mathematics

MDPI

Article

Trajectories Generation for Unmanned Aerial Vehicles Based on Obstacle Avoidance Located by a Visual Sensing System

Luis Felipe Muñoz Mendoza [1], Guillermo García-Torales [1], Cuauhtémoc Acosta Lúa [2,3], Stefano Di Gennaro [3,4] and José Trinidad Guillen Bonilla [1,*]

[1] Departamento de Electro–Fotónica, Centro Universitario de Ciencias Exactas e Ingenierías (C.U.C.E.I.), Universidad de Guadalajara (U. de G.), Blvd. M. García Barragán 1421, Guadalajara 44410, Jalisco, Mexico
[2] Departamento de Ciencias Tecnológicas, Centro Universitario de La Ciénega, Universidad de Guadalajara, Av. Universidad 1115, Ocotlán 47820, Jalisco, Mexico
[3] Center of Excellence DEWS, University of L'Aquila, Via Vetoio, Loc. Coppito, 67100 L'Aquila, Italy
[4] Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Via Vetoio, Loc. Coppito, 67100 L'Aquila, Italy
* Correspondence: trinidad.guillen@academicos.udg.mx; Tel.: +33-13-78-59-00 (ext. 27655)

Abstract: In this work, vectorial trajectories for unmanned aerial vehicles are completed based on a new algorithm named trajectory generation based on object avoidance (TGBOA), which is presented using a UAV camera as a visual sensor to define collision-free trajectories in scenarios with randomly distributed objects. The location information of the objects is collected by the visual sensor and processed in real-time. This proposal has two advantages. First, this system improves efficiency by focusing the algorithm on object detection and drone position, thus reducing computational complexity. Second, online trajectory references are generated and updated in real-time. To define a collision-free trajectory and avoid a collision between the UAV and the detected object, a reference is generated and shown by the vector, symmetrical, and parametric equations. Such vectors are used as a reference in a PI-like controller based on the Newton–Euler mathematical model. Experimentally, the TGBOA algorithm is corroborated by developing three experiments where the F-450 quadcopter, MATLAB® 2022[a], PI-like controller, and Wi-Fi communication are applied. The TGBOA algorithm and the PI-like controller show functionality because the controller always follows the vector generated due to the obstacle avoidance.

Keywords: TGBOA; UAV; trajectory generation; obstacle avoidance; Newton–Euler model; PI-like controller

MSC: 68T45; 93C95

Citation: Muñoz Mendoza, L.F.; García-Torales, G.; Acosta Lúa, C.; Di Gennaro, S.; Guillen Bonilla, J.T. Trajectories Generation for Unmanned Aerial Vehicles Based on Obstacle Avoidance Located by a Visual Sensing System. *Mathematics* **2023**, *11*, 1413. https://doi.org/10.3390/math11061413

Academic Editor: António Lopes

Received: 30 January 2023
Revised: 8 March 2023
Accepted: 10 March 2023
Published: 15 March 2023

1. Introduction

The study of unmanned aerial vehicles (UAVs) has grown in recent decades due to their features and the large field of applications that have been found in agronomy, surveillance, and tracking, among others [1,2]. Researchers' work on new methods for creating autonomous navigation on systems are focused on three stages, mainly the following [3]: obtaining information about the environment using different sensors and methodologies; [4,5] processing data to obtain a reference trajectory using a localization, mapping, or optimization algorithm [6,7]; and implementing flight controllers to keep the drone altitude and track the signal reference [8–10]. The first depends on a great variety of sensors such as cameras, light detection and ranging (LiDAR), or inertial measurement units (IMU). The second stage uses algorithms for object detection, and collision avoidance, as well as route optimization to find the best trajectory. The third stage consists of developing the flight controller, which is focused on approaching the reference to the position and posture desired for the UAV by means of a mathematical model and some control technique.

One way to obtain the reference is based on generating a vector, whose origin point is the position where the UAV starts its route, and the endpoint is the target to be reached, as shown in reference [11]: here, a vector is generated to move the UAV among separate trees by means of a photometric technique. However, various factors influence the drone path, such as changes in the environment and collisions; therefore, it becomes necessary to implement methods for obstacle detection and avoidance while the UAV trajectory is updated [5]. For this case, two methods are considered: manual and automatic. In the first, an operator sends the signal from a remote control to enable a flight mode [12,13]. In the second, a trajectory generator algorithm and a flight controller are combined [14,15]. The generator can be executed from the UAV or with a remote workstation.

To solve the trajectory generation problem, two main methods are considered: offline and online [16]. In the offline method, an algorithm analyzes and calculates a trajectory using a pre-load scenario that consists of a static image or model of the place with constrains already known by the system before the UAV take-off and calculates a trajectory for the controller. The environment must not change, since if it does (if some object appears), the technique loses its functionality due to the existence of a high possibility of a collision between the UAV and the object. On the other hand, the online method uses an algorithm based on a dynamic analysis of its sensors, detecting the object's movement to generate a customized trajectory [17,18]. Both methods are applicable; however, each method has its advantages and disadvantages. For example, the offline method is effective if the scenario is static because the UAV trajectory is calculated before its movement, but for a changing scenario, this method should not be considered because the collision between the UAV and an object is very possible. If the scenario is changing because of an object's movement or the UAV movement, the trajectory creation online is recommended [19] because the UAV trajectory is updated based on the scenario analysis. The algorithm's effectiveness, however, depends on the complexity of the data and the execution time. The literature has proposed different methods and devices to counter these drawbacks. Some works report waypoints with the GPS, as in [20], where a strategic route planning for a multi-UAV routing problem is shown, using a multi-step plan which includes automated definition for GNSS based on a georeferenced three-dimensional domain model, derivation of obstacle-free candidate routes, assignment of waypoints, and definition of time-stamped trajectories for all UAVs. Their aim is to ensure that the designed trajectories are reliable with the given positioning accuracy using propagated covariance as a metric. On the other hand, the authors of [21] improve the computationally fast trajectory generation algorithm using linear interpolation, which gives a fast calculation response, numerical simulation results are shown, and a nonlinear flight controller is also designed to track the generated trajectories. In addition, different works use artificial intelligence and optimization methods to obtain the best path in predetermined scenarios, as in [22], where a vision-based controller for a quadcopter is presented, to enable tracking of a moving target by a neural network which obtains image features.

