accelerate to a certain speed and approach the goal with moderate attention. $\varepsilon_{\text{obstalce}}$ can therefore be set at a small value. The value of ε_{goal} can be increased moderately. $\varepsilon_{heading}$ can be set at a large value.

(2) When the distance to the goal is long and the distance to obstacles is short (i.e., shorter than 2 agent radii), the agent should avoid obstacles first and not rush to approach the goal. The weight of the agent speed should be reduced to guarantee that the agent has enough time to adjust the forward direction. Therefore, $\varepsilon_{obstalce}$ should be set at a large value while ε_{goal}, $\varepsilon_{heading}$, and ε_{speed} should be set at a small value.

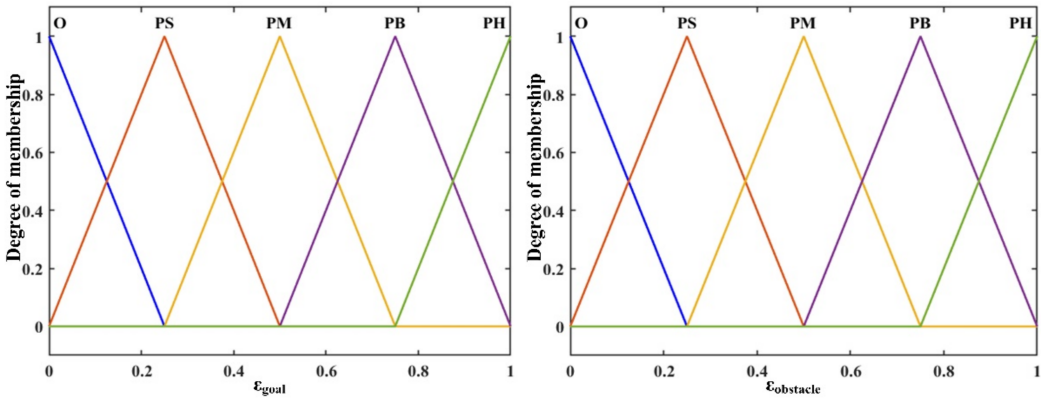
(3) When the distance to the goal is short and the distance to obstacles is long, the agent should approach the goal first and not rush to avoid obstacles. The weight of the agent speed should be appropriately reduced to ensure that it reaches the goal safely. $\varepsilon_{obstalce}$ and ε_{speed} can therefore be set at a small value, while ε_{goal} being set as the predominant weight and $\varepsilon_{heading}$ being set at a medium value.

(4) When the distance to the goal and the distance to obstacles are short, the agent should avoid obstacles first and approach the goal with moderate attention. The weight of the agent's speed should be reduced. ε_{speed} and $\varepsilon_{heading}$ should therefore be set at a small value. $\varepsilon_{obstalce}$ should be set at the predominant weight. ε_{goal} should be set at a medium value.

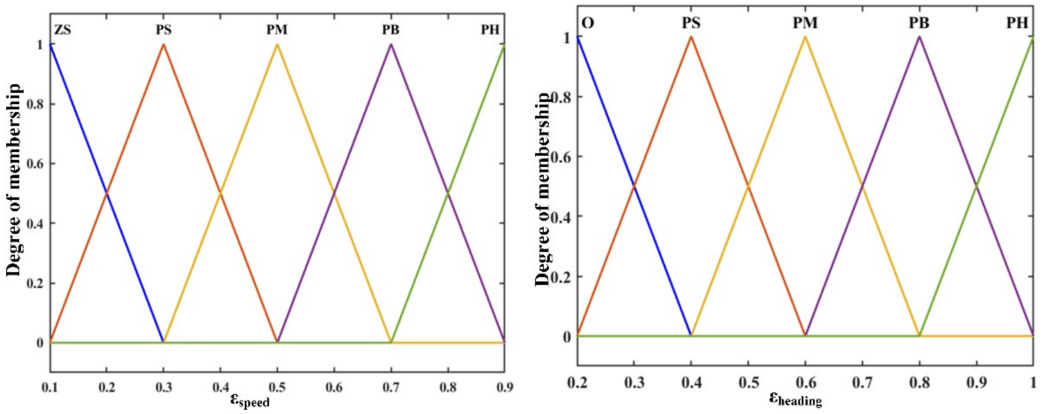


(**a**) Distgoal fuzzy membership function (**b**) Distobstacle fuzzy membership function

Figure 10. Fuzzy input membership functions.



(**a**) ε_{goal} fuzzy output membership function (**b**) $\varepsilon_{obstacle}$ fuzzy output membership function



(**c**) ε_{speed} fuzzy output membership function (**d**) $\varepsilon_{heading}$ fuzzy output membership function

Figure 11. Fuzzy output membership functions.

The fuzzy rules Distgoal, Diatobstacle and ε_{goal}, $\varepsilon_{obstalce}$, ε_{speed}, $\varepsilon_{heading}$ are shown in Tables 2–5.

Table 2. Fuzzy control rules of Distgoal, Distobstacle, and ε_{goal}.

