applied sciences

MDPI

Article

Three-Dimensional Obstacle Avoidance Strategy for Fixed-Wing UAVs Based on Quaternion Method

Yue Qu and Wenjun Yi *

National Key Laboratory of Transient Physics, Nanjing University of Science and Technology, Nanjing 210094, China; quyue@njust.edu.cn
* Correspondence: wjy@mail.njust.edu.cn; Tel.: +86-18061657369

Featured Application: This work provides a generalization of the velocity obstacle (VO) collision avoidance strategy to nonlinear second-order underactuated systems in three-dimensional dynamic uncertain environments.

Abstract: This work provides a generalization of the three-dimensional velocity obstacle (VO) collision avoidance strategy for nonlinear second-order underactuated systems in three-dimensional dynamic uncertain environments. A hierarchical architecture is exploited to deal with conflicting multiple subtasks, which are defined as several rotations and are parameterized by quaternions. An improved VO method considering the kinodynamic constraints of a class of fixed-wing unmanned aerial vehicles (UAV) is proposed to implement the motion planning. The position error and velocity error can be mapped onto one desired axis so that, only relying on an engine, UAVs can achieve the goal of point tracking without collision. Additionally, the performance of the closed-loop system is demonstrated through a series of simulations performed in a three-dimensional manner.

Keywords: collision avoidance; second order; moving obstacles; kinodynamic constraints; hierarchical architecture

check for updates

Citation: Qu, Y.; Yi, W. Three-Dimensional Obstacle Avoidance Strategy for Fixed-Wing UAVs Based on Quaternion Method. *Appl. Sci.* **2022**, *12*, 955. https://doi.org/10.3390/app12030955

Academic Editors: Chengchun Zhang, Aihong Ji, Zirong Luo and Gang Chen

Received: 3 December 2021
Accepted: 13 January 2022
Published: 18 January 2022

1. Introduction

In recent years, fixed-wing UAVs have gained growing applications in the civilian field as well as military fields, such as search, mapping, rescue, surveillance, and patrol. These applications commonly require the UAV to autonomously track predefined waypoints or prescribed trajectories. In addition, the autonomous detection of potential threats and an online obstacle avoidance algorithm are necessary for UAVs to ensure adequate security [1]. Accordingly, the control problem of UAVs is a multi-objective issue, which indicates that it is extremely important to introduce an integrated approach, taking into account global navigation as well as the local potential obstacle avoidance simultaneously. However, fixed-wing UAVs are typical underactuated systems with nonholonomic constraints, and it is still a challenging problem to develop a guidance framework when facing such a system.

Another important case is that UAVs need to perform multiple tasks, especially as there exist conflicts in sub-tasks, which motivates us to carry out our research. To this end, we introduce a hierarchical architecture [2–4] to cope with an arbitrary number of subtasks.

1.1. Related Works

Substantial research on motion planning for UAVs has been conducted, including sampling-based methods [5] and search-based methods [6]. For UAVs, the methods mentioned above are not directly applicable since they heavily rely on target prior information and large-scale iteration.

In view of the aforementioned considerations, reactive collision avoidance methods have been utilized to tackle the motion planning problem of systems with limited computing resources. The VO method was first proposed in [7], and it was originally developed for moving obstacles, considering the velocity of each obstacle in the environment. Since the VO method computes the avoidance velocity reactively based on the instantaneous geometric relationship of collisions, target prior information becomes needless. Compared with other reactive collision avoidance methods, such as artificial potential field (APF) methods [8–10], the VO method guarantees a lower computational cost and is less prone to problems with local minimums. Refs. [11–13] extend VO methods to heterogeneous agents. Ref. [11] relies on specific implicit coordination, Ref. [12] assumes that agents have the same type of control input, and Ref. [13] depends on an abstraction kinodynamic model of the robot; the latter makes the VO algorithm truly independent of the model. VO methods were further extended to the system with non-holonomic constraints by testing the optimality of sampling control [14], by mapping between the holonomic speed and the nonholonomic control input [15]. The latter guarantees a lower computational cost. Following the concept proposed in Ref. [15], we map the avoidance velocity to the nonholonomic control input. Thanks to the contributions of Refs. [16,17] on the three-dimensional velocity obstacle (3D-VO) method, the range of feasible solutions (obstacle avoidance velocity or acceleration) has been significantly expanded. Our method builds on the concept proposed in [17] which was successfully verified by experiments for collision avoidance with static obstacles [18] and extended to the dynamic system with nonholonomic constraints. The approaches mentioned above directly update velocities to avoid moving obstacles, and thus, it is summarized as the first-order method. Actually, compared with the quadcopter, the fixed-wing UAV cannot instantly change the direction of the velocity vector. Instead of computing an avoidance velocity command, it is possible to produce an avoidance acceleration that ensures that the collision avoidance can always be maintained.

A systematical extension of VO methods to second-order or higher-order systems has been conducted [12,13,19–21]. However, the aforementioned approaches contain some assumptions that limit their application to UAVs, such as only considering the simplified linear robot model, calculating the solution in a two-dimensional plane, and supposing that the robot can implement omni-directional acceleration. In [22], a 3D collision cone was utilized to examine collision condition and rigid quadrotor dynamics were considered. However, the obstacle avoidance only considers static obstacles in the environment. The fixed-wing UAV involved in our research has more complex dynamics. To the best of our knowledge, the extension of the 3D-VO method to a second-order nonlinear dynamic system of an underactuated fixed-wing UAV cannot be found in the literature. Mainly, there are the following difficulties: (1) a fixed-wing UAV is not only a nonholonomic system, but also a kinodynamic constrained nonholonomic system. It must take explicitly into account this nature of UAVs in planning strategies. However, collision avoidance may then no longer be guaranteed. (2) The dynamics of a fixed-wing UAV is represented as a second-order nonlinear differential equation with variable coefficients, which make it extremely difficult to directly derive a feasible velocity set or acceleration set, as described in [12,19,20].

1.2. Contributions

(1) Inspired by the work [19], we incorporate second-order kinodynamic constraints into the velocity by limiting the set of feasible velocities to those that can be achieved with position error below a predefined value. A reference trajectory resulting from the 3D-VO approach can be retained as formulated in [20]. We derive the control input of the velocity loop as well as the attitude loop to enable the UAV to converge to the reference trajectory, such that the motion planning problem can be transformed into a trajectory tracking problem, which has been extensively studied [23,24]. Since the derivation of the reference trajectory does not directly depend on the kinodynamic model of the UAV, complex derivation of control obstacles (CO) [20] or acceleration–velocity obstacles (AVO) [19] can be avoided.

(2) The additional tasks add to the overall flexibility of the UAV. Inspired by the concept of behavior-based control [2,25], we establish a guidance framework, where multi-tasks are

arranged according to a hierarchy plan and conducted in a desired priority. The subtasks can be defined as velocity vectors as presented in [25]. In line with the conventional behavior-based approaches and similar to the approach proposed in [2], we decompose the overall mission into several rotations, which are parameterized by quaternions.

1.3. Organization

The rest of the paper is organized as follows. In Section 2, we give some important definitions, the dynamic model of a fixed-wing UAV, and the problem statement. In Section 3, we introduce an improved 3D-VO method for the UAV collision avoidance. In Section 4, a hierarchical architecture to deal with the multiple task is represented. In Section 5, we describe in detail the design of the translation loop and attitude loop controllers. The results presented in this paper are illustrated by a series of simulations in Section 6. The paper ends with conclusion in Section 7.

2. Problem Formulation

2.1. Notation

Matrices are in capital $\boldsymbol{M}$, vectors are in bold $\boldsymbol{x}$, sets are in mathcal $\mathcal{H}$, and the Minkowski sum of two sets is expressed as $\mathcal{H} \oplus \mathcal{F}$. The Euclidean norm of vector $\boldsymbol{x}$ is expressed as $\|\boldsymbol{x}\|$. The superscript indicates the coordinate, and the coordinate systems involved in this article include the inertial coordinate system [26], n; body coordinate system, b; velocity coordinate system, s; error coordinate, e; avoidance coordinate system, a; and desired coordinate system, d.

The conversion between coordinate systems commonly relies on rotation matrixes or quaternions. A velocity vector $\boldsymbol{v}$ in b-frame $\boldsymbol{v}^b = \begin{bmatrix} v_{\boldsymbol{x}}^b & v_y^b & v_z^b \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^3$ coincides with the x-axis of the s-frame. Then, the velocity vector $\boldsymbol{v}$ in the s-frame is expressed as $\boldsymbol{v}^s = \begin{bmatrix} \|v^b\| & 0 & 0 \end{bmatrix}^{\mathrm{T}} = \boldsymbol{R}_b^s \boldsymbol{v}^b$, where the rotation matrix

$$\boldsymbol{R}_b^s \in SO^3 = \left\{ \boldsymbol{R} \in \mathbb{R}^3 : \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} = \boldsymbol{I},\ \det(\boldsymbol{R}) = 1 \right\}$$

($\boldsymbol{I}$ is an identity matrix) represents the rotation from frame-b to frame-s. The quaternion is represented by a column matrix:

$$\boldsymbol{q}_{sb} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \eta_{sb} & \boldsymbol{\varepsilon}_{sb}{}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \cos(\frac{\theta_{sb}}{2}) & \boldsymbol{k}_{sb}{}^{\mathrm{T}} \sin(\frac{\theta_{sb}}{2}) \end{bmatrix}^{\mathrm{T}}$$

where η_{sb} is a scalar, $\boldsymbol{\varepsilon}_{sb} \in \mathbb{R}^3$ is a vector, $\theta_{sb} = \cos^{-1}\left(\frac{v^b \cdot v^s}{\|v^s\|^2}\right)$ represents the rotation angle, and $\boldsymbol{k}_{sb} \in \mathbb{R}^3 = \frac{v^b \times \boldsymbol{v}^s}{\|v^b \times \boldsymbol{v}^s\|}$ represents the rotation axis. The quaternion and the rotation matrix can be transformed through the equation:

$$\boldsymbol{R}_b^s = I + 2\eta_{sb} S(\boldsymbol{\varepsilon}_{sb}) + 2S^2(\boldsymbol{\varepsilon}_{sb}),\ S(\boldsymbol{v}) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{1}$$

Taking the derivative of the rotation matrix, we get $\dot{\boldsymbol{R}}_b^s = \boldsymbol{R}_b^s S\left(\boldsymbol{w}_{sb}^b\right)$, where is the projection of the rotation of vector $\boldsymbol{b}$ relative to vector $\boldsymbol{c}$ on $\boldsymbol{b}$.

Quaternion multiplication can deal with the problem of combing finite time rotations of the rigid body. Supposing there are two consecutive rotation quaternions, then they can be combined into an equivalent quaternion through quaternion multiplication: $\boldsymbol{q} \otimes \boldsymbol{p} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{p}$, where $\boldsymbol{M}(\cdot)$ is the quaternion multiplication operator: $\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix}$.

Theorem 1. **([27]).** *If a rigid body makes a finite number of consecutive rotations around the axes that pass the same fixed point, then the size of the composite rotation angle has nothing to do with the order of each continuous rotation, but the direction of the composite rotation is related to the order of each continuous rotation.*

That is to say, the final attitude of the rigid body is related to the sequence of continuous rotation, and the position of the composite rotation axis changes with the sequence of continuous rotation.

The rotation of the UAV between two coordinate systems can be disassembled into multiple continuous rotations: $\boldsymbol{q}_{na}$=$\boldsymbol{q}_{ne} \otimes \boldsymbol{q}_{ea}$, where the formula represents the rotation from frame-a to frame-e, and then to frame-n. According to Theorem 1, defining the current quaternion q_{ea} relative to the previous quaternion q_{ne}.guarantees the direction that a vector rotated by q_{na} is pointing in is determine by the last quaternion q_{ea}. This inspires us to define each rotation as a subtask and arrange subtasks according to their importance, then combine subtasks into a total task. For example, the task represented by quaternion $\boldsymbol{q}_{ea}$ takes precedence over the task represented by $\boldsymbol{q}_{ne}$. After the high-level task $\boldsymbol{q}_{ea}$ is completed, it will degenerate into a unit quaternion $\begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, and its effect on the composite rotation will reduce to zero. Then, the system will proceed to the next task defined as $\boldsymbol{q}_{ne}$.