For online trajectory cases, the most complex situation occurs when both the UAV and the objects are moving at the same time. Here, the algorithm must develop dynamic trajectories at a high speed according to the data analysis. To solve the problems mentioned above, several authors have proposed different methodologies. For example, in [23], the authors use a convolutional neural network to recognize images, path planning, and its implementation on an FPGA. In reference [24], the authors use the so-called simultaneous localization and mapping (SLAM) algorithm that allows the map building or updating of an unknown environment while tracking the drone location. A route planner is developed by updating itself as it explores various areas of the surrounding environment. Such methods can be functional in unknown scenarios but not dynamic. The processing tasks are hard to process due to the constant updating of the map. In [19], the authors propose a direct visual serving system for UAVs, using an onboard high-speed monocular camera with an integrated GPU. This technology is characterized by environment recognition through artificial vision and by sending signals to the controller, which reduces latency.

However, the equipment used for this technique is expensive and requires prior knowledge or estimation of the objects in its spatial environment. In [25], a vision-guided autonomous flight system for quadrotor navigation is presented. Using a real-time obstacle avoidance technique based on optical flow, the frontal obstacles can be detected and guide the quadrotor with a proper direction. Nevertheless, optical flow is difficult to achieve in the real-time performance on the low-cost hardware. On the other hand, implementing the generated trajectories is carried out by flight controllers that allow the drone to move into space by manipulating the speed of its rotors. The flight control system is the fundamental aspect of the quadrotor, becoming an important platform for UAV research and development, as in [25], where different control techniques are reported. Furthermore, Newton–Euler, Euler–Lagrange, or quaternions [26–31] are examples of mathematical models applied in controllers whose aim is to follow the reference by reducing the tracking error by adjusting the UAV posture and its rotational and translational movement.

Finally, Table 1 shows a comparation between some of the related work mentioned above.

Table 1. Comparation between different UAV trajectories' generation methods.

	Relevant Aspect	Disadvantage
[19]	High image sampling rates to improve the environment recognition. The position control signals are transmitted to the flight controller directly.	Expensive hardware and requires prior knowledge or estimation of the objects in its spatial environment.
[23]	Convolutional neural networks used to possible path recognition. ARM and FPGA implementation.	Large amount of computational resources used for CNN. Three vision sensors needed.
[24]	Monocular-inertial SLAM in the loop of navigation in a previously unknown environment.	Updating environment requires a lot of time and computer resources. Dynamic scenarios are not considered.
[25]	Path planning and real-time obstacle avoidance techniques. Frontal static obstacles can be detected and guide the quadrotor with a proper direction.	The presence of moving obstacles is not considered. Optical flow is not sufficient to achieve the real-time performance on the low-cost hardware.

In this article, a solution with low computational resources is presented for obstacle detection and avoidance, trajectory generation, and autonomous drone navigation using only a camera and without specialized high-performance hardware. It focuses on implementing a novel algorithm called trajectory generation based on object avoidance (TGBOA), which works in a real-time analysis through video frames taken from the camera placed on the UAV and communication with a workstation. Furthermore, a PI-like controller is developed and simulated, using as a reference the vector generated by the TGBOA algorithm. The Newton–Euler mathematical model is considered by the controller to represent the equations of the rigid body of the drone.

2. Materials and Methods

2.1. Materials

Figure 1 shows the proposed system for trajectory generation based on a video analysis and the evasion of localized objects through a vision system. The required material consists of a model OV5647 camera (Raspberry PI REES52, Taiwan) that works as a sensor with a Raspberry Pi 4B (Raspberry Pi Foundation, Raspberry Pi Foundation Maurice Wilkes Building St. John's Innovation Park Cambridge, UK) responsible for transmitting the video, a model F-450 quadcopter (INVENTO, India) (shown in Figure 2), a flight controller card (Pixhawk 3 (Bombay Electronics, Mumbai, India)) installed, and a laptop as a workstation. The camera technical features are M12 $\times$ 0.5 lenses, a 5MPixels sensor, a resolution of 640 $\times$ 480 pixels, 90 frames/s, and a size of 2.5 cm $\times$ 2.5 cm $\times$ 2.6 cm. The Raspberry features are a 1.5 GHz quad-core 64-bit ARM Cortex processor, 2 GB of LPDDR4 SDRAM, and Dual-band 802.11 ac wireless networking. The drone has four brushless direct current

motors (BLDC) and a 3DR-PIXHAWK controller card. The features of the laptop are 8Gb RAM and Intel i7 CPU. Finally, the software used is MATLAB® 2022a executing on the Windows 11 Operative System.