	Distobstacle				
Distgoal	**ZS**	**PS**	**PM**	**PB**	**PH**
Z	PM	PM	PM	PB	PH
PS	PS	PS	PM	PB	PH
PM	Z	PS	PM	PB	PB
PB	Z	Z	PS	PM	PB
PH	Z	Z	Z	PS	PM

Table 3. Fuzzy control rules of Distgoal, Distobstacle, and $\varepsilon_{obstacle}$.

	Distobstacle				
Distgoal	**ZS**	**PS**	**PM**	**PB**	**PH**
Z	PB	PB	PM	PS	Z
PS	PH	PB	PM	PS	Z
PM	PH	PB	PM	PS	Z
PB	PH	PB	PM	PS	PS
PH	PH	PH	PB	PM	PS

Table 4. Fuzzy control rules of Distgoal, Distobstacle, and ε_{speed}.

	Distobstacle				
Distgoal	**ZS**	**PS**	**PM**	**PB**	**PH**
Z	ZS	ZS	PS	PM	PB
PS	ZS	ZS	PS	PM	PB
PM	ZS	ZS	PS	PM	PB
PB	ZS	PS	PM	PM	PB
PH	ZS	PS	PM	PB	PH

Table 5. Fuzzy control rules of Distgoal, Distobstacle, and $\varepsilon_{yawheading}$.

	Distobstacle				
Distgoal	**ZS**	**PS**	**PM**	**PB**	**PH**
Z	Z	Z	Z	PS	PS
PS	Z	Z	PS	PS	PS
PM	PS	PS	PS	PS	PS
PB	PS	PS	PM	PM	PB
PH	PS	PS	PM	PB	PB

where Z is zero; ZS is the zero small; PS is small; PM is the middle; PB is big; PH is the height big.

The Mamdani fuzzy inference algorithm is used while defuzzification adopts the center-of-gravity method. The evaluation function can be obtained:

$$\begin{aligned} f(v_t, \omega_t) = \; & \widehat{\varepsilon}_{speed} \bullet speed(v_t, \omega_t) + \widehat{\varepsilon}_{goal} \bullet distgoal(v_t, \omega_t) \\ & + \widehat{\varepsilon}_{pitchheading} \bullet heading_{pitch}(v_t, \omega_t) + \widehat{\varepsilon}_{yawheading} \bullet heading_{yaw}(v_t, \omega_t) \\ & + \widehat{\varepsilon}_{obstacle} \bullet dist_{obstacle}(i) \end{aligned} \tag{30}$$

Based on fuzzy rules, if the minimum value of the input obstacle distance is set larger, the agent's reaction time for obstacle avoidance will be earlier. Meanwhile, the minimum value of the ε_{speed} and $\varepsilon_{heading}$ fuzzy domain is not set to zero, which can prevent the agent from falling into local minima because of the slow velocity. The specific steps of DF_DWA algorithm are as follows.

4. Simulation Results

In this section, the simulation results will be discussed. The simulations were run with 64 bit MATLAB R2018a on the Intel Core I7-9750H, 2.6GHz processor. The parameters used in the numerical simulations are listed in Table 6.

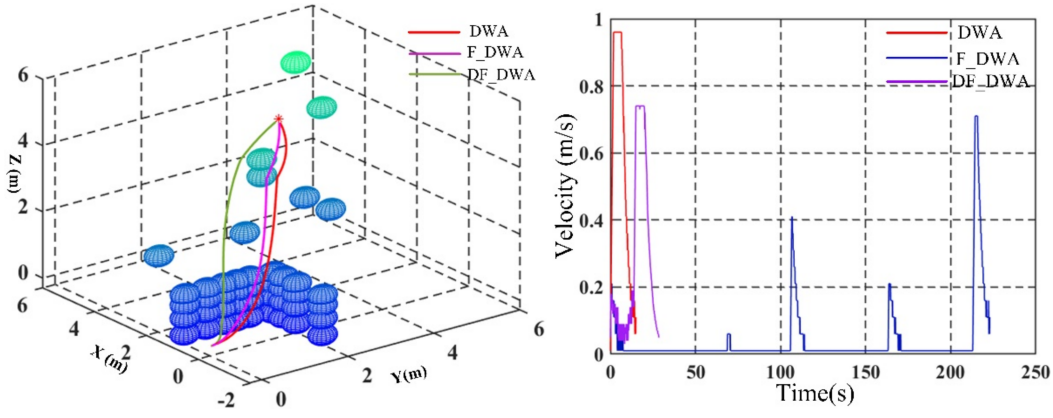
Table 6. Agent and obstacles parameters.

Parameters	Value
The limited linear velocity (m/s)	2
The minimum linear velocity (m/s)	0
The limited angular velocity (degree/s)	60
The limited linear acceleration (m/s^2)	0.5
The limited angular acceleration (degree/s^2)	90
The radius of the agent R_{agent} (m)	0.15
The radius of the obstacles $R_{obstacle}$ (m)	0.15
The safe distance (m)	0.3
The allowable position error: p_e (m)	0.1
Prediction time (s)	3

4.1. Static Obstacle Avoidance

In this scenario, there are multiple static obstacles in the space. Meanwhile, an L-shaped obstacle with a height of 2.4 m and a length of 1.8m is added to the obstacles. We have discussed the performance of the planned trajectory when the reward value takes different weights in the second section. In this test, the fuzzy logic is introduced to adjust the weight of the parameters.