2.2. UAV Dynamics Model

The dynamic of a fixed-wing UAV is represented by the following equations of motion:

$$\dot{\boldsymbol{p}}^{n}_{UAV} = R^{n}_{b}\boldsymbol{v}^{b}_{UAV} \tag{2}$$

$$m\dot{\boldsymbol{v}}^{b}_{UAV} = F^{b}_{ag}\left(v^{b}_{UAV}\right) - mS\left(\boldsymbol{w}^{b}_{nb}\right)\boldsymbol{v}^{b}_{UAV} + m\boldsymbol{u}_{1} \tag{3}$$

$$J\dot{\boldsymbol{w}}^{b}_{nb} = M^{b}_{at}\left(\boldsymbol{w}^{b}_{nb}, \boldsymbol{v}^{b}_{UAV}\right) - S\left(\boldsymbol{w}^{b}_{nb}\right)J\boldsymbol{w}^{b}_{nb} + A\boldsymbol{u}_{2} \tag{4}$$

In the Equation (2), $\boldsymbol{p}^{n}_{UAV}$ represents the position of the UAV relative to the n-frame, and $\boldsymbol{v}^{b}_{UAV} = \begin{bmatrix} v^{b}_{x} & v^{b}_{y} & v^{b}_{z} \end{bmatrix}^{\mathrm{T}}$ is the velocity of the UAV in b-frame. In the Equation (3), m is the quality of the UAV, F^{b}_{ag} is the combined force of the aerodynamic and gravity forces of the UAV in b-frame, which is a nonlinear function of $\boldsymbol{v}^{b}_{UAV}$. $\boldsymbol{u}_{1} = \begin{bmatrix} T^{b}_{x} & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ is the control input of the translation loop, and only includes a thrust of the body. In the Equation (4), $\boldsymbol{J}$ is the inertia matrix, M^{b}_{at} is the resultant moment of the aerodynamic and thrust moments of the UAV, which is a nonlinear function of $\boldsymbol{w}^{b}_{nb}, \boldsymbol{v}^{b}_{UAV}$, and $\boldsymbol{u}_{2} = \begin{bmatrix} \varpi_{a} & \varpi_{e} & \varpi_{r} \end{bmatrix}^{\mathrm{T}}$ is the control input of the rotation loop, denoting the rudder angle.

The guidance system computes a desired quaternion, a desired angular velocity, and a desired angular acceleration. The translation loop and rotation loop solve the control inputs $\boldsymbol{u}_{1}, \boldsymbol{u}_{2}$ to track the desired speed and attitude, respectively. The structure of the UAV controller is shown in Figure 1.

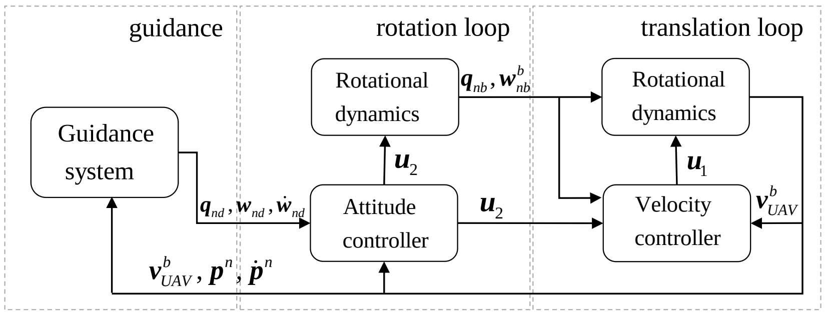

guidance
rotation loop
translation loop
Rotational dynamics
$\boldsymbol{q}_{nb}, \boldsymbol{w}^{b}_{nb}$
Rotational dynamics
Guidance system
$\boldsymbol{u}_2$
$\boldsymbol{u}_1$
$\boldsymbol{q}_{nd}, \boldsymbol{w}_{nd}, \dot{\boldsymbol{w}}_{nd}$
Attitude controller
$\boldsymbol{u}_2$
Velocity controller
$\boldsymbol{v}^{b}_{UAV}$
$\boldsymbol{v}^{b}_{UAV}, \boldsymbol{p}^{n}, \dot{\boldsymbol{p}}^{n}$

Figure 1. Controller structure.

2.3. Problem Statement

Assumption:

1. The UAV and obstacle are seen as spheres with radii r_{UAV}, r_{OBS}, respectively;
2. The obstacle mentioned in this article is assumed to be a moving but non-maneuvering obstacle;
3. Suppose sensors mounted on the UAV can accurately provide measurements, such as locations $\boldsymbol{p}_{UAV}, \boldsymbol{p}_{OBS} \in \mathbb{R}^3$, velocities $\boldsymbol{v}_{UAV}, \boldsymbol{v}_{OBS} \in \mathbb{R}^3$ and radii r_{UAV}, r_{OBS} of the UAV itself and obstacles;
4. The control system cycle is consistent with the detection system cycle;
5. The influence of wind is ignored during the flight of the UAV to simplify the problem.

The UAV and the obstacle are moving in a three-dimensional space $\mathcal{W} = \mathbb{R}^3$. $\mathcal{X} \subset \mathbb{R}^{m1}$ represents the set of states of the UAV, $m1$ denotes the dimension of the state space, $\mathcal{U} \subset \mathbb{R}^{m2}$ represents the set of control input of the UAV, and $m2$ denotes the dimension of the control input

$$\dot{\boldsymbol{x}}(t) = f_1(\boldsymbol{x}(t)) + f_2(\boldsymbol{x}(t))\boldsymbol{u}(t) \tag{5}$$

where $\boldsymbol{x} \in \mathcal{X}, \boldsymbol{u} \in \mathcal{U}$, $f_1(\cdot)$ and $f_2(\cdot)$ are nonlinear functions of the state $\boldsymbol{x}$. Given an initial $\boldsymbol{x}(t_0)$ and a control input $\boldsymbol{u}(t)$, the state of the robot at time $t \geq t_0$ can be solved. If $\mathrm{rank}[f_2(\boldsymbol{x}(t))] < \dim[\boldsymbol{x}(t)]$, the system described in Equation (5) is said to be underactuated. The position of the UAV $\boldsymbol{p}_{\mathrm{UAV}}$ can be derived from $\boldsymbol{x}(t_0)$ and $\boldsymbol{u}(t)$:

$$\boldsymbol{p}_{UAV}(t) = f_3(\boldsymbol{x}(t_0), \boldsymbol{u}(t)) \tag{6}$$

where $f_3(\cdot) \in \mathcal{X} \times \mathcal{U} \to \mathbb{R}^3$. $D(\boldsymbol{p}, r) = \{\boldsymbol{l} \mid \|\boldsymbol{l}\text{-}\boldsymbol{p}\| \leq r\}$ denotes the area occupied by a sphere, centered at $\boldsymbol{p}$ with radius r. Similarly, the airspace occupied by the UAV can be expressed as $O(\boldsymbol{p}_{\mathrm{UAV}}) = D(\boldsymbol{p}_{UAV}, r_{UAV}) \in \mathbb{R}^3$, where r_{UAV} is the radius of the UAV. This also applies to obstacles $O(\boldsymbol{p}_{OBS}) = D(\boldsymbol{p}_{OBS}, r_{OBS}) \in \mathbb{R}^3$.

Definition 1. *The UAV will remain collision-free with the obstacle in period* τ.

$$O(\boldsymbol{p}_{UAV}(t)) \cap O(\boldsymbol{p}_{OBS}(t)) = \varnothing, \ \forall t \in \begin{bmatrix} t_0 & t_0 + \tau \end{pmatrix} \tag{7}$$

Definition 2. *The UAV arrives at the destination* $\boldsymbol{p}_{pre}$.

$$\boldsymbol{e} := \boldsymbol{p}_{UAV} - \boldsymbol{p}_{pre}, \ \|e(t_0)\| > \varepsilon > 0, \ \exists t > 0, \ \varepsilon \leq e(t) \leq \Lambda \tag{8}$$

When $\|e(t)\| \leq \Lambda$, we conclude that the UAV has arrived at the destination point. To achieve Definitions 1 and 2, we define a desired frame-*d* and make the frame-*s* align with the desired frame-d by controlling the attitude and speed of the UAV. Quaternions $\boldsymbol{q}_{ea}$ and $\boldsymbol{q}_{ne}$ are used to represent the obstacle avoidance and target tracking sub-task, respectively, and then the quaternion multiplication $\boldsymbol{q}_{na}$=$\boldsymbol{q}_{ne} \otimes \boldsymbol{q}_{ea}$ is utilized to combine these sub-tasks into a total task, which is defined as the desired quaternion, q_{nd}: $:= q_{na}$. The purpose of this work can be defined as follows: given the initial state $\boldsymbol{x}(t_0)$ and the information of the obstacle, computing a control input $\boldsymbol{u}(t)$ for the UAV, such that

$$q_{nd} \to \begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{9}$$

3. Improved 3D-VO Method for UAV Collision Avoidance

Under the guidance framework proposed in this work, all subtasks are described by position vectors in the *n*-frame. In this section, we introduce obstacle avoidance points resulting from the improved 3D-VO algorithm.

3.1. Formation of 3D-Velocity Obstacle Set

For UAVs, 3D-VO [17] methods define the set of velocities that will collide with moving obstacles within the time horizon τ. The set is denoted by $\mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS})$:

$$\mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS}) = \{\boldsymbol{v}_{UAV} |\ \mathcal{R}(\boldsymbol{p}_{UAV}, \boldsymbol{v}_{UAV} - \boldsymbol{v}_{OBS}) \cap (\mathcal{D}(\boldsymbol{p}_{UAV}, r_{UAV}) \oplus \mathcal{D}(\boldsymbol{p}_{OBS}, r_{OBS})) \neq \varnothing\ \} \tag{10}$$

where $\mathcal{R}(\boldsymbol{p}, \boldsymbol{v}) = \{\boldsymbol{p} + t\boldsymbol{v} | t > 0\}$ includes the rays with the starting point $\boldsymbol{p}$ and the direction $\boldsymbol{v}$. The UAV will not collide with the moving obstacle with velocity $\boldsymbol{v}_{OBS}$ when $\boldsymbol{v}_{UAV}$ lies outside $\mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS})$. The geometry of $\mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS})$ is a cone. Related parameters are shown in Equation (11).

$$\begin{cases} O_{vo} = \boldsymbol{v}_{OBS}, d_{ai} = \sqrt{d_{uo}^2 - r_{ps}^2} \\ d_{vo} = \frac{d_{ai}^2}{d_{uo}}, r_{vo} = r_{ps}\frac{d_{ai}}{d_{uo}} \\ \eta_{vo} = \arcsin\left(\frac{r_{vo}}{d_{ai}}\right) \\ \boldsymbol{k}_{vo} = \begin{bmatrix} \cos\theta_{uo}\cos\psi_{uo} \\ \cos\theta_{uo}\sin\psi_{uo} \\ \sin\theta_{uo} \end{bmatrix} d_{vo} \end{cases} \tag{11}$$

where d_{uo} is the distance between the UAV and the moving obstacle, $\boldsymbol{k}_{vo}$ and η_{VO} represent the axis and the half angle of the cone, respectively, as shown in Figure 2a. The position of the obstacle is described by the spherical coordinates relative to frame-n. The vertex $O_{\boldsymbol{VO}}$ of the cone is translated from the position O_{vo}^* of the UAV along $\boldsymbol{v}_{OBS}$, as shown in Figure 2b.