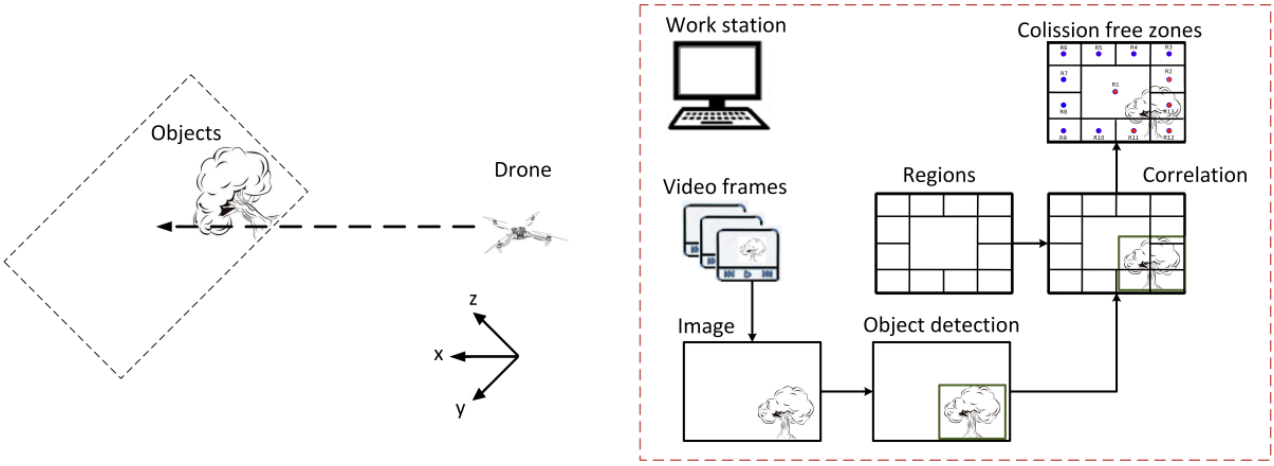


Figure 1. Vision system proposed for the TGBOA algorithm.

Figure 2. Quadcopter F-450.

The drone camera observes the front part where the UAV moves and generates a video during the flight. The video is transmitted in real-time to the workstation via Wi-Fi using the Raspberry Pi, and the transmission rate depends on the bandwidth Wi-Fi signal [32] and the adaptor of the workstation. In this case, the connection is made by wireless transmission at 2.4 GHz with a usable speed of 50–70 Mbps. Subsequently, the workstation analyzes the video frames considering the following objectives: object detection on the stage, collision-free zone calculation, and generation of free-collision trajectories for the flight controller.

2.2. Object Detection and Assignation of Collision-Free Zones

The generated video is 90 frames/second with 640 × 480 pixels for each frame and is sent from the UAV to the workstation. For the analysis, a frame is taken from the video, obtaining a digital RGB image as shown in Figure 3a. The image is represented through the discrete function $f(m,n)$ where $m = 1,2,\ldots,M$ and $n = 1,2,\ldots,N$ are the coordinates of the image pixels. Subsequently, the RGB image is transformed into an 8-bit gray-scale image and converted by the Otsu algorithm into a binary image [33]. In the binary image, the detected objects are white, and the background is marked in black, as illustrated in Figure 3b. Finally, the size, position, and centroid are obtained for each object.

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

Figure 3. Test case: (**a**) RGB image obtained from vision sensor; (**b**) objects located using the binary image obtained from the environment; (**c**) free zones marked in blue, and zones with objects marked in red.

The next step is calculating the collision-free regions, so the image $f(m,n)$ is divided into thirteen regions: $R_k(k = 1,2,\ldots,13)$. The first region R_1 has been assigned concerning the UAV size and is in the image center, and the image resolution limits the remaining thirteen zones, as observed in Figure 3c. If objects in the scenario are in one or more regions, then a correlation "A" between the detected objects and the region exists, and the following condition is true:

$$R_k \rightarrow \begin{cases} R_{k,\alpha} & iff\ A = 0 \\ R_{k,\beta} & iff\ A \neq 0' \end{cases} \tag{1}$$

where $R_{k,\alpha}$ indicates the $\alpha - th$ collision-free region and $R_{k,\beta}$ indicates the $\beta - th$ region with possible collision. Now, based on Equation (1), a region $R_{k,\alpha}$ is collision-free if, and only if, the correlation area A is equal to 0. On the other hand, region $R_{k,\beta}$ has a possible collision if, and only if, the correlation area A is different from zero. To illustrate the above, Figure 2b is considered, where the objects are in the regions R_4, R_5, $R_{9–13}$ (marked as red), and consequently, collision-free regions are $R_{1–3}$ y $R_{6–8}$ (marked in blue).

2.3. Trajectory Generation

Once the collision-free regions have been defined, it is possible to generate flight trajectories for the UAV controller to avoid a collision and improve the success of missions.

Figure 4 shows the geometry for generating the trajectories of the controller. Based on the figure mentioned above, the origin is the drone position whose coordinates are $P_c(0,0,0)$, and the endpoint of the trajectory is the center of the collision-free region. In this case, the coordinates will be $C_k(x_k,\ y_k, z_k)$: x_k is the distance from the UAV to the scenario, y_k represents the length, and z_k is the altitude. To generate the trajectory, the vector equation is used:

$$\vec{f}_k = \vec{P}_c + t_i \vec{v}_k \qquad i \in \mathbb{Z}, \tag{2}$$

where $\vec{P}_c$ is a vector such that $\vec{v}_k = \vec{P_c C_k}$ vector calculated from the UAV to the center of the *k-th* region R_k, $\vec{f}_k$ is the collision-free trajectory, and t are instants of time. From here, we go from a 2D problem to 3D as illustrated in Figure 4.