Figure 12a shows the trajectories planned by the three methods. Figure 12b shows the velocity change of the agent. Table 7 shows the number of iterations and the minimum distance between the agent and obstacles. In terms of the time spent in planning the trajectory, it can be seen from Figure 12b and Table 7 that it takes the most time when using F_DWA to plan trajectory, and DWA spends the least time in planning trajectory. Regarding the safety factor, it can be seen that the trajectory planned by DF_DWA is the farthest from obstacles, so the trajectory is the safest. The minimum distance between the trajectory planned by DWA and obstacles is 0.28mm, less than the safety distance of 0.3 mm, meaning that although agents can reach the goal with the least time when following the DWA planned trajectory, the safety is lowered. F_DWA takes the most time to plan the trajectory because of the velocity falling into the local minima many times. Therefore, DF_DWA is superior to F_DWA and DWA in terms of computational efficiency and safety.



(**a**) (**b**)

Figure 12. (**a**) planning trajectory and (**b**) velocity in the static obstacle avoidance scenario.

Table 7. The iteration time and minimum distance.

Method	Iteration Time	Minimum Distance (mm)
DWA	179	0.28
F_DWA	2230	0.35
DF_DWA	289	0.4

It can be seen from the simulation results that compared with the algorithms DW, BUG, and PF in the literature [19], DF_DWA can ensure that the agent's trajectory is safer when escaping the local minima. Compared with the hybrid dynamic window method [18], DF_DWA can make agent keep a safer distance from obstacles when escaping U-shaped traps. Compared with DW4DO [15], DF_WDA can adjust the parameter weight of the optimization function according to the changes in the obstacle scene.

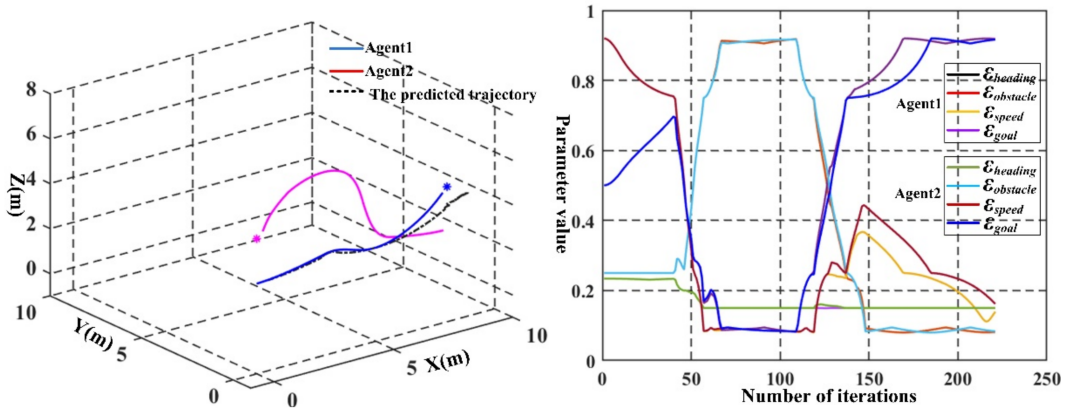
4.2. Dynamic Obstacle Avoidance

In space activities, multiple agents are sometimes required to complete tasks at the same time. For one agent, the other agents are regarded as dynamic obstacles. Each agent is equipped with sensors that can perceive the external environment around it. Therefore, for abnormal situations, if an agent is chasing the agent ahead, it can be perceived by the agent ahead.

In this section, the agent can detect the location of other agents by sensors and calculate the distance between them [19,33,42]. When using DF_DWA, although the speed does not change much, it is not constant. Therefore, the safety distance between obstacles must be strictly ensured. In this paper, the current maximum window speed is used to calculate the braking distance (i.e., $R_{braking}$).

(1) Two agents

To further test the obstacle avoidance ability of the algorithm in a dynamic environment, we set up a scenario in which two agents move to each other and are obstacles to each other. As shown in Figure 13a, the starting point of Agent 1 is (0,0,3). The goal point of Agent 1 is (8,0.5,5). The starting point of Agent 2 is (8,0,3). The goal point of Agent 2 is (0,0.7,5). It is observed from the test result that this situation is safe in that Agent 1 moves upward and Agent 2 moves downward when the separation of the two agents is short. The black dotted line is the predicted trajectory of Agent 2 to Agent 1.



(**a**) (**b**)

Figure 13. (**a**) planning trajectory and (**b**) the changes in the weight of reward value during iterations in two dynamic obstacle avoidance scenarios.