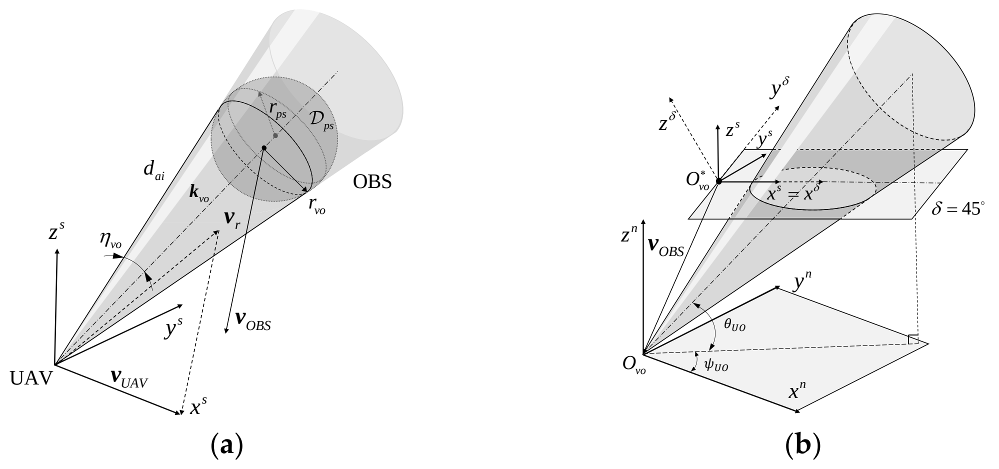


Figure 2. (**a**) Geometric elements of the VO cone. (**b**) The VO cone relative to different coordinate systems.

A standard cone parametric equation is illustrated in Equation (12)

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} q \\ q\tan\eta_{vo}\cos\varphi \\ q\tan\eta_{vo}\sin\varphi \end{bmatrix} \tag{12}$$

where $q \in [0, d_{VO}]$, $\varphi \in [0, 2\pi]$. Further, it is necessary to rotate the above cone twice to align its cone axis with $\boldsymbol{k}_{vo}$ and translate along $\boldsymbol{v}_{OBS}$ until the vertex coincides with O_{vo}. Finally, we obtain the geometry of $\mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS})$, which is called the VO cone. This process is represented by Equation (13).

When the distance d_{uo} between the UAV and the obstacle below the safety threshold distance $d_{safe}(d_{safe} > r_{UAV} + r_{OBS})$, $\mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS})$ is updated in real time to determine whether the collision will happen in the future. This is mainly based on whether the positive extension of $\boldsymbol{v}_{UAV}$ is inside the VO cone.

$$\begin{bmatrix} x_{VO} \\ y_{VO} \\ z_{VO} \end{bmatrix} = \begin{bmatrix} \cos\psi_{UO} & \sin\psi_{UO} & 0 \\ -\sin\psi_{UO} & \cos\psi_{UO} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_{UO} & 0 & -\sin\theta_{UO} \\ 0 & 1 & 0 \\ \sin\theta_{UO} & 0 & \cos\theta_{UO} \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + O_{vo} \quad (13)$$

$$\begin{cases} d_{UO} \leq d_{safe} \\ [\boldsymbol{v}_{UAV} - O_{vo}]'\boldsymbol{k}_{vo} > \cos\eta_{vo}\|\boldsymbol{v}_{UAV} - O_{vo}\| \end{cases} \quad (14)$$

If the condition in Equation (14) is satisfied, the positive extension of $\boldsymbol{v}_{UAV}$ is inside the VO cone, then the set of avoidance velocities further generate $\mathcal{AV}_{UAV|OBS} = W\backslash\mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS})$. When the avoidance velocity $\boldsymbol{v}_{avo}$ of the UAV in the next time horizon belongs to $\mathcal{AV}_{UAV|OBS}$, the collision can be avoided. The solution of $\boldsymbol{v}_{avo}$ is introduced in Section 4.2. The termination of obstacle avoidance mode cannot be judged by $v_{UAV} \notin \mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS})$ alone. When the obstacle avoidance mode is nearing the end, it is often the moment that the distance is close to the minimum, which can easily distort the VO cone, resulting in failing to find the correct obstacle avoidance velocity and even causing the control commands of the UAV chattering. Consequently, we introduce a λ angle, which is the angle between two vectors. The stage shown in Figure 3a is the initial stage of obstacle avoidance $\lambda_1 > \lambda^*$, where λ^* is a design parameter. Figure 3c shows $\lambda_3 < \lambda^*$, indicating that the avoidance mode has ended. The UAV is between the obstacle and the destination point. The stage shown in Figure 3b is the critical stage and $\lambda_2 = \lambda^*$. Hence, the judgment basis of the avoidance mode is redefined as

$$\begin{cases} d_{UO} \leq d_{safe} \\ [\boldsymbol{v}_{UAV} - O_{vo}]'\boldsymbol{k}_{vo} > \cos\eta_{vo}\|\boldsymbol{v}_{UAV} - O_{vo}\| \\ \lambda > \lambda^* \end{cases} \quad (15)$$

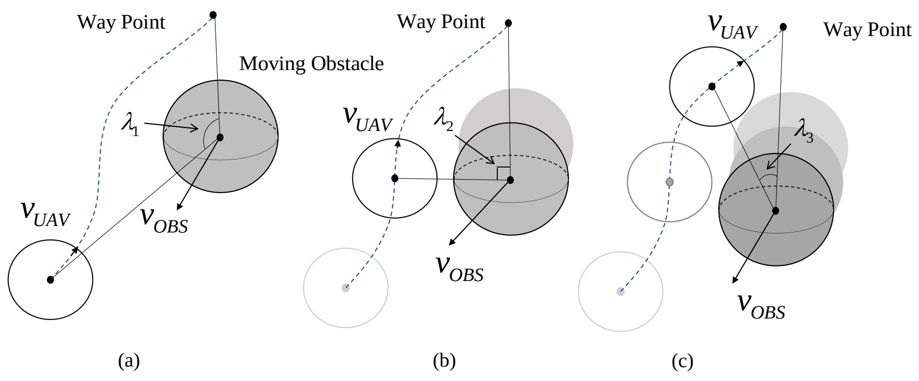


Figure 3. Task status: (**a**) collision avoidance, (**b**) critical state, (**c**) way point tracking.

3.2. Avoidance Planes

Similar to two-dimensional velocity obstacle (2D-VO) methods, the UAV needs to independently find the avoidance velocity in the set $\mathcal{AV}_{UAV|OBS}$ that is constantly being updated in real time. However, compared with the 2D-VO method, the topological dimension of the configuration space is expanded to three, and the difficulty of solving the avoidance velocity also increases. Hence, we follow the concept on the avoidance planes mentioned in Refs. [17,28]. Avoidance planes help transform the 3D problems into a series of 2D setups. $\mathcal{S}$ is defined as a series of discrete planes that rotate around the x-axis of the s-frame at certain angles. The plane of rotation angle δ is denoted as $S^\delta \in \mathcal{S}$, shown in Figure 2b.

The cross section obtained by the intersection of $\mathcal{VO}_{UAV|OBS}(\boldsymbol{v}_{OBS})$ and the plane $S^\delta = \mathbb{R}^2$ is denoted as $S^\delta_{UAV|OBS}(\boldsymbol{v}_B)$. If there is a collision in the S^δ plane, then selecting the avoidance velocity $\boldsymbol{v}_{avo} \in S^{\delta} \backslash S^{\delta}_{UAV|OBS}(\boldsymbol{v}_B)$ outside the area $S^{\delta}_{UAV|OBS}(\boldsymbol{v}_B)$ can avoid the collision.

To obtain expressions of conic sections in any obstacle avoidance plane, we implement a rotation-plane coordinate system $\{\boldsymbol{x}^{\delta}, y^{\delta}, z^{\delta}\}$ by rotating the s-frame around the $\boldsymbol{x}^{s}$ axis at an angle of δ and then projecting the VO cone under the rotation-plane coordinate system:

$$\begin{bmatrix} \boldsymbol{x}^{\delta} \\ y^{\delta} \\ z^{\delta} \end{bmatrix} = R^{\delta}_{s} \begin{bmatrix} x^{s} \\ y^{s} \\ z^{s} \end{bmatrix}, \qquad \boldsymbol{R}^{\delta}_{s} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\delta & \sin\delta \\ 0 & -\sin\delta & \cos\delta \end{bmatrix} \tag{16}$$

We set the $z^{\delta}-$axis coordinate of the VO cone to zero and obtain the coordinate points set
$\mathcal{F} == \left\{ \left(x^{\delta}_{vo}, y^{\delta}_{vo}\right) \middle| \; z^{\delta}_{vo} = 0 \right\}$, which is the coordinates of the conic section on the avoidance plane S^{δ}. When $z^{\delta} = 0$, for each φ, there exists a unique q corresponding to it. The mapping relationship is described as

$$q_{|z^{\delta}=0} = \frac{O_{vo_y}\sin\delta - O_{vo_z}\cos\delta}{\sin\psi_{uo}\cos\theta_{uo}\sin\delta - \tan\eta_{vo}\cos\varphi\cos\psi_{uo}\sin\delta - \tan\eta_{vo}\sin\varphi\sin\psi_{uo}\sin\theta_{uo}\sin\delta + \sin\theta_{uo}\cos\delta + \tan\eta_{vo}\sin\varphi\cos\theta_{uo}\cos\delta} \tag{17}$$

When δ takes different values, the shape of the conic section is different, roughly divided into ellipse, hyperbola, and triangle. Generally, there is a strong correlation between the shape of the conic section and collision risk for the UAV. The shapes of the collision risk from high to low are the triangle, hyperbola, and ellipse. Therefore, we would better find an appropriate $\boldsymbol{v}_{avo}$ in the elliptical conic section, which can be selected by judging the relationship between ζ_{δ} and η_{VO}:

$$\zeta_{\delta} = \arccos\left(\frac{\boldsymbol{k}'_{vo}}{d_{vo}} \begin{bmatrix} 0 \\ \sin\delta \\ \cos\delta \end{bmatrix} \right) \tag{18}$$

When

$$\zeta_{\delta} < \frac{\pi}{2} - \eta_{VO} \tag{19}$$

the conic section shape is an ellipse; otherwise it is a hyperbola or triangle.

3.3. Avoidance Velocity

We rely on the concept of the VO cone and the avoidance plane introduced in the previous sections to solve the avoidance velocity, mainly referring to Ref. [17], additionally taking into account the kinematic and dynamic constraints of the fixed-wing UAV. The original VO methods usually update the avoidance velocity to a point on the boundary of the VO cone. There are many ways to solve such a velocity, for example: by changing the magnitude without changing the direction, changing the direction without changing the magnitude, or a combination the two strategies, according to a certain optimization index. For fixed-wing UAVs, its velocity magnitude must be greater than the stall speed. At the same time, the top speed is limited, due to physical limitations. Therefore, the velocity control range seems to be narrow. Moreover, due to the inertia of the UAV, the thrust is less sensitive for controlling the velocity. Consequently, relying on rudders to change the direction of the velocity is considered to be the best way in this work.