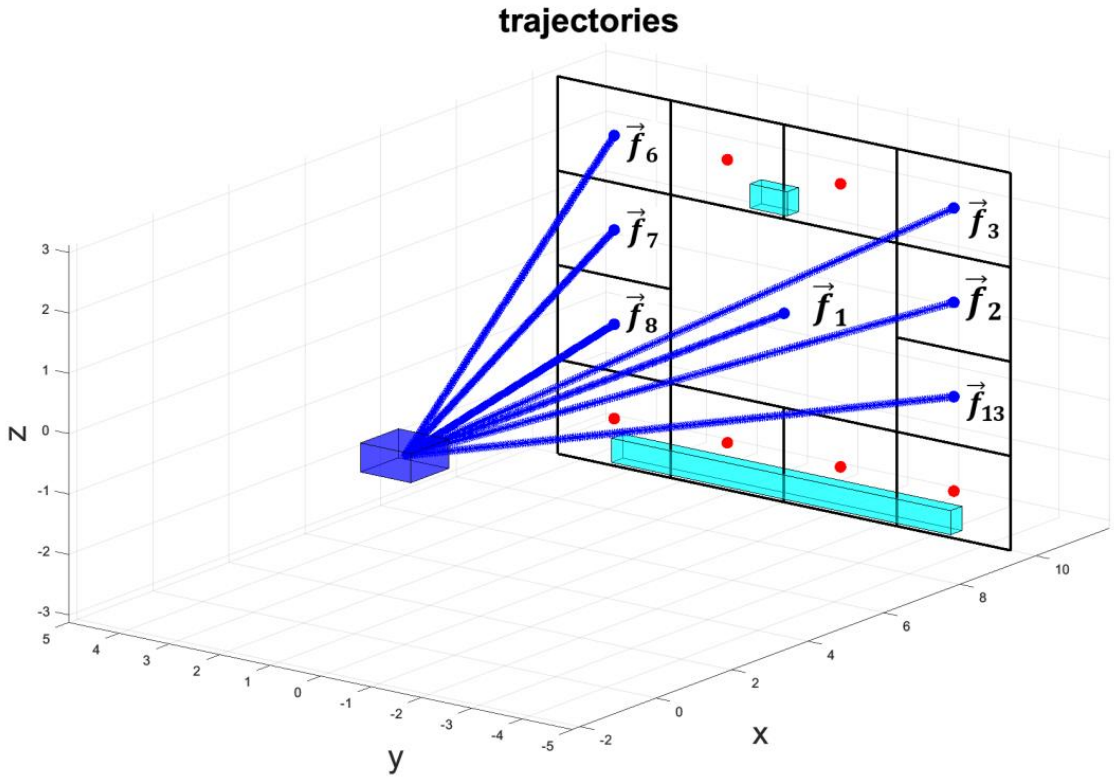


Figure 4. Geometry and collision-free paths generated from object detection and definition of collision-free regions.

Taking into consideration the geometry of the problem, Equation (2) can be rewritten as follows:

$$\vec{f}_k = (0,\ 0,\ 0) + t(x_k, y_k, z_k\ ), \tag{3}$$

From Equation (3), the following parametric equations are obtained,

$$\begin{aligned} x &= x_k t \\ y &= y_k t, \\ z &= z_k t \end{aligned} \tag{4}$$

Finally, the vector equation is represented in the form of symmetric equations,

$$\frac{x}{x_k} = \frac{y}{y_k} = \frac{z}{z_k}, \tag{5}$$

Equations (4)–(6) can be applied as a reference in flight controllers for the UAV. Figure 4 shows the graphical representation for Equations (3)–(5) (possible trajectory vectors), $\vec{f}_k$ $(k = 1, 2, 3, 6\text{–}8)$ where free regions are R_k $(1, 2, 3, 6\text{–}8)$ and the objects are detected in regions R_k $(4, 5, 9\text{–}13)$. Notice that vectors $\vec{f}_k$ $(k = 1, 2, 3, 6\text{–}8)$ are free trajectories and the drone does not have a collision due to the object evasion.

2.4. Controller Design

In this work, a controller is designed to verify the functionality of the trajectories generated in Section 2.3. To achieve this, the Newton–Euler mathematical model is used, which is presented below.

2.4.1. Mathematical Model

The UAV considered in this work is based on a quadcopter shown in Figure 5, which consists of a rigid body with four rotors. The movement of the drone is generated from the interaction of the forces resulting from each motor, which are represented as $F_i = b\omega_{p.i}^2$, where b is the thrust coefficient calculated by the physical and dynamic characteristics of the propellers while $\omega_{pi}, i = 1, 2, 3, 4$ are the angular velocities of each motor. Propellers 1 and 3 rotate clockwise, and propellers 2 and 4 rotate counterclockwise.

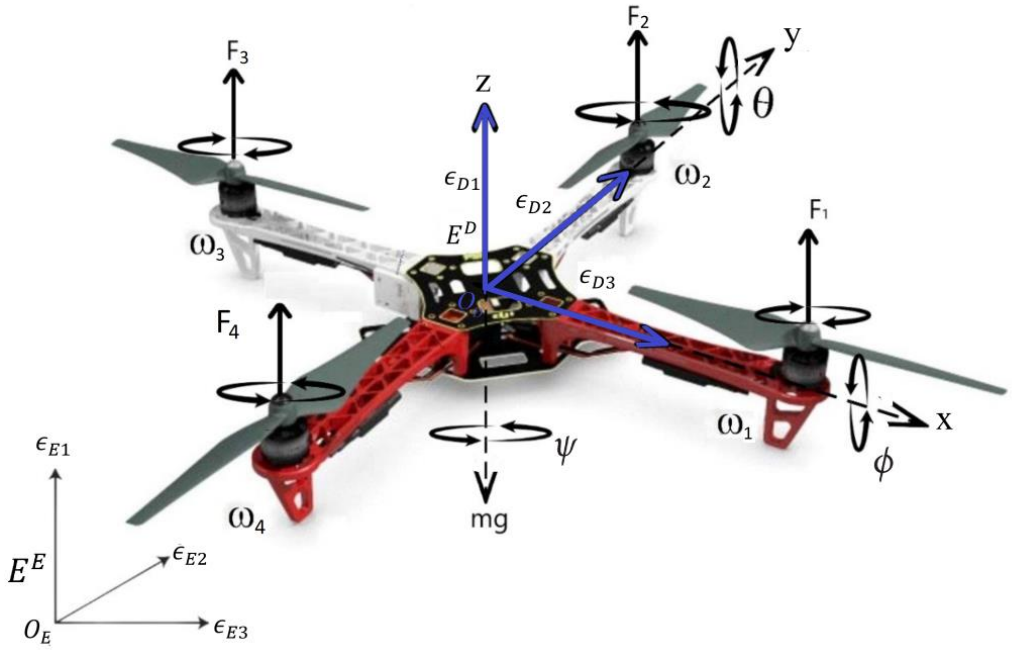


Figure 5. Frameworks used for quadcopter orientation.