It can be seen from Figure 13b that the parameter ε_{speed} is set as a predominant weight when the two agents are far apart initially. As the two agents approach each other, the weight of the $\varepsilon_{obstalce}$ gradually increases. When the agents reach the obstacle avoidance

range, the weight of the $\varepsilon_{obstalce}$ exceeds 0.9. After escaping the threat posed by one another, the first task of each agent is to reach its goal. Therefore, ε_{goal} gradually increases. When the goal is about to be reached, the weight of the agent speed should be decreased to prevent the agent from passing over the goal while traveling at an extremely high speed.

(2) Four agents

In the previous section, when two agents move towards each other, each agent has sufficient space for movement. However, when multiple agents fly close to each other, the space to avoid obstacles will be shrunk. To verify the obstacle avoidance ability of the agent in this situation, two scenarios are set up in this section: one is that four agents fly to four locations within the same plane that are close to each other (FFLSP), and the other is that four agents cross each other and fly to the designated locations (FFCL).

In the FFLSP scenario, four agents fly to different positions at the same height to complete tasks similarly to formation or assembly. The starting point of Agent 1 is (0,2,0). The goal point of Agent 1 is (5,4,5). The starting point of Agent 2 is (0,0,0). The goal point of Agent 2 is (5,6,5). The starting point of Agent 3 is (4,2,0). The goal point of Agent 3 is (3,4,5). The starting point of Agent 4 is (4,0,0). The goal point of Agent 4 is (3,6,5). As shown in Figure 14a, the agents safely reach their goal positions. Meanwhile, Figure 14c shows the minimum distance between agents, that is, the smallest value in the set of distances between each agent and other agents at the same moment during the flight. It can be seen that the minimum distance of Agent 1 and Agent 2 is similar before the 95th iteration. This result shows that Agent 1 and Agent 2 are the main obstacles to each other. When running the 78th iteration, the minimum safety distance is 0.5324 m, which is greater than the safety distance of 0.3 m.

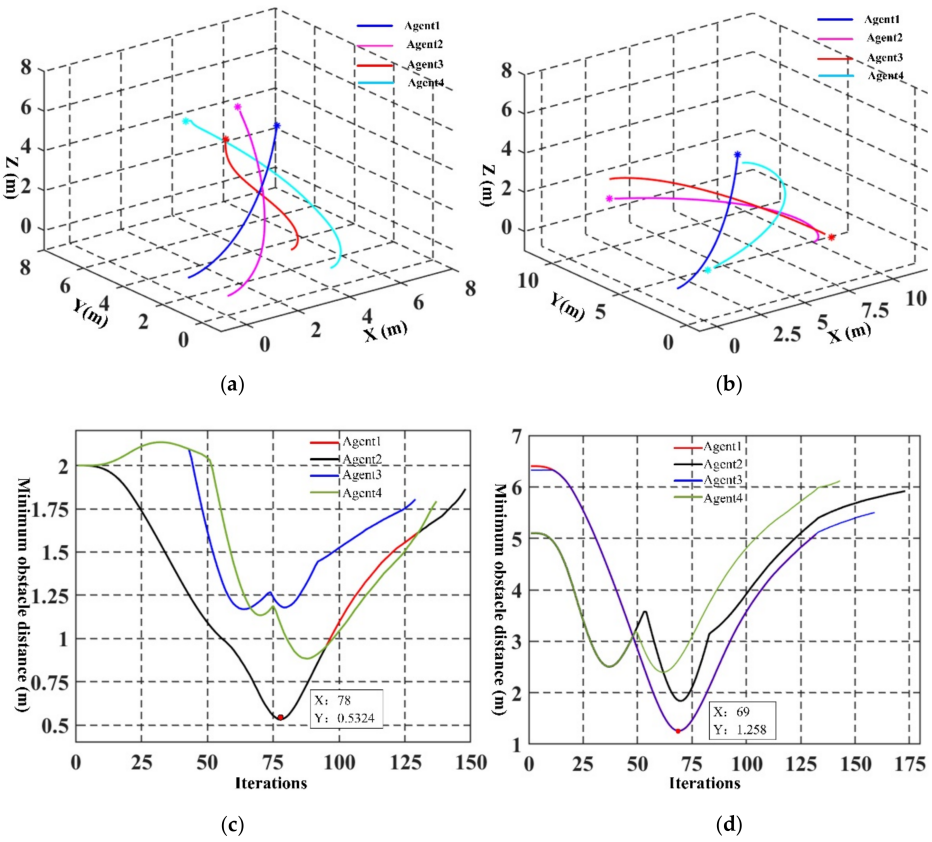
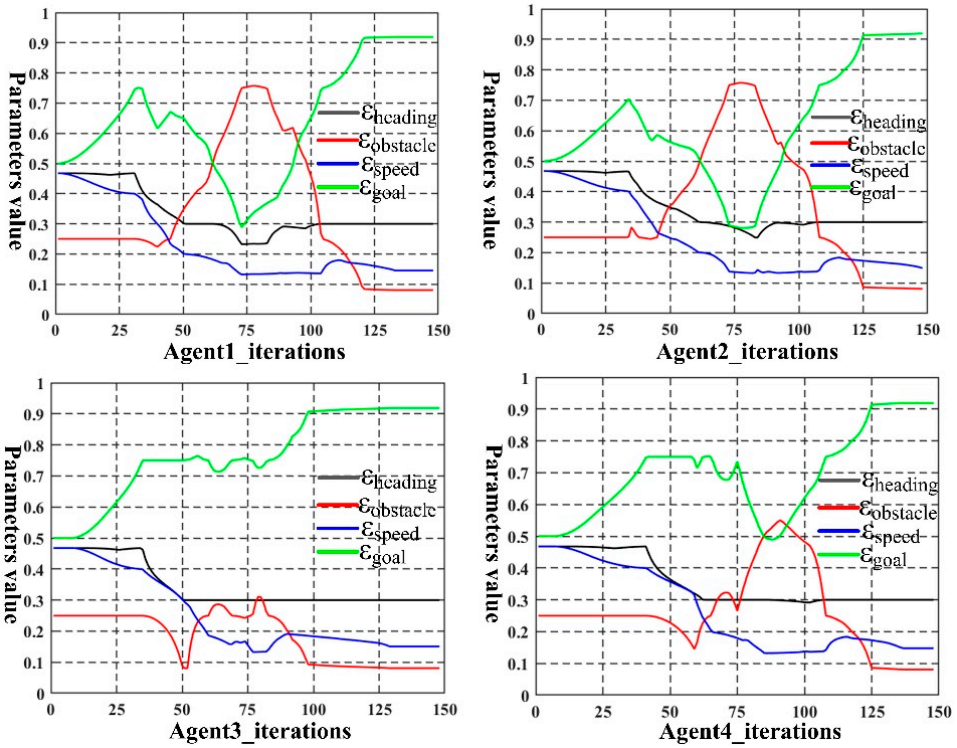