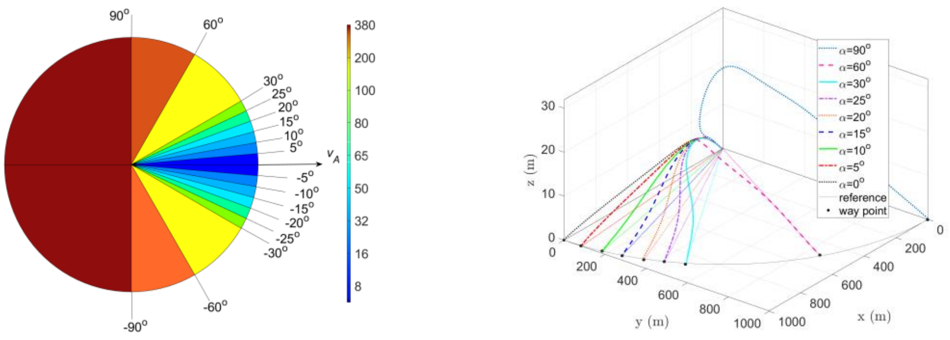
Constraints 1 (dynamic constraints): We introduce the concept of the maximum deflection angle α, which is the maximum change of the velocity in the direction, in order to characterize the dynamic constraints of the UAV, similar to the motion primitives proposed in Ref. [13]. In this work, $\boldsymbol{v}_{avo}$ is a continuous control parameter $\boldsymbol{v}_{avo} \mapsto \boldsymbol{u}(\boldsymbol{v}_{avo})$, indicating the magnitude and direction of the velocity of the UAV, defining the reference trajectory from the moment the UAV avoidance mode starts.

$$\boldsymbol{p}_{ref}(t) = \boldsymbol{p}(t_0) + (t - t_0)\boldsymbol{v}_{avo} \;, t \geq t_0$$

Meanwhile, the trajectory $\boldsymbol{p}_{UAV}(t)$ is the bijective function of the initial state $\boldsymbol{x}(t_0)$ and the control input $\boldsymbol{u}(t)$. The control input needs to meet the dynamic constraints of the UAV:

$$\mathcal{U} = \left\{ \boldsymbol{u} \in \mathbb{R}^3, \text{respect all dynamic constrains} \right\} \tag{20}$$

The trajectory of the UAV gradually converges to the reference trajectory under the control force. The maximum error between the trajectory and the reference trajectory $\chi(t) = \max\left(\| p_{UAV}(t) - p_{ref}(t) \| \right)$ shows a significant difference under different initial states and control input. The influence of the control parameter $\boldsymbol{v}_{avo}$ on χ is what we are concerned about. As shown in Figure 4b, the UAV can track the reference trajectory of the original direction with a small error, and the maximum error value increases almost linearly with the increase in α Figure 4a. We can quickly query the maximum error corresponding to a different value of α, which is constant relative to a given initial value. Given the initial state $\boldsymbol{x}(t_0)$, the set of control parameters that limit the maximum error below χ^* is defined as follows:



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

Figure 4. (**a**) Maximum tracking errors in (m) for the fixed-wing UAV and for varying direction of control velocities $\boldsymbol{v}_{avo}$. (**b**) The flight trajectory of the fixed-wing UAV under varying direction of control velocities. Example in the picture, the initial velocity is set to $\boldsymbol{v}_{UAV}$ = 40 m/s and the initial quaternion to $\boldsymbol{q}_{bs} = \left[1 \; \boldsymbol{0}^{\mathrm{T}}\right]^{\mathrm{T}}$.

$$\mathcal{A} := \left\{ \boldsymbol{v}_{avo} \middle| \max\left(\| \boldsymbol{p}_{UAV}(t) - \boldsymbol{p}_{ref}(t) \| \right) \leq \chi^* \; \boldsymbol{u} \in \mathcal{U} \right\} \tag{21}$$

where $\boldsymbol{v}^*_{avo}$ refers to the boundary of the set $\mathcal{A}$, corresponding to the maximum deflection angle $\alpha_{\max}$. In the avoidance plane $S^\delta = \mathbb{R}^2$, the relationship between the original velocity of the UAV and the avoidance velocity is expressed as follows:

$$\begin{cases} \boldsymbol{v}^\delta_{avo} = \boldsymbol{K}(\alpha) \cdot \boldsymbol{v}^\delta_{UAV} \\ \boldsymbol{K}(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \end{cases} \quad (\|\alpha\| \leq \alpha_{\max}) \tag{22}$$

Constraints 2 (collision avoidance and kinematic constraints): In addition to the magnitude constraint, $\boldsymbol{v}_{avo}$ must satisfy the obstacle avoidance constraint as well, that is, the end of the vector, which we call the velocity point, should be on the conic section $\mathcal{F}$.

$$\left(v^\delta_{avo_x}, v^\delta_{avo_y} \right) \in \left\{ \left(x^\delta_{vo}, y^\delta_{vo} \right) \middle| \; z^\delta_{vo} = 0 \right\} \cap \left\{ \left(x^\delta, y^\delta \right) \middle| \left(x^\delta \right)^2 + \left(y^\delta \right)^2 = \| \boldsymbol{v}_{UAV} \|^2 \right\} \tag{23}$$

In Equation (23), the velocity point in the avoidance plane S^δ is defined as the intersection of the conic section $\mathcal{F}$ and circular curve $\left\{ \left(x^\delta, y^\delta\right) \middle| \left(x^\delta\right)^2 + \left(y^\delta\right)^2 = \|\boldsymbol{v}_{UAV}\|^2 \right\}$. As shown in Figure 5, three situations need to be considered: (a) $\mathcal{F}$ and the circular curve only have one intersection, $\boldsymbol{v}_{avo_1}$, in the feasible region (gray). Then this point is defined as the velocity point. (b) There are two intersection points of $\mathcal{F}$ and the circular curve in the feasible region. Then the vector with the smallest α is chosen as the avoidance velocity. (c) There is no intersection point of $\mathcal{F}$ and the circular curve in the feasible region. Then the vector corresponding to $\alpha_{\max}$ is the avoidance velocity. In this case, a reasonable selection of d_{safe} [11] can ensure that the UAV avoids collision in the worst scenario (the UAV flying toward the obstacle):

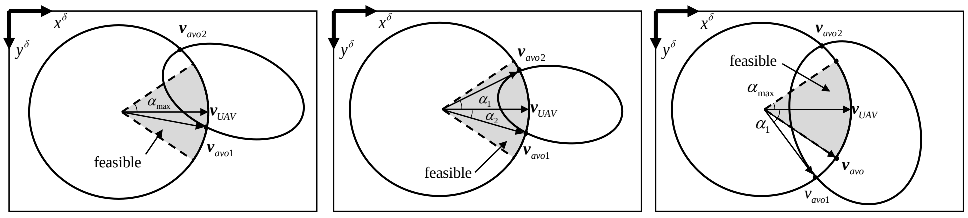


Figure 5. Different situations of the conic sections of the VO cone.

$$\begin{cases} d_{\min} = \sqrt{\left(2\sqrt{\dfrac{\|\boldsymbol{v}_{UAV}\| r_{pz}\tau}{\alpha_{\max}}} + \|\boldsymbol{v}_{OBS}\| T\right)^2 + r_{pz}^2} \\ T = \dfrac{\tau}{\alpha_{\max}} \arctan\left(\dfrac{2\sqrt{\|\boldsymbol{v}_{UAV}\| r_{pz}\tau\alpha_{\max}}}{\|\boldsymbol{v}_{UAV}\|\tau - \alpha_{\max} r_{pz}}\right) \\ d_{\text{safe}} \geq d_{\min} \end{cases} \tag{24}$$

where τ is the control system cycle. The smaller $\alpha_{\max}$ is, the larger d_{safe} is needed. Nevertheless, the selection of d_{safe} needs to meet the requirements of $\alpha_{\max}$ and consider the detection range of the sensor.

When Equation (15) is satisfied, $\boldsymbol{v}_{avo}^\delta$ is calculated in each avoidance plane $S^\delta(\delta = \delta_1, \delta = \delta_2 \cdots \delta = \delta_n)$. Among these solutions, the avoidance velocity with the smallest angle α will be selected and mapped to the inertial coordinate system to generate the avoidance points:

$$\boldsymbol{p}_{avo}^n(t) = \boldsymbol{p}_{UAV}^n(t_0) + \int_{t_0}^{t} \boldsymbol{v}_{avo}^n(t)\, dt \tag{25}$$

where t_0 is the initial time of the avoidance mode, $\boldsymbol{v}_{avo}^n(t)$ is constantly updated, and $\boldsymbol{p}_{avo}^n(t)$ will be used in the next chapter to construct the avoidance coordinate system and be mapped onto the control input.

4. Multiple Task Implementation Using a Hierarchical Architecture

Dealing with conflicting tasks is considered the key to behavior control, the difference of which is reflected in the way they arrange conflicting tasks and the way they construct subtasks into a total task. The hierarchical architecture used in this work is a form of behavior control. It arranges conflicting tasks in order of their importance. High-level tasks always subsume low-level tasks. The construction structure is shown in Figure 6. Ref. [29] utilized quaternion to describe the desired attitude of the UAV and applied it to the waypoint tracking problem. Our approach extends this previous work, defines the outputs of each subtask module as a task quaternion and calculates the total task quaternion by the quaternion multiplication. We assume that the UAV is flying under the effect of two behaviors.

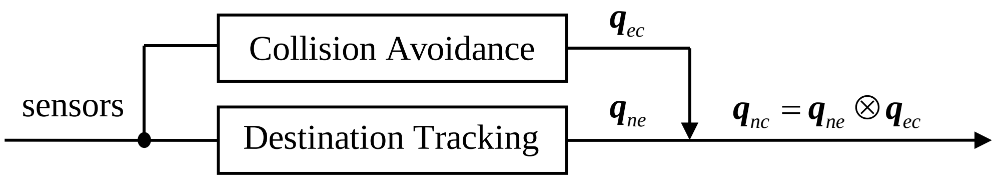


Figure 6. The hierarchical architecture.

Behavior 1: destination point tracking, task quaternion $\boldsymbol{q}_{ne}$.
Behavior 2: obstacle avoiding, task quaternion $\boldsymbol{q}_{ec}$.

The task quaternion of obstacle avoiding is established relative to the task quaternion of destination point tracking, so $\boldsymbol{q}_{ec}$ always subsumes $\boldsymbol{q}_{ne}$, which coincides with the concept of hierarchical architecture. Each task is defined by describing the error between the current position and the desired position. As shown in Figure 7, $\boldsymbol{p}^n_{UAV}$, $\boldsymbol{p}^n_{avo}$, $\boldsymbol{p}^n_{OBS}$, $\boldsymbol{p}^n_{d}$ respect the position of the UAV, avoidance point, obstacle, and destination point, respectively.

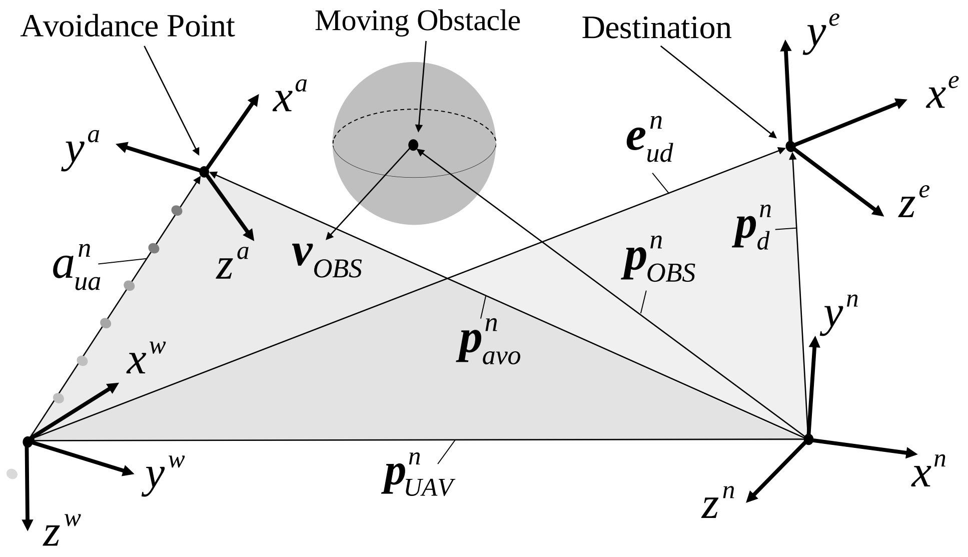


Figure 7. Position vectors in the relative three-dimensional space.