For the mathematical model, two frameworks are considered: the one that remains fixed with respect to the Earth that we will call $E^E(O_E, \epsilon_{E1}, \epsilon_{E2}, \epsilon_{E3})$ and the frame of reference of the drone located at its center of mass $E^D(O_D, \epsilon_{D1}, \epsilon_{D2}, \epsilon_{D3})$. The absolute position of the UAV in terms of E^E is expressed by $\rho = (x, y, z)^T$ and its attitude is described by Euler angles $\Theta = (x, y, z)^T$: roll (ϕ) rotates on the x-axis, pitch (θ) works around the y-axis, and the z-turn is called yaw (ψ). All angles exist in the range $(-\pi/2\,, \pi/2)$. This allows the drone to have six degrees of freedom, according to the framework E^D, three to translation $v_D = \left(v_x, v_y, v_z\right)^T$ and three to rotation $\Omega = (\omega_1, \omega_2, \omega_3)^T$, which are linear and angular velocities, respectively.

To continue with the Newton–Euler model, the dynamics of the rigid body are obtained by the forces applied to the center of mass of the quadcopter, and the equation of motion for the quadcopter can be described by means of Figure 4 and the translation dynamics, expressed in the following system of equations.

$$\begin{aligned} \dot{\rho} &= v_D \\ \dot{v}_D &= \tfrac{1}{m} R_{DE} F_{prop} - \tfrac{1}{m} F_{grav} \\ \dot{\Theta} &= \dot{R}_{DE} S_\omega \\ J\,\dot{\Omega} &= -S_\omega J \Omega + \tau_{prop} - \tau_{gyro} \end{aligned} \tag{6}$$

where m is the quadcopter mass, $J = diag\left(I_x, I_y, I_z\right)$ is the inertia matrix of the quadcopter expressed in E^E, and F_{prop} and τ_{prop} are forces and moments produced by the propellers. Moreover, F_{grav} are forces and moments applied to the body of the helicopter, consisting of

its own weight, and τ_{gyro} is defined as the gyroscopic effects resulting from helix rotations. The forces and torques due to the external disturbances are here assumed to be negligible.

In Equation (6), S_ω is the antisymmetric matrix defined as:

$$S_\omega = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}, \tag{7}$$

Equation (7) in known as the dyadic representation of Ω.

Matrix R_{DE} describes the spatial orientation of the drone framework E^D to the Earth framework E^E. The sequence $R_{z,\psi}R_{y,\theta}R_{x,\phi}$ is used [34], and it is obtained by applying the product of three rotation matrices for each axis, so that

$$R_{x,\phi} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{pmatrix}; \; R_{y,\theta} = \begin{pmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{pmatrix}; R_{z,\psi} = \begin{pmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{DE} = R_{z,\psi}R_{y,\theta}R_{x,\phi} = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\psi s_\phi \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\theta s_\psi & c_\phi s_\theta s_\psi - c_\psi s_\phi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}, \tag{8}$$

with $c_\gamma = \cos(\gamma)$ and $s_\gamma = \sin(\gamma)$. $\gamma = \phi, \theta, \psi$.

The dynamics of angular velocities are expressed in matrix form:

$$\dot{R}_{DE} = \begin{pmatrix} 1 & s_\phi tg_\theta & c_\phi tg_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi sc_\theta & c_\phi sc_\theta \end{pmatrix}, \tag{9}$$

with $tg_\gamma = \tan(\gamma)$ and $sc_\gamma = \sec(\gamma)$. Moreover, using small angles, Equation (9) reduces to an identity matrix I_{3x3}. This assumption is justified by the fact that the movements of the quadcopter are slow and soft [35].

Furthermore, the input forces and moments are the torque applied to the roll angle τ_1, produced by the forces of the motors F_4 and F_2, respectively; the torque applied to pitch τ_3 is produced by the forces F_1 and F_3, respectively; due to Newton's third law, the propellers produce a yawing torque τ_2 on the body of the quadrotor, in the opposite direction of the propeller rotation. The helices' drag produces a torque in the angle of rotation in the body of the quadcopter and is represented by c. The distance between the quadcopter's center of mass and rotor shaft is represented by l. All these variables are defined as:

$$\begin{aligned} F_{prop} &= \begin{pmatrix} 0 \\ 0 \\ \Sigma F_i \end{pmatrix} \\ F_{grav} &= \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} \\ \tau_{prop} = \begin{pmatrix} \tau_1 \\ \tau_3 \\ \tau_2 \end{pmatrix} &= \begin{pmatrix} l(F_4 - F_2) \\ l(F_1 - F_3) \\ c(-F_1 + F_2 - F_3 + F_4) \end{pmatrix} = \begin{pmatrix} lu_3 \\ lu_2 \\ u_4 \end{pmatrix}. \end{aligned} \tag{10}$$

Next, to the dynamic model described in (6), it is possible to include the gyroscopic term τ_{gyro}, caused by the combination of the rotations of the airframe and the four rotors, i.e., due to the fact that the pair of rotors 1–3 rotate in the opposite direction of the pair 2–4 [36]:

$$\tau_{gyro} = \sum_{i=1}^{4} (-1)^i I_R \omega_R S\epsilon_3, \tag{11}$$

where $\epsilon_3 = (0\ 0\ 1)^T$ is the vector along the axis z in E^E, J_R is the moment of inertia of the propeller with respect to its axis of rotation, and $\omega_R = \omega_{R,1} - \omega_{R,2} + \omega_{R,3} - \omega_{R,4}$ is the so-called relative rotor speed.