Figure 14. Dynamic obstacle avoidance: ((**a**) four agents flying to the same plane; (**b**) four agents crossing each other; (**c**) the minimum obstacle distance between the agents at the same moment during the iterations in FFLSP scenario; (**d**) the minimum obstacle distance between the agents at the same moment during the iterations in FFCL scenario).

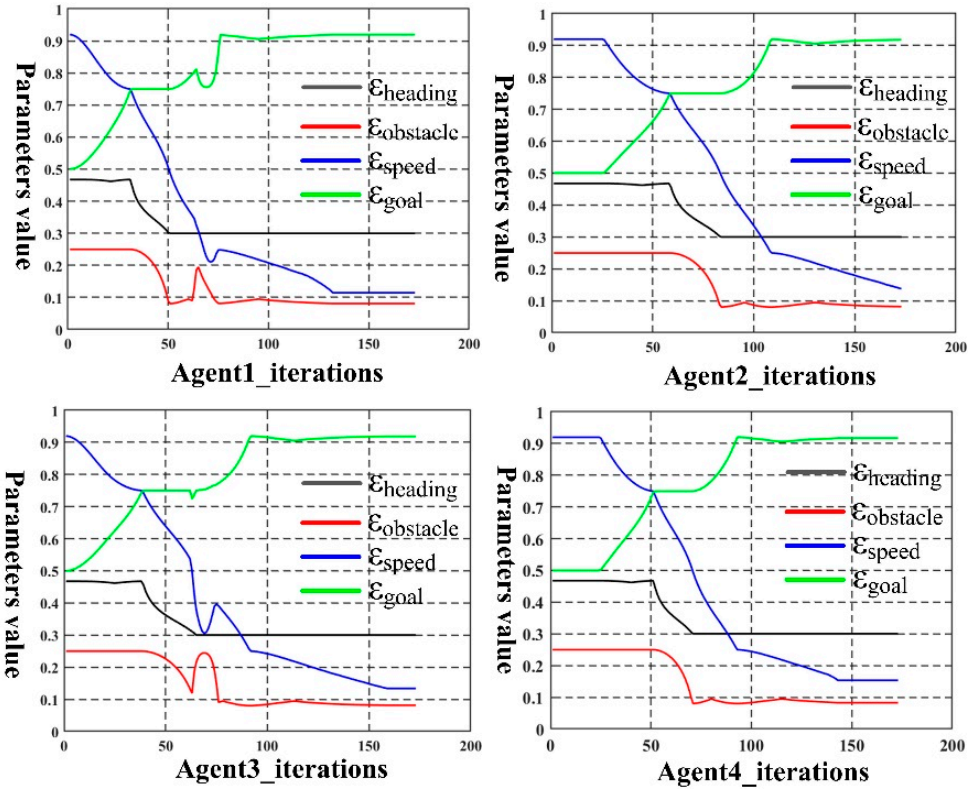
In the FFCL scenario, four agents fly from four diagonal directions. This setting is used to simulate multiple agents reaching their positions at the same time to carry out their operational tasks, respectively. During the process, the agents pass through similar path points. The four agents are unknown dynamic obstacles to each other. The starting position of Agent 1 are (7,1,1). The goal point of Agent 1 is (8,1,1). The starting position of Agent 2 is (0,2,0). The goal point of Agent 2 is (5,4,5). The starting position of Agent 3 is (0,7,4). The goal position of Agent 3 is (8,1,1). The starting position of Agent 4 is (6,5,4). The goal position of Agent 4 is (1,1,1). It can be seen from Figure 14b that the agents safely reach their goal points, respectively. Figure 14d shows the results of the minimum distance between agents at the same moment. At the 69th iteration, the minimum distance between Agent 1 and Agent 3 is 1.258 m, which is far greater than the safety distance of 0.3m.