4.1. Task Quaternion 1: Destination Point Tracking

To guide the UAV to the destination point, the position error of the UAV is projected onto the $\boldsymbol{x}^e$ axis of the error coordinate system. Then, the UAV translates along with $\boldsymbol{x}^e$ under the action of the attitude controller until the destination point is reached. As in Figure 7, $\boldsymbol{e}^n_{ud}$ denotes the position error between $\boldsymbol{p}^n_d$ and $\boldsymbol{p}^n_{UAV}$ in the n-frame. The positive direction of the x-axis of the e-frame is aligned with $\boldsymbol{e}^n_{ud}$, so the e-frame can be defined through the rotation:

$$\boldsymbol{e}^e_{ud} = \begin{bmatrix} \|\boldsymbol{e}^n_{ud}\| & 0 & 0 \end{bmatrix}^{\mathrm{T}} = \boldsymbol{R}^e_n(\boldsymbol{p}^n_d - \boldsymbol{p}^n_{UAV}) = \boldsymbol{R}^e_n \boldsymbol{e}^n_{ud} \tag{26}$$

The control objective is to make $\boldsymbol{e}^e \to \left[\boldsymbol{0}^{\mathrm{T}}\right]^{\mathrm{T}}$. The rotation quaternion between the n-frame and the e-frame and the rotation matrix can be defined as

$$\theta_{ne} = \cos^{-1}\left(\frac{\boldsymbol{e}^e_{ud} \cdot \boldsymbol{e}^n_{ud}}{\|\boldsymbol{e}^n_{ud}\|^2}\right).\ \boldsymbol{k}_{ne} = \frac{\boldsymbol{e}^e_{ud} \times \boldsymbol{e}^n_{ud}}{\|\boldsymbol{e}^e_{ud} \times \boldsymbol{e}^n_{ud}\|} \tag{27}$$

$$\boldsymbol{q}_{ne} = \begin{bmatrix} \eta_{ne} & \boldsymbol{\varepsilon}^{\mathrm{T}}_{ne} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \cos(\frac{\theta_{ne}}{2}) & \boldsymbol{k}_{ne}{}^{\mathrm{T}} \sin(\frac{\theta_{ne}}{2}) \end{bmatrix}^{\mathrm{T}} \tag{28}$$

$$\boldsymbol{R}^n_e = \boldsymbol{I} + 2\eta_{ne} S(\boldsymbol{\varepsilon}_{ne}) + 2S^2(\boldsymbol{\varepsilon}_{ne}) \tag{29}$$

To obtain the desired angular velocity, we first take the derivative of Equation (26), which gives

$$\dot{\boldsymbol{e}}^e = -S(\boldsymbol{w}_{ne}^e)e^e - R_b^e \boldsymbol{v}_{UAV}^b \tag{30}$$

Then, substituting equation $-S(\boldsymbol{w}_{ne}^e)\boldsymbol{e}^e = S(\boldsymbol{e}^e)\boldsymbol{w}_{ne}^e$ into Equation (30), gives

$$\begin{bmatrix} \|\dot{\boldsymbol{e}}_{\boldsymbol{ud}}^n\| \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \|\boldsymbol{e}_{\boldsymbol{ud}}^n\|\boldsymbol{w}_{ne}^e(2) \\ \|\boldsymbol{e}_{\boldsymbol{ud}}^n\|\boldsymbol{w}_{ne}^e(3) \end{bmatrix} - R_b^e \boldsymbol{v}_{UAV}^b \tag{31}$$

It can be seen from Equation (31) that $\boldsymbol{w}_{ne}^e$ only contains the second and third terms. Therefore, rewriting Equation (30) as Equation (32) will not affect the final calculation result of $\boldsymbol{w}_{ne}^e$.

$$S(\boldsymbol{e}_{ud}^e)\boldsymbol{w}_{ne}^e = R_b^e \boldsymbol{v}_{UAV}^b \tag{32}$$

$$\boldsymbol{w}_{\boldsymbol{ne}}^{\boldsymbol{e}} = S^{\dagger}(\boldsymbol{e}_{ud}^e)\boldsymbol{R}_{\boldsymbol{b}}^{\boldsymbol{n}}\boldsymbol{R}_{\boldsymbol{n}}^{\boldsymbol{e}}\boldsymbol{v}_{UAV}^b \tag{33}$$

Equation (28) defines the rotation between two coordinate systems. When the task quaternion in Equation (28) degenerates into a unit quaternion, we think that two coordinate systems have reached a coincidence and the task is completed. At the same time, the desired angular velocity in Equation (33) converges to zero.

4.2. Task Quaternion 2: Collision Avoidance

What remains is establishing the avoidance coordinate system denoted as an a-frame. Ideally, the a-frame should be aligned with $\boldsymbol{v}_{avo}$; however, this cannot be achieved since the acceleration of obstacles is difficult to measure in practice. Therefore, we build the a-frame based on the avoidance points mentioned in Section 2. As shown in Figure 7, a_{ua}^n denotes the error between $\boldsymbol{p}_{UAV}^n$ and $\boldsymbol{p}_{avo}^n$, and the positive x-axis direction of the a-frame is aligned with a_{ua}^n.

$$\boldsymbol{a}^a = \begin{bmatrix} \|\boldsymbol{a}^e\| & 0 & 0 \end{bmatrix}^{\mathrm{T}} = \boldsymbol{R}_{\boldsymbol{e}}^a\boldsymbol{R}_{\boldsymbol{n}}^{\boldsymbol{e}}(\boldsymbol{p}_{avo}^n - \boldsymbol{p}_{UAV}^n) = \boldsymbol{R}_{\boldsymbol{e}}^a\boldsymbol{R}_{\boldsymbol{n}}^{\boldsymbol{e}}\boldsymbol{a}^{\boldsymbol{n}} \tag{34}$$

where $\boldsymbol{a}^n$ rotates from n-frame to e-frame, and then generates the quaternion to rotate from $\boldsymbol{a}^{\boldsymbol{e}}$ to a^a. That is to say, $\boldsymbol{a}^{\boldsymbol{n}}$ first rotates to align with the e-frame, and then to align with the a-frame. The obstacle avoidance task quaternion, the rotation matrix between the e-frame and the a-frame, and the obstacle avoidance angular velocity are expressed by the following Equations (35)–(38).

$$\theta_{ea} = \cos^{-1}\left(\frac{\boldsymbol{a}^a \cdot \boldsymbol{a}^{\boldsymbol{e}}}{\|\boldsymbol{a}^{\boldsymbol{e}}\|^2}\right).\ \boldsymbol{k}_{ea} = \frac{\boldsymbol{a}^a \times \boldsymbol{a}^{\boldsymbol{e}}}{\|\boldsymbol{a}^a \times \boldsymbol{a}^{\boldsymbol{e}}\|} \tag{35}$$

$$\boldsymbol{q}_{\boldsymbol{ea}} = \begin{bmatrix} \eta_{ea} & \boldsymbol{\varepsilon}_{\boldsymbol{ea}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \cos\left(\frac{\theta_{ea}}{2}\right) & \boldsymbol{k}_{\boldsymbol{ea}}^{\mathrm{T}}\sin\left(\frac{\theta_{ea}}{2}\right) \end{bmatrix}^{\mathrm{T}} \tag{36}$$

$$\boldsymbol{R}_a^e = \boldsymbol{I} + 2\eta_{ea}S(\boldsymbol{\varepsilon}_{ea}) + 2S^2(\boldsymbol{\varepsilon}_{ea}) \tag{37}$$

$$\boldsymbol{\omega}_{ea}^a = S^{\dagger}(a^a)(\boldsymbol{R}_{\boldsymbol{e}}^a S(\boldsymbol{\omega}_{\boldsymbol{ne}}^{\boldsymbol{e}})\boldsymbol{R}_{\boldsymbol{n}}^{\boldsymbol{e}} a^{\boldsymbol{n}} - \boldsymbol{R}_{\boldsymbol{n}}^{\boldsymbol{e}}\boldsymbol{R}_{\boldsymbol{e}}^a(v_{OBS}^n - v_{UAV}^n)) \tag{38}$$

Finally, two subtask quaternions are combined into a total task quaternion, which is defined as the desired quaternion. The desired angular velocity is also derived:

$$\begin{cases} \boldsymbol{q}_{nd} := \boldsymbol{q}_{na} = \boldsymbol{q}_{\boldsymbol{ne}} \otimes \boldsymbol{q}_{ea} \\ \boldsymbol{w}_{nd}^d := \boldsymbol{w}_{na}^a = \boldsymbol{R}_{\boldsymbol{e}}^a\boldsymbol{w}_{\boldsymbol{ne}}^{\boldsymbol{e}} + \boldsymbol{w}_{ea}^a \end{cases} \tag{39}$$

5. Attitude and Velocity Control Commands

The desired quaternion and the desired angular velocity are derived in the previous section. What remains is deriving the attitude and velocity control commands to track the desired attitude and velocity, respectively. For underactuated UAVs, Ref. [29] designed a sliding mode attitude controller, Ref. [30] proposed a velocity controller based on the backstepping technique, and Ref. [31] utilized a virtual saturated controller guaranteed the attitude error and the velocity error converge to zero. The controller designed in this chapter is mainly based on Refs. [29,30].

5.1. Velocity Controller

Given the Equation (3), we know the UAV is underactuated since $\boldsymbol{u}_1$ contains only a thrust T_x^b along $\boldsymbol{x}^b$. Hence, our approach introduces a velocity scalar and then utilizes the backstepping method to design a velocity controller to make the velocity scalar converge to a desired value $\boldsymbol{v}_d$. The speed scalar of the UAV is defined as

$$v_{UAV} = \sqrt{\left(\boldsymbol{v}_{UAV}^b\right)^{\mathrm{T}} \boldsymbol{v}_{UAV}^b} \tag{40}$$

Taking the derivative of Equation (40), we obtain

$$\dot{v}_{UAV} = \frac{1}{v_{UAV}} \left(\boldsymbol{v}_{UAV}^b\right)^{\mathrm{T}} \dot{\boldsymbol{v}}_{UAV}^b = \frac{\left(\boldsymbol{v}_{UAV}^b\right)^{\mathrm{T}} F_{ag}^b\left(\boldsymbol{v}_{UAV}^b\right)}{m v_{UAV}} + \frac{\boldsymbol{v}_{UAV_x}^b T_x^b}{m v_{UAV}} \tag{41}$$

where we use the fact, $\boldsymbol{v}_1^{\mathrm{T}} S(\boldsymbol{v}_2)\boldsymbol{v}_1 = 0$. The desired velocity scalar is a constant $v_d > 0$, and the velocity error is defined as $\hat{v} := v_{UAV} - v_d$. Given Equation (41) and the introduced variables $x_1 = \int_0^t \hat{v}\, d\tau$ and $x_2 = \hat{v}$, we find

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \frac{\left(\boldsymbol{v}_{UAV}^b\right)^{\mathrm{T}} F_{ag}^b\left(\boldsymbol{v}_{UAV}^b\right)}{m v_{UAV}} + \frac{\boldsymbol{v}_{UAV_x}^b T_x^b}{m v_{UAV}} \end{cases} \tag{42}$$

We further introduce a Lyapunov function $V_1 = \frac{1}{2}x_1^2$, then take the derivative of it, giving $\dot{V}_1 = x_1 x_2$. Another Lyapunov function is constructed as $V_2 = V_1 + \frac{1}{2}z^2$, where $z = x_2 + k_1 x_1$. Taking the derivative of V_2 gives

$$\dot{V}_2 = \dot{V}_1 + z\dot{z} = x_1 z - k_1 x_1^2 + z\left(\dot{x}_2 + k_1 x_2\right) = x_1 z - k_1 x_1^2 + z\left(\frac{\left(\boldsymbol{v}_{UAV}^b\right)^{\mathrm{T}} F_{ag}^b\left(\boldsymbol{v}_{UAV}^b\right)}{m v_{UAV}} + \frac{\boldsymbol{v}_{UAV_x}^b T_x^b}{m v_{UAV}} + k_1 x_2\right) \tag{43}$$

Above, a control input of the thrust is designed as

$$T_x^b = \frac{m v_{UAV}}{v_{UAV_x}^b}\left(-\frac{\left(\boldsymbol{v}_{UAV}^b\right)^{\mathrm{T}} F_{ag}^b\left(\boldsymbol{v}_{UAV}^b\right)}{m v_{UAV}} - k_1 x_2 - x_1 - k_2 z\right) \tag{44}$$

where parameters $k_1, k_2 > 0$. Substituting Equation (44) into Equation (43), we find

$$\dot{V}_2 = -k_1 x_1^2 - k_2 z^2 \leq -k_1 x_1^2 - k_2 {x_2}^2 \tag{45}$$

According to the Lyapunov stability theory, when $(x_1, x_2) \to (0, 0)$, the velocity error $\hat{v}$ will go to zero.