Finally, using Equations (6) and (10) and under the small angle assumption, the quadcopter model is

$$\begin{aligned}
\ddot{x} &= \tfrac{1}{m}\left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right)u_1 \\
\ddot{y} &= \tfrac{1}{m}\left(c_\phi s_\theta s_\phi - s_\phi c_\psi\right)u_1 \\
\ddot{z} &= -g + \tfrac{1}{m}c_\phi c_\theta u_1 \\
\ddot{\phi} &= \left(\tfrac{J_y - J_z}{J_x}\right)\dot{\theta}\dot{\psi} - \tfrac{J_R}{J_x}\dot{\theta}\omega_R + \tfrac{l}{J_x}u_3 \\
\ddot{\theta} &= \left(\tfrac{J_z - J_x}{J_y}\right)\dot{\phi}\dot{\psi} + \tfrac{J_R}{J_y}\dot{\phi}\omega_R + \tfrac{l}{J_y}u_2 \\
\ddot{\psi} &= \left(\tfrac{J_x - J_y}{J_z}\right)\dot{\phi}\dot{\theta} + \tfrac{1}{J_z}u_4
\end{aligned} \tag{12}$$

where the control variable is u_i, $i = 1, 2, 3, 4$ are defined as in (10), and the parameter values used in (12) are shown in Table 2.

Table 2. Coefficients and variables of the quadcopter.

Parameter	Variable	Value
Mass	m	1.3 kg
Frame length	l	0.233 m
Inertia	J_x	8.825×10^{-3} kg m^2
	J_y	8.825×10^{-3} kg m^2
	J_z	14.39×10^{-3} kg m^2
Propeller inertia	J_R	104×10^{-6} kg m^2
Gravity	g	9.81 m/s^2
Thrust	b	52.5×10^{-6}
Drag	c	1.15×10^{-6} N s^2 rad^{-2}

Subsequently, the control problem consists of ensuring the convergence of the set of variables $X = (x, y, z, \psi)$ to references $X_{ref} = \left(x_{ref}, y_{ref}, z_{ref}, \psi_{ref}\right)$. To achieve this, control subsystems composed of the rotational and translational dynamics of the system are considered, which relate the control inputs u_i with spatial coordinates (x, y, z) by means of Euler angles $(\phi,\ \theta,\ \psi)$. The following coordinate change is considered: $\phi = \phi_1$, $\dot{\phi}_1 = \phi_2$; $\theta = \theta_1$, $\dot{\theta}_1 = \theta_2$; $\psi = \psi_1$, $\dot{\psi}_1 = \psi_2$ and the system of Equation (12) became:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
x_2 &= \tfrac{1}{m}\left(c_{\phi_1} s_{\theta_1} c_{\psi_1} + s_{\phi_1} s_{\psi_1}\right)u_1 \\
\dot{y}_1 &= y_2 \\
y_2 &= \tfrac{1}{m}\left(c_{\phi_1} s_{\theta_1} s_{\psi_1} - s_{\phi_1} c_{\psi_1}\right)u_1 \\
\dot{z}_1 &= z_2 \\
\dot{z} &= -g + \tfrac{1}{m}c_{\phi_1} c_{\theta_1} u_1 \\
\dot{\phi}_1 &= \phi_2 \\
\dot{\phi}_2 &= \left(\tfrac{J_y - J_z}{J_x}\right)\theta_2\psi_2 - \tfrac{J_R}{J_x}\theta_2\omega_R + \tfrac{l}{J_x}u_3 \\
\dot{\theta}_1 &= \theta_2 \\
\dot{\theta}_2 &= \left(\tfrac{J_z - J_x}{J_y}\right)\phi_2\psi_2 + \tfrac{J_R}{J_y}\phi_2\omega_R + \tfrac{l}{J_y}u_2 \\
\dot{\psi}_1 &= \psi_2 \\
\dot{\psi}_2 &= \left(\tfrac{J_x - J_y}{J_z}\right)\phi_2\theta_2 + \tfrac{1}{J_z}u_4
\end{aligned} \tag{13}$$

Therefore, the following four subsystems shall be considered: the altitude subsystem S_1, the latitudinal subsystem S_2, the longitudinal subsystem S_3, and the yaw S_4, and then the set of Equation (13) becomes:

$$\begin{aligned}
S_1 &= \begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = -g + \frac{1}{m} c_{\phi_1} c_{\theta_1} u_1 \end{cases} \\
S_2 &= \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \frac{1}{m}(c_{\phi_1} s_{\theta_1} c_{\psi_1} + s_{\phi_1} s_{\psi_1}) u_1 \\ \dot{\theta}_1 = \theta_2 \\ \dot{\theta}_2 = \left(\frac{J_z - J_x}{J_y}\right) \phi_2 \psi_2 + \frac{J_R}{J_y} \phi_2 \omega_R + \frac{l}{J_y} u_3 \end{cases} \\
S_3 &= \begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = \frac{1}{m}(c_{\phi_1} s_{\theta_1} s_{\psi_1} - s_{\phi_1} c_{\psi_1}) u_1 \\ \dot{\phi}_1 = \phi_2 \\ \dot{\phi}_2 = \left(\frac{J_y - J_z}{J_x}\right) \theta_2 \psi_2 - \frac{J_R}{J_x} \theta_2 \omega_R + \frac{l}{J_x} u_2 \end{cases} \\
S_4 &= \begin{cases} \dot{\psi}_1 = \psi_2 \\ \dot{\psi}_2 = \left(\frac{J_x - J_y}{J_z}\right) \phi_2 \theta_2 + \frac{1}{J_z} u_4 \end{cases}
\end{aligned} \tag{14}$$

Following this, each of the subsystems is described.