The analysis of the minimum distance shows that the planned trajectory is safe and reliable. Figure 15 shows the weight of reward value changes in FFLSP and FFCL scenarios. In FLSP scenario, when an agent starts moving, the distance to its goal, and the distance to the obstacles are relatively long. Therefore, in Figure 12a, the weight of ε_{goal} is set as the predominant weight at the beginning. As the agents approach the goal point, the weight of $\varepsilon_{obstacle}$ of the Agent 1 and Agent 2 gradually increases before the 75th iteration, and the weight of $\varepsilon_{obstacle}$ is set as the predominant weight until the 95th iteration. Corresponding to Figure 14c, the minimum obstacle distance of Agent 1 and Agent 2 is also in this iteration interval, so the agents avoid obstacles first. Then, when Agents 1 and 2 escape the collision threat, the weight of $\varepsilon_{obstacle}$ gradually decreases, whereas the weight of ε_{goal} gradually increases. For Agent 3, since the weight of ε_{goal} is always set as the predominant weight, it is more likely that the Agent 3 will not be collided during operation. As for Agent 4, the weight of $\varepsilon_{obstacle}$ gradually increases since the 59th iteration, and reaches the maximum at the 91st iteration. It is noticeable from Figure 14c that Agent 4 reaches the smallest collision position at the 91st iteration. The agent needs to avoid obstacles first, and then fly to the goal, so the weight of $\varepsilon_{obstacle}$ is the maximum.

It can be seen from Figure 14d that as the minimum obstacle distance is far longer than the safe distance of 0.3m, the distance between the agents is safe. Therefore, in Figure 15b, the weight of the reward value slightly fluctuates. In the process of approaching the goal, the weight of the ε_{speed} and $\varepsilon_{obstacle}$ gradually decreases, while the weight of the ε_{goal} gradually increases. When approaching the goal, the weight of the ε_{speed} and $\varepsilon_{obstacle}$ reaches the minimum, but the weight of the ε_{goal} reaches the maximum. The analysis of the parameter changes shows that DF_DWA is effective, because the weight of the reward value can be adjusted with the change of the environment, thereby ensuring the safety of the agent's trajectory. Compared with DW4DO [15], DF_DWA can ensure the smoothness of operation by moving obstacles in advance. Compared with ASTRO [33], it reduces the mathematical constraints between two motion agents, thus making it more conducive for multiple agents to operate together and complete space tasks through data sharing. Table 8 shows all the numerical results of the dynamic obstacle avoidance test. It is obvious that the planned trajectory is smooth, and it is suitable for application in the navigation framework.



(**a**)



(**b**)

Figure 15. The changes in the weight of the reward value during iterations in (**a**) FFLSP and (**b**) FFCL scenarios.

Table 8. Dynamic obstacle avoidance: numerical results.

Experiment	Agents	α	$\overline{v}$	$\overline{\omega}_{pitch}$	$\overline{\omega}_{yaw}$	σ_v^2	$\sigma_{\omega_{pithch}}^2$	$\sigma_{\omega_{yaw}}^2$
Two agents	Agent1	222	0.4045	0.0223	−0.066	0.0535	0.0136	0.0251
	Agent2	222	0.4824	−0.0516	0.0085	5.37×10^{-4}	0.0034	0.0487
FFLSP	Agent1	133	0.5476	0.1548	0.0395	0.0029	0.1059	0.033
	Agent2	148	0.6463	0.0709	0.0828	0.0073	0.1152	0.021
	Agent3	129	0.5007	0.1288	0.2764	0.0578	0.1003	0.0099
	Agent4	137	0.6686	0.1802	0.1684	0.0269	0.0924	0.1031
FFCL	Agent1	132	0.5214	−0.0114	0.1871	0.0615	0.0937	0.0373
	Agent2	173	0.6789	0.0139	0.1799	0.0411	0.1245	8.8×10^{-4}
	Agent3	159	0.6081	−0.023	0.0534	0.0036	0.1388	0.0012
	Agent4	143	0.6	−0.1670	−0.205	0.0439	0.1085	0.0746

5. Conclusions

This paper proposes a three-dimensional obstacle-avoidance approach for dynamic environments based on DWA. The weight of each parameter in the evaluation function adjusts in real-time in different working environments to ensure an agent selects a safe trajectory. By the simulation test, the approach responds well to large obstacles and dynamic obstacles and plans a safe trajectory. In addition, the numerical results show that the planned trajectory is smooth and is applicable in the navigation framework. Compared with other methods, the developed DF_DWA does not require complicated constraint formulas. Also, it reduces calculation requirements and increases the ability to respond to environmental changes. In future work, we will continue to study obstacle avoidance algorithms for dynamic obstacles and improve experimental conditions.