5.2. Attitude Controller

In this section, we implement a sliding mode attitude controller to align the s-frame of the UAV with the d-frame. We do so firstly by defining the rotational quaternion between the s-frame and the d-frame: $\boldsymbol{q}_{ds} = \boldsymbol{q}_{dn} \otimes \boldsymbol{q}_{nb} \otimes \boldsymbol{q}_{bs}$. When $q_{ds} \to \begin{bmatrix} 1 & \mathbf{0} \end{bmatrix}^{\mathrm{T}}$ under the effect of the control input, we think that the s-frame coincides with the d-frame. However, from a control point of view, we are more accustomed to the control variable converging to the relative origin. Therefore, we introduce a virtual variable $\boldsymbol{q}_{ds}^* := \begin{bmatrix} 1 & \mathbf{0} \end{bmatrix}^{\mathrm{T}} - \boldsymbol{q}_{ds}$ such that $\boldsymbol{q}_{ds}^* \to [\mathbf{0}]^{\mathrm{T}}$ is equivalent to $\boldsymbol{q}_{ds} \to \begin{bmatrix} 1 & \mathbf{0} \end{bmatrix}^{\mathrm{T}}$. Then, taking the derivative of $\boldsymbol{q}^*_{ds}$ gives $\dot{\boldsymbol{q}}^*_{ds} = D_q(\boldsymbol{q}^*_{ds})\boldsymbol{w}^s_{ds}$, where $D_q(\boldsymbol{q}^*_{ds}) := \frac{1}{2}\begin{bmatrix} \varepsilon^{\mathrm{T}}_{ds} \\ \eta_{ds}I + S(\varepsilon_{ds}) \end{bmatrix}$. Finally, we define the sliding variable [29,30]:

$$\boldsymbol{s} = \boldsymbol{w}^b_{nb} - \boldsymbol{w}^b_{ref} \tag{46}$$

$$\boldsymbol{w}^b_{ref} = \boldsymbol{w}^b_{nd} - \boldsymbol{w}^b_{bs} - \kappa_1 R^b_s D^{\mathrm{T}}_q \boldsymbol{q}^*_{ds} \tag{47}$$

where $\boldsymbol{w}^b_{ref}$ is the projection of the reference angular velocity relative to n-frame on the b-frame.

$$\dot{\boldsymbol{s}} = \frac{1}{J}\left(M^b_{at}\left(\boldsymbol{w}^b_{nb}, \boldsymbol{v}^b_{UAV}\right) - S\left(\boldsymbol{w}^b_{nb}\right)J\boldsymbol{w}^b_{nb} + A\boldsymbol{u}_2\right) - \dot{\boldsymbol{w}}^b_{ref} \tag{48}$$

We introduce a Lyapunov function $V_3 = \frac{1}{2}\kappa_2(\boldsymbol{q}^*_{ds})^{\mathrm{T}}\boldsymbol{q}_{ds} + \frac{1}{2}\boldsymbol{s}^{\mathrm{T}}J\boldsymbol{s}$ and take the derivative of V_3, giving

$$\dot{V}_3 = \kappa_2(\boldsymbol{q}^*_{ds})^{\mathrm{T}} D_q \boldsymbol{w}^s_{ds} + \boldsymbol{s}^{\mathrm{T}}\left(M^b_{at}\left(\boldsymbol{w}^b_{nb}, \boldsymbol{v}^b_{UAV}\right) - S\left(\boldsymbol{w}^b_{nb}\right)J\boldsymbol{w}^b_{nb} + A\boldsymbol{u}_2 - J\dot{\boldsymbol{w}}^b_{ref}\right) \tag{49}$$

where $\kappa_2 > 0$, and since $\boldsymbol{w}^s_{ds} = R^s_b\left(\boldsymbol{s} - \kappa_1 R^b_s D^{\mathrm{T}}_q \boldsymbol{q}^*_{ds}\right)$, $\dot{V}_3$ can be rewritten as

$$\dot{V}_3 = -\kappa_1\kappa_2(\boldsymbol{q}^*_{ds})^{\mathrm{T}} D_q D^{\mathrm{T}}_q \boldsymbol{q}^*_{ds} + \boldsymbol{s}^{\mathrm{T}}\left(M^b_{at}\left(\boldsymbol{w}^b_{nb}, \boldsymbol{v}^b_{UAV}\right) - S\left(\boldsymbol{w}^b_{nb}\right)J\boldsymbol{w}^b_{nb} + A\boldsymbol{u}_2 - J\dot{\boldsymbol{w}}^b_{ref} + \kappa_2 R^b_s D^{\mathrm{T}}_q \boldsymbol{q}^*_{ds}\right) \tag{50}$$

The control input $\boldsymbol{u}_2$ can be given similar to before as

$$\boldsymbol{u}_2 = A^{-1}\left(S\left(\boldsymbol{w}^b_{nb}\right)J\boldsymbol{w}^b_{nb} - M^b_{at}\left(\boldsymbol{w}^b_{nb}, \boldsymbol{v}^b_{UAV}\right) + J\dot{\boldsymbol{w}}^b_{ref} - \kappa_2 R^b_s D^{\mathrm{T}}_q \boldsymbol{q}^*_{ds} - K_s\boldsymbol{s}\right) \tag{51}$$

where K_s is a positive-definite matrix. Substituting Equation (51) into Equation (50) gives

$$\dot{V}_3 = -\kappa_1\kappa_2(\boldsymbol{q}^*_{ds})^{\mathrm{T}} D_q D^{\mathrm{T}}_q \boldsymbol{q}^*_{ds} - \boldsymbol{s}^{\mathrm{T}}K_s\boldsymbol{s} \le \frac{-\kappa_1\kappa_2}{8}\|\boldsymbol{q}^*_{ds}\|^2 - \check{}_{\min}(K_s)\|\boldsymbol{s}\|^2 \tag{52}$$

where $\check{}_{\min}$ is the minimal eigenvalue of the matrix K_s. According to the Lyapunov stability theory and Equation (52), we find $\left(\boldsymbol{q}^*_{ds}, \boldsymbol{s}\right) \to (0,0)$; then, as a consequence, the s-frame coincides with the d-frame.

A flowchart of the overall method is shown in Figure 8. Algorithm 1 summarizes the collision avoidance guidance frame, where $\mathcal{V}_k$ is the set of the avoidance velocity at the K-th step, and the function feasible Section(δ) checks if the shape of the conic section is expected, given a rotation angle δ in Equation (18). If Equation (19) satisfied, then return 'true'; else 'false'. We assumed that the control system as well as the detection system operate at the same rate and uniformly set the cycle as the time constant τ.

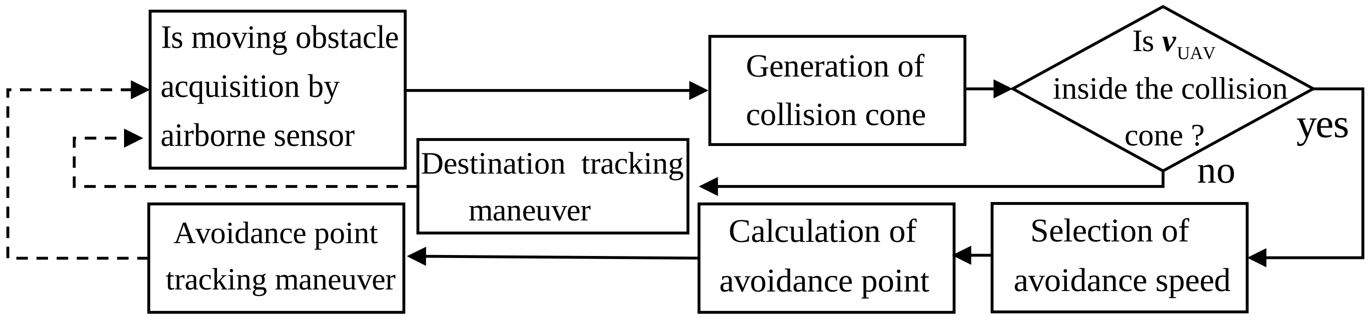

Is moving obstacle acquisition by airborne sensor
Generation of collision cone
Is $\boldsymbol{v}_{\mathrm{UAV}}$ inside the collision cone ?
yes
no
Destination tracking maneuver
Avoidance point tracking maneuver
Calculation of avoidance point
Selection of avoidance speed

Figure 8. Collision avoidance flow chart.

Algorithm 1. Attidude and Velocity Commands at the K-th Step

```
1. design parameter: $r_{ps}$ , $d_{safe}$ , $\alpha_{\max}$ , $p_d^n$, $\tau$, $\mathcal{AG} = [0°, \pm\frac{\pi}{4}, \frac{\pi}{2}]$
2. Input: $p_{UAV}^n$ , $p_{OBS}^n$ , $v_{UAV}$ , $v_{OBS}$ , $r_{OBS}$
3. Output: $u_2$ , $T_x^b$
4. begin
5.   obtain $q_{ne}(k), w_{ne}^e(k)$ by substituting $p_{UAV}^n(k), p_d^n$ into Equations (26)–(33)
6.   if $\|p_{UAV}^n - p_{OBS}^n\|_2 < d_{safe}$ and meet the conditions: Equation (15) then
7.     for each $\delta$ in $\mathcal{AG}$, do
8.     $v_{avo}^\delta \leftarrow$ Solution of quadratic Equation (23) with constrain Equation (22)
9.     if feasible Section($\delta$) = 'Ture'
10.       $\mathcal{V}_k \leftarrow \mathcal{V}_k \cup v_{avo}^\delta$
11.     end if
12.     end for
13.     if $\mathcal{V}_k \neq \varnothing$ then
14.       select $v_{avo}(k) \in \mathcal{V}_k$ with the smallest rotation angle $\alpha$
15.       return $v_{avo}(k)$
16.       // If $\mathcal{V}_k = \varnothing$, the $\mathcal{AG}$ needs to be extended
17.     end
18.     Map $v_{avo}(k)$ to $q_{ea}(k)$ , $w_{ea}^a(k)$ following Equation (25), Equations (34)–(38)
        $q_{nd}(k) \leftarrow$Equation (39)
19.   else
20.     $q_{ea}(k) \leftarrow \begin{bmatrix} 1 & \mathbf{0} \end{bmatrix}^{\mathrm{T}}$ ; $w_{ea}^a(k) \leftarrow [\mathbf{0}]^{\mathrm{T}}$; $q_{nd}(k) \leftarrow$ Equation (39)
21.   end if
22.   Compute velocity command $T_x^b(k)$ following Equations (40)–(44)
23.   Compute Attitude command $u_2(k)$ following Equations (46)–(51)
24.   Apply controls
25. end
```

6. Simulation

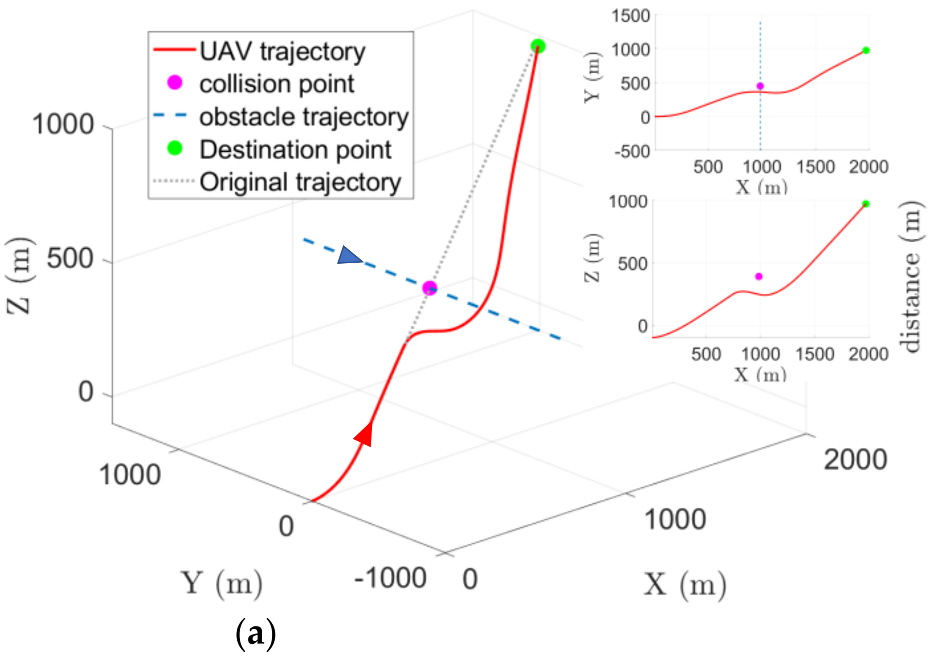
We performed a numerical simulation on a laptop to verify the effectiveness of the three-dimensional obstacle avoidance guidance algorithm proposed above. The laptop ran Windows 10 Professional 64-bit with an AMD Ryzen 7 4800H with Radeon Graphics CPU (2.9 GHz) and 16 GB RAM. All simulations were developed in the MATLAB 2020b environment. The parameters of the UAV, collision avoidance algorithm, and control system are summarized in Table 1. The initial states of the obstacle are set as $p_{OBS}^n(0) = \begin{bmatrix} 984.4 & 1396.43 & 390.1 \end{bmatrix}^{\mathrm{T}}$, and $\mathbf{v}_{OBS} = \begin{bmatrix} 0 & -40 & 0 \end{bmatrix}^{\mathrm{T}}$. The deflection angle is constrained in the range of $\pm 15°$, and the maximum thrust of the engine is limited to 250 N.