2.4.2. Altitude Control

As mentioned, in order to verify the applicability of the references for the UAV flight controller, an altitude control is proposed in this section. The altitude control subsystem S_1 is defined by the following:

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= -g + \frac{1}{m} c_{\phi_1} c_{\theta_1} u_1 \end{aligned} \tag{15}$$

Since the aim is to keep the UAV at a constant altitude according to the desired reference z_{ref}, with $z_{2,ref} = \dot{z}_{1,ref}$, then the tracking error and its dynamics are set for altitude control via the following:

$$\begin{aligned} e_{z_1} &= z_1 - z_{1,ref} \\ \dot{e}_{z_1} &= z_2 - \dot{z}_{1,ref} \end{aligned} \tag{16}$$

For virtual control gains k are used such that

$$-k_{z_1} e_{z_1} - k_{0z_1} I e_{z_1} = z_{2,d} - \dot{z}_{1,ref}$$

where an integrative term is added in e_{z_1}, in such a way that, $\dot{I}e_{z_1} = e_{z_1}$, and therefore,

$$\begin{aligned} z_{2,d} &= \dot{z}_{1,ref} - k_z \\ {}_1 e_{z_1} &- k_{0z_1} I e_{z_1}, \end{aligned} \tag{17}$$

The derivative of (17) with respect to time will be:

$$\begin{aligned} \dot{z}_{2,d} &= \ddot{z}_{1,ref} - k_{z1} \dot{e}_{z_1} - k_{0z_1} \dot{I} e_{z_1} \\ \dot{z}_{2,d} &= \ddot{z}_{1,ref} - k_{z1} \left(z_2 - \dot{z}_{1,ref}\right) - k_{0z_1} e_{z_1}, \end{aligned} \tag{18}$$

Now, the tracking error e_{z2} is obtained by the following:

$$e_{z_2} = z_2 - z_{2d};$$

and its dynamics results are

$$\dot{e}_{z_2} = -g + \frac{1}{m} c_{\phi 1} c_{\theta 1} u_1 - \dot{z}_{2,d} \quad , \tag{19}$$

then, the control input of altitude u_1 proposed is

$$u_1 = \frac{m}{c_{\phi 1} c_{\theta 1}} \left(\dot{z}_{2,d} + g - k_{z_2} e_{z_2} - k_{0z_2} I e_{z_2} \right), \tag{20}$$

with $c_\phi \neq 0$, $c_\theta \neq 0$.

Considering [37], the closed-dynamics considering (17) and (20) become

$$\begin{aligned} \begin{pmatrix} \dot{Ie}_{z_1} \\ \dot{e}_{z,1} \end{pmatrix} &= A_{z,1} \begin{pmatrix} Ie_{z_1} \\ e_{z,1} \end{pmatrix}, \quad A_{z,1} = \begin{pmatrix} 0 & 1 \\ -k_{0z_1} & -k_{z_1} \end{pmatrix} \\ \begin{pmatrix} \dot{Ie}_{z_2} \\ \dot{e}_{z,2} \end{pmatrix} &= A_{z,2} \begin{pmatrix} Ie_{z_2} \\ e_{z,2} \end{pmatrix}, \quad A_{z,2} = \begin{pmatrix} 0 & 1 \\ -k_{0z_2} & -k_{z_2} \end{pmatrix} \end{aligned} \tag{21}$$

where the integral gains k_{0z_1}, k_{0z_2} and proportional gains k_{z_1}, k_{z_2} are fixed so that the error dynamics are exponentially stable.

2.4.3. Longitudinal Motion Control

In this subsection, making use of the subsystem S_2 and applying the technique of virtual control, longitudinal motion is studied, which is described by the following system of equations.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \tfrac{1}{m} (c_{\phi_1} s_{\theta_1} c_{\psi_1} + s_{\phi_1} s_{\psi_1}) u_1 \\ \dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= \left(\tfrac{J_z - J_x}{J_y} \right) \phi_2 \psi_2 + \tfrac{J_R}{J_y} \phi_2 \omega_R + \tfrac{l}{J_y} u_3 \end{aligned} \tag{22}$$

The tracking error $e_{x_1} = x_1 - x_{1,ref}$ is considered, and the dynamics of error results:

$$\dot{e}_{x_1} = x_2 - \dot{x}_{1,ref}. \tag{23}$$

To eliminate disturbances, an integrative term $\dot{Ie}_{x_1} = e_{x_1}$ is added to the controller, and the virtual control is proposed,

$$x_{2,d} = \dot{x}_{1,ref} - k_{x_1} e_{x_1} - k_{0x_1} I e_{x_1}. \tag{24}$$

Differentiation $x_{2,d}$ with respect to the time,

$$\begin{aligned} \dot{x}_{2,d} &= \ddot{x}_{1,ref} - k_{x_1} \dot{e}_{x_1} - k_{0x_1} \dot{Ie}_{x_1} \\ \dot{x}_{2,d} &= \ddot{x}_{1,ref} - k_{x_1} \left(x_2 - \dot{x}_{1,ref} \right) - k_{0x_1} e_{x1}. \end{aligned} \tag{25}$$