Author Contributions: Conceptualization, C.X.; methodology, C.X.; validation, C.X.; formal analysis, C.X.; investigation, C.X.; data curation, C.X.; writing—original draft preparation, C.X.; writing—review and editing, M.X.; funding acquisition, Z.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by the National Science Foundation of China (No.11672290, No.11972343, No. 91848202).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: This work is supported in part by the National Science Foundation of China (No.11672290, No.11972343, No. 91848202).

Conflicts of Interest: No conflict of interest exists in the submission of this manuscript, and the manuscript is approved by all authors for publication.

References

1. Rybus, T. Obstacle avoidance in space robotics: Review of major challenges and proposed solutions. *Prog. Aerosp. Sci.* **2018**, *101*, 31–48. [CrossRef]
2. Seder, M.; Petrović, I. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1986–1991. [CrossRef]
3. Szczerba, R.J.; Galkowski, P.; Glickstein, I.S.; Ternullo, N. Robust algorithm for real-time route planning. *IEEE Trans. Aerosp. Electron. Syst.* **2000**, *36*, 869–878. [CrossRef]
4. Stentz, A. Optimal and efficient path planning for partially-known environments. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; pp. 3310–3317. [CrossRef]
5. Stentz, A. The Focussed D* Algorithm for Real-Time Replanning. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI'95, San Francisco, CA, USA, 1995; Volume 2, pp. 1652–1659.

6. Bruce, J.; Veloso, M.M. Real-time randomized path planning for robot navigation. In *Robot Soccer World Cup*; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2752, pp. 288–295. [CrossRef]
7. Lumelsky, V.J.; Stepanov, A.A. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica* **1987**, *2*, 403–430. [CrossRef]
8. Kamon, I.; Rivlin, E.; Rimon, E. New range-sensor based globally convergent navigation algorithm for mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, 22–28 April 1996; Volume 1, pp. 429–435. [CrossRef]
9. Borenstein, J.; Koren, Y. The vector field histogram—Fast obstacle avoidance for mobile robots. *IEEE Trans. Robot. Autom.* **1991**, *7*, 278–288. [CrossRef]
10. Ulrich, I.; Borenstein, J. VFH+: Reliable obstacle avoidance for fast mobile robots. In Proceedings of the 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), Leuven, Belgium, 20–20 May 1998; Volume 2, pp. 1572–1577. [CrossRef]
11. Ulrich, I.; Borenstein, J. VFH*: Local obstacle avoidance with look-ahead verification. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; Volume 3, pp. 2505–2511. [CrossRef]
12. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robotics Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
13. Ozdemir, A.; Sezer, V. A hybrid obstacle avoidance method: Follow the gap with dynamic window approach. In Proceedings of the 2017 First IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, 10–12 April 2017; pp. 257–262. [CrossRef]
14. Simmons, R. The Curvature-velocity method for local obstacle avoidance. In Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, 22–28 April 1996; Volume 4, pp. 3375–3382. [CrossRef]
15. Molinos, E.J.; Llamazares, Á.; Ocaña, M. Dynamic window based approaches for avoiding obstacles in moving. *Rob. Auton. Syst.* **2019**, *118*, 112–130. [CrossRef]
16. Nak, Y.K.; Simmons, R.G. The lane-curvature method for local obstacle avoidance. In Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190), Victoria, BC, Canada, 17 October 2002; pp. 1615–1621. [CrossRef]
17. Fernández, J.L.; Sanz, R.; Benayas, J.A.; Diéguez, A.R. Improving collision avoidance for mobile robots in partially known environments: The beam curvature method. *Rob. Auton. Syst.* **2004**, *46*, 205–219. [CrossRef]
18. Moon, J.; Lee, B.Y.; Tahk, M.J. A hybrid dynamic window approach for collision avoidance of VTOL UAVs. *Int. J. Aeronaut. Sp. Sci.* **2018**, *19*, 889–903. [CrossRef]
19. Xinyi, Y.; Yichen, Z.; Liang, L.; Linlin, O. Dynamic window with virtual goal (DW-VG): A new reactive obstacle avoidance approach based on motion prediction. *Robotica* **2019**, *37*, 1438–1456. [CrossRef]
20. Balan, K.; Manuel, M.P.; Faied, M.; Krishnan, M.; Santora, M. A fuzzy based accessibility model for disaster environment. In Proceedings of the IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019; pp. 2304–2310. [CrossRef]
21. Mbede, J.B.; Melingui, A.; Zobo, B.E.; Merzouki, R.; Bouamama, B.O. zSlices based type-2 fuzzy motion control for autonomous robotino mobile robot. In Proceedings of the 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, Suzhou, China, 8–10 July 2012; pp. 63–68. [CrossRef]
22. Mohanty, P.K.; Parhi, D.R. Path generation and obstacle avoidance of an autonomous mobile robot using intelligent hybrid controller. In *International Conference on Swarm, Evolutionary, and Memetic Computing, Proceedings of the International Conference on Swarm, Evolutionary, and Memetic Computing, Maribor, Slovenia, 10–12 July 20109*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7677, pp. 240–247. [CrossRef]
23. Tarokh, M. Hybrid intelligent path planning for articulated rovers in rough terrain. *Fuzzy Sets Syst.* **2008**, *159*, 2927–2937. [CrossRef]
24. Chen, L.; Hu, X.; Tang, B.; Cheng, Y. Conditional DQN-based motion planning with fuzzy logic for autonomous driving. *IEEE Trans. Intell. Transp. Syst.* **2020**, 1–12. [CrossRef]
25. Upadhyay, S.; Ratnoo, A. Smooth path planning for unmanned aerial vehicles with airspace restrictions. *J. Guid. Control. Dyn.* **2017**, *40*, 1596–1612. [CrossRef]
26. Mattei, M.; Blasi, L. Smooth flight trajectory planning in the presence of no-fly zones and obstacles. *J. Guid. Control. Dyn.* **2010**, *33*, 454–462. [CrossRef]
27. Marchidan, A.; Bakolas, E. Collision avoidance for an unmanned aerial vehicle in the presence of static and moving obstacles. *J. Guid. Control. Dyn.* **2020**, *43*, 96–110. [CrossRef]
28. Hu, C.; Zhang, Z.; Yang, N.; Shin, H.S.; Tsourdos, A. Fuzzy multiobjective cooperative surveillance of multiple UAVs based on distributed predictive control for unknown ground moving target in urban environment. *Aerosp. Sci. Technol.* **2019**, *84*, 329–338. [CrossRef]
29. Allen, R.E. A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance. *Rob. Auton. Syst.* **2019**, *115*, 174–193. [CrossRef]