Table 1. Parameters of the fixed-wing UAV, collision avoidance and control system.

Fixed-Wing UAV		**3D-VO Collision Avoidance**		**Control System**	
$m = 20.64$ kg	$p_{UAV}^n(0) = [0\ 0\ -100]^{\mathrm{T}}$	$r_{ps} = 100$ m	$d_{safe} = 500$ m	$k_1 = 2$	$k_2 = 2$
$v_{UAV}^b(0) = [30\ 0\ 0]^{\mathrm{T}}$	$q_{nb}(0) = [1\ 0\ 0\ 0]^{\mathrm{T}}$	$\lambda^* = 90°$	$v_d = 40$ m/s	$\kappa_1 = 2$	$\kappa_2 = 2$
$w_{nb}^b(0) = [0\ 0\ 0]^{\mathrm{T}}$	$p_d^n = [20\ 10\ 10]^{\mathrm{T}} * 10^3$	$\alpha_{\max} = 90°$	$\tau = 0.5$ s	$K_s = 2I$	$\Lambda = 5$ m

Simulation 1: Use the algorithm that updates the avoidance plane in real time.

Figure 9 a shows the trajectory of the UAV and the obstacle in the inertial coordinate system. The solid line represents the flight trajectory of the UAV. It can be seen from Figure 9a that the controller of the UAV leads to a smooth and continuous trajectory. Dashed and dotted lines respectively represent the moving target and the UAV in a non-avoidance mode. They intersect at a collision point (t = 23.8 s), indicating that if the avoidance maneuver is not performed, a collision will occur at that time. Figure 9b shows the change of the distance between the UAV and the moving obstacle over time, and the minimum distance between the UAV and the moving obstacle is greater than r_{ps} as expected. It took 41.35 s to complete the task.



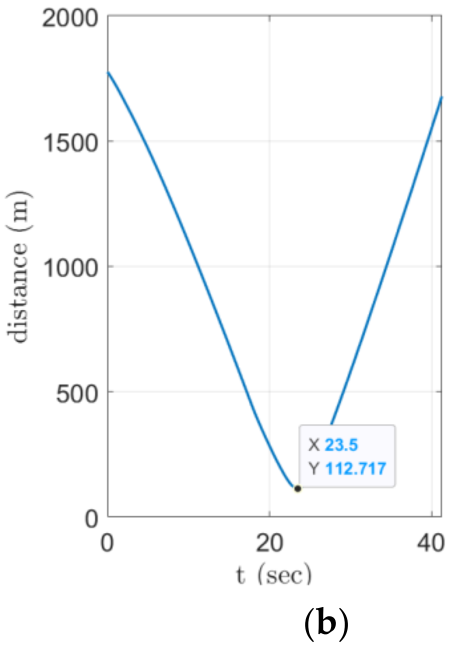


Figure 9. (**a**) Trajectory of fix-wing UAV and obstacle in inertial coordinate system, (**b**) distance between fixed-wing UAV and obstacle (Simulation 1).

Figure 10 shows the control input of the rotation loop and the translation loop, the velocity, the angular velocity, and the attitude information of the UAV.

Figure 10c shows that the thrust engine is started in the initial stage (t = 0–2.1 s), pushing the UAV to a predetermined velocity. As shown in Figure 10d,e during the period (t = 0–17.2 s), the quaternion $\boldsymbol{q}_{\mathrm{ds}}^{*}$ representing the error between the d-frame and the s-frame gradually converges to zero, indicating that the UAV can be directed toward the destination point. When an obstacle is detected and Equation (15) is satisfied, signal Y becomes 1 and the controller continuously controls the UAV keeps a certain safety distance with the obstacle, as in Figure 10a,b. In Figure 10f, the dotted, dashed, and solid lines represent $\boldsymbol{v}_{avo_x}$, $\boldsymbol{v}_{avo_x}$ and $\boldsymbol{v}_{avo_z}$, respectively. In the avoidance mode (t = 17.2–18.8 s), $\boldsymbol{v}_{avo_x}$ increases, $\boldsymbol{v}_{avo_x}$ as well as $\boldsymbol{v}_{avo_z}$ decrease, and the thrust remains at 0, indicating that the UAV has only changed the direction of the velocity. Due to the avoidance maneuver, the attitude of the UAV changes drastically. Figure 10 g shows the attitude, including the roll, pitch and yaw angle. During the period (t = 17.2–18.8 s), the yaw angle continues to decrease, indicating that the UAV has adopted the strategy of avoiding obstacles on the right side. The sudden increase in the attack angle and the sideslip angle are shown in Figure 10h reflects the fact that the b-frame cannot be aligned with the s-frame immediately due to the existence of the UAV's moment of inertia. As shown in Figure 10d, $\boldsymbol{q}_{ds}^{*}$ has undergone two obvious divergence and reconvergence processes after experiencing the initial disturbance and then converging to 0. The first time is the process of converging to the obstacle avoidance point in the avoidance mode, and the latter is the process of re-converging to the destination point in the guidance mode.

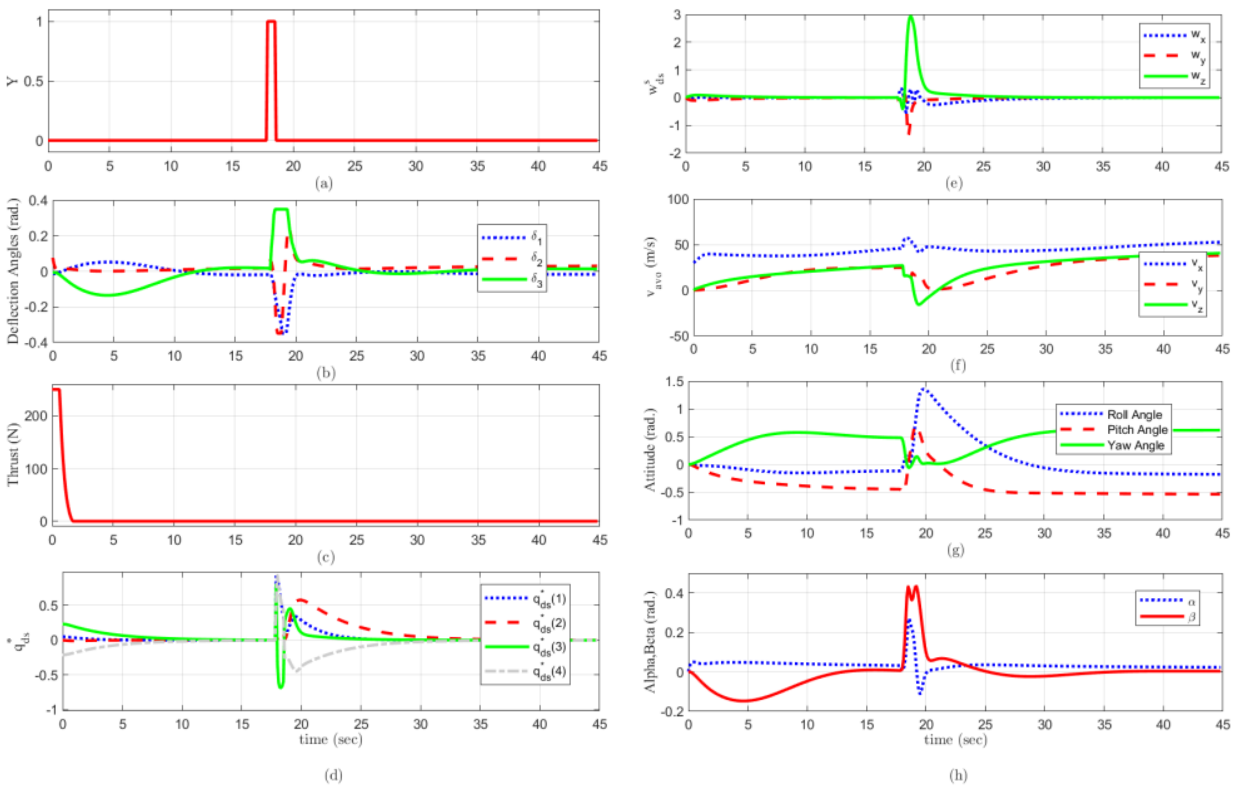
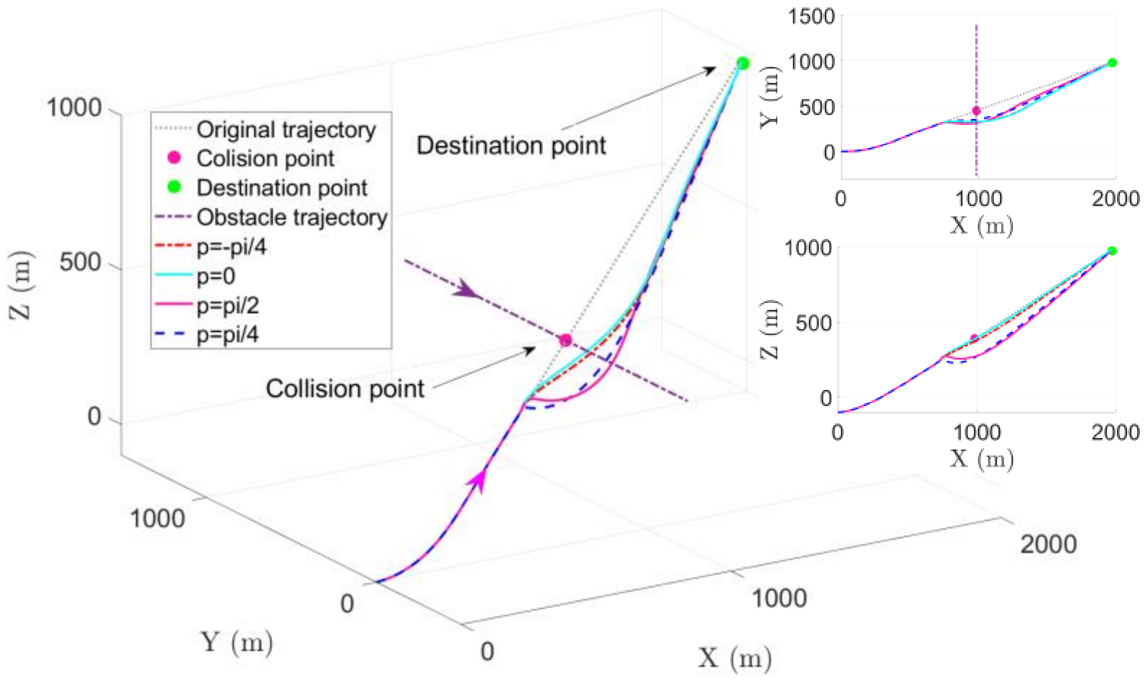


Figure 10. Related parameters of fix-wing UAV (Simulation 1): (**a**) status signal of UAV, (**b**) control input, (**c**) thrust, (**d**) attitude tracking error, (**e**) error of angular velocity, (**f**) velocity in inertial coordinate system, (**g**) attitude, (**h**) alpha and beta.

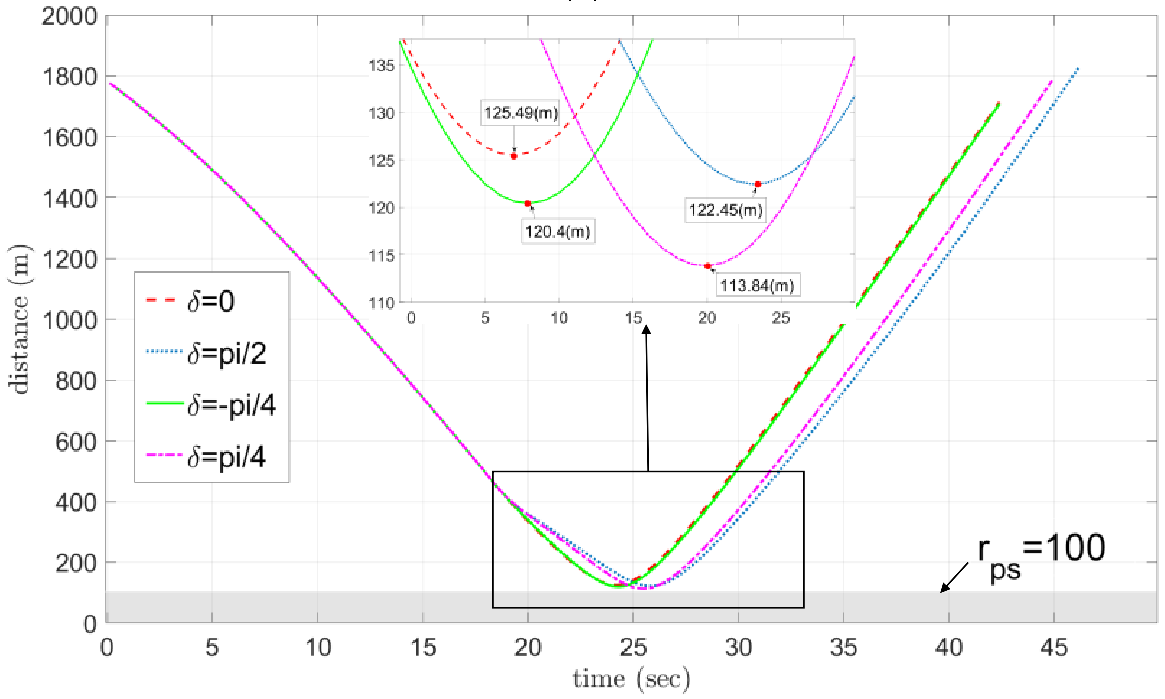
Simulation 2: Fixed avoidance plane with different rotation angles: $\delta_1 = 0°$, $\delta_2 = 45°$, $\delta_3 = -45$ and $\delta_4 = 90°$.

We also performed a simulation where the UAV does not update the avoidance plane but chooses a fixed avoidance plane with a pre-set rotation angle, while the initial state and control parameters are the same as before. The simulation was implemented to quantify the impact of the rotation angle of the obstacle avoidance plane on the obstacle avoidance. Figure 11a shows four trajectories of the UAV with different avoidance planes. It can be seen from Figure 11b that different avoidance planes mainly affect the minimum distance and avoidance time in the macroscopic view.

When the obstacle avoidance plane rotation angle is $\delta = 90°$, 45, the flight time is 46.2 s and 44.9 s, respectively. However, the flight time is 42.4 s when the obstacle avoidance plane rotation angle is $\delta = 0°$ or -45, and the trajectory is smoother. When we adopted the strategy of updating the rotation angle, under the same initial conditions, the flight time was 41.35 s, which is less than that in the fixed avoidance plane case. By analyzing the minimum distance between the UAV and the moving obstacle, we can observe that the algorithm adopting an updated avoidance plane can guarantee a less conservative avoidance trajectory. The reason for this difference is that in the fixed-angle avoidance plane, the choice of the avoidance velocity cannot always be guaranteed to be the smallest deviation from the original velocity; thus, the UAV flew over an unnecessarily long path. As shown in Figure 11b the minimum distance between the UAV and the obstacle was 113.84 m when the rotation angle of avoidance planes was defined as $\frac{\beta}{4}$. This is because the velocity of the UAV and the moving obstacle are almost simultaneously in the S^{δ} plane during the avoidance stage. Therefore, comparing the fixed avoidance plane and updated avoidance plane implementations of the method, we found that the latter guarantees the trajectory to be a less conservative and faster process.



(**a**)



(**b**)

Figure 11. (**a**) Trajectories of fixed-wing UAVs in Inertial coordinate system for several values of δ (Simulation 2), (**b**) distance between the fixed-wing UAV and obstacle for several values of δ.

7. Conclusions

In this work, we present a guidance framework that can enable the UAV to move along the collision-free smooth reference trajectory. In particular, we extend the traditional 3D-VO methods to kinodynamic constrained nonholonomic UAVs. The approach was to limit the set of feasible velocities to those that can be achieved with a position error below a predefined value. A reference collision-free trajectory resulting from the 3D-VO approach can be retained by integrating the updated avoidance velocity. The simulation results verified that the UAV can implement collision avoidance maneuvers under allowable trust and torque. We discussed fixed avoidance plane and updated avoidance plane implementations of the method and showed that the latter guarantees the trajectory to be more optimized but processed slower. Compared with other VO-based algorithms, the main innovation of this method is reflected in the extension of the 3D-VO to underactuated systems. The approach is to map underactuated variables to existing actuators. The method we proposed in this paper allows for fast online control and low computational cost, and flexible task assignment was presented in the detailed simulation experiments.

Although the guidance framework proposed in this paper is desirable, there is still large room for improvement. This algorithm was not tested in a multi-obstacle scenario. The extension to cooperative obstacle avoidance of UAV formation is yet to be considered, which will become our next research focus. The robustness against external disturbances, such as wind gusts, was not discussed.

Author Contributions: Writing—original draft preparation, Y.Q.; writing—review and editing, Y.Q.; supervision, W.Y.; funding acquisition, W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the financial support from the National Natural Science Foundation of China (11472136), China Postdoctoral Science Foundation Project (2019M651838) and Jiangsu Postgraduate Research and Practice Innovation Program (KYCX20_0327).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Zhu, X.; Yan, B.; Yue, Y. Path planning and collision avoidance in unknown environments for usvs based on an improved d* Lite. *Appl. Sci.* **2021**, *11*, 7863. [CrossRef]
2. Oland, E.; Andersen, T.S.; Kristiansen, R. Subsumption architecture applied to flight control using composite rotations. *Automatica* **2016**, *69*, 195–200. [CrossRef]
3. Moe, S.; Antonelli, G.; Pettersen, K.Y.; Schrimpf, J. Experimental results for set-based control within the singularity-robust multiple task-priority inverse kinematics framework. In Proceedings of the 2015 IEEE International Conference on Robotics and Biomimetics, Zhuhai, China, 6–9 December 2015; pp. 1233–1239.
4. Moe, S.; Teel, A.R.; Antonelli, G.; Pettersen, K.Y. Stability analysis for set-based control within the singularity-robust multiple task-priority inverse kinematics framework. In Proceedings of the 2015 54th IEEE Conference on Decision and Control, Osaka, Japan, 15–18 December 2015; pp. 171–178.
5. Lin, Y.; Saripalli, S. Sampling-Based Path Planning for UAV Collision Avoidance. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 3179–3192. [CrossRef]
6. Fan, Y.; Sun, X.; Wang, G.; Mu, D. Collision Avoidance Controller for Unmanned Surface Vehicle Based on Improved Cuckoo Search Algorithm. *Appl. Sci.* **2021**, *11*, 9741. [CrossRef]
7. Fiorini, P.; Shiller, Z. Motion planning in dynamic environments using velocity obstacles. *Int. J. Robot. Res.* **1998**, *17*, 760–772. [CrossRef]
8. Park, S.O.; Lee, M.C.; Kim, J. Trajectory Planning with Collision Avoidance for Redundant Robots Using Jacobian and Artificial Potential Field-based Real-time Inverse Kinematics. *Int. J. Control Autom. Syst.* **2020**, *18*, 2095–2107. [CrossRef]
9. Zhang, J.; Yan, J.; Zhang, P. Fixed-wing UAV formation control design with collision avoidance based on an improved artificial potential field. *IEEE Access* **2018**, *6*, 78342–78351. [CrossRef]
10. Choi, D.; Kim, D.; Lee, K. Enhanced potential field-based collision avoidance in cluttered three-dimensional urban environments. *Appl. Sci.* **2021**, *11*, 11003. [CrossRef]
11. Jenie, Y.I.; Van Kampen, E.J.; De Visser, C.C.; Ellerbroek, J.; Hoekstra, J.M. Selective velocity obstacle method for deconflicting maneuvers applied to unmanned aerial vehicles. *J. Guid. Control Dyn.* **2015**, *38*, 1140–1145. [CrossRef]
12. Bareiss, D.; Van Den Berg, J. Generalized reciprocal collision avoidance. *Int. J. Robot. Res.* **2015**, *34*, 1501–1514. [CrossRef]
13. Alonso-Mora, J.; Beardsley, P.; Siegwart, R. Cooperative Collision Avoidance for Nonholonomic Robots. *IEEE Trans. Robot.* **2018**, *34*, 404–420. [CrossRef]

14. Wilkie, D.; Van Den Berg, J.; Manocha, D. Generalized velocity obstacles. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 11–15 October 2009; pp. 5573–5578.
15. Alonso-mora, J.; Breitenmoser, A.; Rufli, M.; Beardsley, P.; Siegwart, R. *Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 83, pp. 203–216.
16. Chakravarthy, A.; Ghose, D. Generalization of the collision cone approach for motion safety in 3-D environments. *Auton. Robots* **2012**, *32*, 243–266. [CrossRef]
17. Jenie, Y.I.; Van Kampen, E.J.; De Visser, C.C.; Ellerbroek, J.; Hoekstra, J.M. Three-dimensional velocity obstacle method for uncoordinated avoidance maneuvers of unmanned aerial vehicles. *J. Guid. Control Dyn.* **2016**, *39*, 2312–2323. [CrossRef]
18. Tan, C.Y.; Huang, S.; Tan, K.K.; Teo, R.S.H. Three Dimensional Collision Avoidance for Multi Unmanned Aerial Vehicles Using Velocity Obstacle. *J. Intell. Robot. Syst. Theory Appl.* **2020**, *97*, 227–248. [CrossRef]
19. Van Den Berg, J.; Snape, J.; Guy, S.J.; Manocha, D. Reciprocal collision avoidance with acceleration-velocity obstacles. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3475–3482.
20. Rufli, M.; Alonso-Mora, J.; Siegwart, R. Recipro collision avoidance with motion continuity constraints. *IEEE Trans. Robot.* **2013**, *29*, 899–912. [CrossRef]
21. Ma, Y.; Manocha, D.; Wang, W. AutoRVO: Local Navigation with Dynamic Constraints in Dense Heterogeneous Traffic. *arXiv* **2018**, arXiv:1804.02915.
22. 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]
23. 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]
24. Elmokadem, T.; Zribi, M.; Youcef-Toumi, K. Trajectory tracking sliding mode control of underactuated AUVs. *Nonlinear Dyn.* **2016**, *84*, 1079–1091. [CrossRef]
25. Arrichiello, F.; Chiaverini, S.; Indiveri, G.; Pedone, P. The null-space-based behavioral control for mobile robots with velocity actuator saturations. *Int. J. Robot. Res.* **2010**, *29*, 1317–1337. [CrossRef]
26. Egeland, O.; Gravdahl, J.T. *Modeling and Simulation for Automatic Control*; Marine Cybernetics: Trondheim, Norway, 2002; ISBN 9788292356012.
27. Xiao, S. Multiplication of Quaternion Matrix and Its Feasibility. *ACTA Mech. Sin.* **1984**, *16*, 159–166.
28. Ellerbroek, J.; Brantegem, K.C.R.; Rene Van Paassen, M.M.; Mulder, M. Design of a coplanar airborne separation display. *IEEE Trans. Hum.-Mach. Syst.* **2013**, *43*, 277–289. [CrossRef]
29. Oland, E.; Schlanbusch, R.; Kristiansen, R. *Underactuated Waypoint Tracking of a Fixed-Wing UAV*; IFAC: Compiegne, France, 2013; Volume 2, ISBN 9783902823571.
30. Guan, J.; Yi, W.J.; Chang, S.J.; Liang, Z.D.; Lyu, Y.P. Study of flight path tracking and control of an UAV in 3D sp7ace. *Binggong Xuebao/Acta Armamentarii* **2016**, *37*, 64–70.
31. Oland, E.; Kristiansen, R. Trajectory tracking of an underactuated fixed-wing UAV. *AIP Conf. Proc.* **2014**, *1637*, 1345–1354.