30. Zhen, Z.; Chen, Y.; Wen, L.; Han, B. An intelligent cooperative mission planning scheme of UAV swarm in uncertain dynamic environment. *Aerosp. Sci. Technol.* **2020**, *100*, 105826. [CrossRef]
31. Zhang, N.; Gai, W.; Zhong, M.; Zhang, J. A fast finite-time convergent guidance law with nonlinear disturbance observer for unmanned aerial vehicles collision avoidance. *Aerosp. Sci. Technol.* **2019**, *86*, 204–214. [CrossRef]
32. Rastgoftar, H.; Atkins, E.M. Safe multi-cluster UAV continuum deformation coordination. *Aerosp. Sci. Technol.* **2019**, *91*, 640–655. [CrossRef]
33. Chamitoff, G.E.; Saenz-Otero, A.; Katz, J.G.; Ulrich, S.; Morrell, B.J.; Gibbens, P.W. Real-time maneuver optimization of space-based robots in a dynamic environment: Theory and on-orbit experiments. *Acta Astronaut.* **2018**, *142*, 170–183. [CrossRef]
34. Hong, Z.; Chun-Long, S.; Zi-Jun, Z.; Wei, A.; De-Qiang, Z.; Jing-Jing, W. A modified dynamic window approach to obstacle avoidance combined with fuzzy logic. In Proceedings of the 2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Guiyang, China, 18–24 August 2015; pp. 523–526. [CrossRef]
35. Abubakr, O.A.; Jaradat, M.A.K.; Hafez, M.A. A reduced cascaded fuzzy logic controller for dynamic window weights optimization. In Proceedings of the 2018 11th International Symposium on Mechatronics and its Applications (ISMA), Sharjah, UAE, 4–6 March 2018; pp. 1–4. [CrossRef]
36. Faisal, M.; Hedjar, R.; Al Sulaiman, M.; Al-Mutib, K. Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment. *Int. J. Adv. Robot. Syst.* **2013**, *10*. [CrossRef]
37. Zavlangas, P.G.; Tzafestas, S.G. Industrial robot navigation and obstacle avoidance employing fuzzy logic. *J. Intell. Robot. Syst. Theory Appl.* **2000**, *27*, 85–97. [CrossRef]
38. Morrell, B.J.; Chamitoff, E.; Gibbens, P.W. Autonomous operation of multiple free-flying robots on the international space station. In Proceedings of the 25th AAS/AIAA Spaceflight Mechanics Conference, Williamsburg, VA, USA, 11–15 January 2015.
39. Gao, Q.; Liu, J.; Tian, T.; Li, Y. Free-flying dynamics and control of an astronaut assistant robot based on fuzzy sliding mode algorithm. *Acta Astronaut.* **2017**, *138*, 462–474. [CrossRef]
40. Yang, H.I.; Zhao, Y.J. Trajectory planning for autonomous aerospace vehicles amid known obstacles and conflicts. *J. Guid. Control. Dyn.* **2004**, *27*, 997–1008. [CrossRef]
41. Park, J.; Baek, H. Stereo vision based obstacle collision avoidance for a quadrotor using ellipsoidal bounding box and hierarchical clustering. *Aerosp. Sci. Technol.* **2020**, *103*, 105882. [CrossRef]
42. Llamazares, Á.; Ivan, V.; Molinos, E.; Ocaña, M.; Vijayakumar, S. Dynamic obstacle avoidance using Bayesian occupancy filter and approximate inference. *Sensors* **2013**, *13*, 2929–2944. [CrossRef]