The error is $e_{x_2} = x_2 - x_{2d}$ and its dynamics results are:

$$\begin{aligned} \dot{e}_{x_2} &= \dot{x}_2 - \dot{x}_{2,d} \\ \dot{e}_{x_2} &= \tfrac{1}{m} (c_{\phi_1} s_{\theta_1} c_{\psi_1} + s_{\phi_1} s_{\psi_1}) u_1 - \dot{x}_{2,d}; \end{aligned} \tag{26}$$

for stabilization of e_{x_2}, the value for s_{θ_1} can be fixed by solving (26) as follows:

$$s_{\theta_{1,d}} = \frac{m}{c_{\phi_1} c_{\psi_1} u_1} \left(\dot{x}_{2,d} - \frac{1}{m} s_{\phi_1} s_{\psi_1} u_1 - k_{x_2} e_{x_2} - k_{0x_2} I e_{x_2} \right). \tag{27}$$

To impose the reference, θ is considered the tracking error $e_{\theta_1} = s_{\theta_1} - s_{\theta_{1,d}}$ and its derivative

$$\begin{aligned} \dot{e}_{\theta_1} &= \dot{s}_{\theta_1} - \dot{s}_{\theta_{1,d}} \\ \dot{e}_{\theta_1} &= c_{\theta_1}\theta_2 - \dot{s}_{\theta_{1,d}}, \end{aligned} \tag{28}$$

For virtual control, $\theta_{2,d}$ is used

$$\begin{aligned} -k_{\theta_1}e_{\theta_1} - k_{0\theta_1}Ie_{\theta_1} &= c_{\theta_1}\theta_{2,d} - \dot{s}_{\theta_{1,d}}; \\ \theta_{2,d} &= \tfrac{1}{c_\theta}\left(\dot{s}_{\theta_{1,d}} - k_{\theta_1}e_{\theta_1} - k_{0\theta_1}Ie_{\theta_1}\right), \end{aligned} \tag{29}$$

Defining e_{θ_2}

$$\begin{aligned} e_{\theta_2} &= \theta_2 - \theta_{2,d}; \\ \dot{e}_{\theta_2} &= \dot{\theta}_2 - \dot{\theta}_{2,d}, \end{aligned} \tag{30}$$

the dynamics of the error results

$$\dot{e}_{\theta_2} = \frac{J_z - J_x}{J_y}\phi_2\psi_2 + \frac{J_R}{J_y}\phi_2\omega_R + \frac{l}{J_y}u_3 - \dot{\theta}_{2,d}, \tag{31}$$

and finally, the control input u_3 is obtained as

$$u_3 = \frac{J_y}{d}\left(\dot{\theta}_{2,d} - k_{\theta_2}e_{\theta_2} - k_{0\theta_2}Ie_{\theta_2} - \frac{J_z - J_x}{J_y}\phi_2\psi_2 - \frac{J_R}{J_y}\phi_2\omega_R\right), \tag{32}$$

2.4.4. Lateral Movement Control

A lateral movement controller is designed in this section and its operation is along the y-axis and the UAV. Using a procedure similar to that described in Section 2.4.2, the subsystem S_3 is used:

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= \tfrac{1}{m}(c_{\phi_1}s_{\theta_1}s_{\psi_1} - s_{\phi_1}c_{\psi_1})u_1 \\ \dot{\phi}_1 &= \phi_2 \\ \dot{\phi}_2 &= \left(\tfrac{J_y - J_z}{J_x}\right)\theta_2\psi_2 - \tfrac{J_R}{J_x}\theta_2\omega_R + \tfrac{l}{J_x}u_2 \end{aligned} \tag{33}$$

the tracking error and dynamic error are defined by,

$$\begin{aligned} e_{y1} &= y - y_{1,ref} \\ \dot{e}_{y1} &= y_2 - \dot{y}_{1,ref}, \end{aligned} \tag{34}$$

adding an integrative term to the controller $\dot{I}e_{y_1} = e_{y_1}$, the virtual control for $y_{2,d}$ is obtained

$$\begin{aligned} -k_{y_1}e_{y_1} - k_{0y_1}Ie_{y_1} &= y_{2,d} - \dot{y}_{1,ref} \\ y_{2,d} &= \dot{y}_{1,ref} - k_{y_1}e_{y_1} - k_{0y_1}Ie_{y_1}, \end{aligned} \tag{35}$$

differentiation with respect to the time of Equation (35) yields

$$\begin{aligned} \dot{y}_{2,d} &= \ddot{y}_{1,ref} - k_{y_1}\dot{e}_{y_1} - k_{0y_1}\dot{I}e_{y_1} \\ \dot{y}_{2,d} &= \ddot{y}_{1,ref} - k_{y_1}\left(y_2 - \dot{y}_{1,ref}\right) - k_{0y_1}e_{y1}. \end{aligned} \tag{36}$$

Subsequently, the tracking $e_{y_2} = y_2 - y_{2,d}$ is defined, and the dynamic is as follows:

$$\begin{aligned} \dot{e}_{y2} &= \dot{y}_2 - \dot{y}_{2,d}; \\ \dot{e}_{y_2} &= \tfrac{1}{m}\left(c_{\phi_1} s_{\theta_1} s_{\psi_1} - s_{\phi_1} c_{\psi_1}\right) u_1 - \dot{y}_{2,d}. \end{aligned} \tag{37}$$

Now, it clears s_{ϕ_1} to obtain

$$\begin{aligned} -k_{y_2} e_{y_2} - k_{0y_2} I e_{y_2} &= \tfrac{1}{m} c_{\phi_1} s_{\theta_1} s_{\psi_1} u_1 - \tfrac{1}{m} s_{\phi_1} c_{\psi_1} u_1 - \dot{y}_{2,d} \\ s_{\phi_{1,d}} &= \tfrac{m}{c_{\psi_1} u_1}\left(\tfrac{1}{m} c_{\phi_1} s_{\theta_1} s_{\psi_1} u_1 - \dot{y}_{2,d} + k_{y_2} e_{y_2} + k_{0y_2} I e_{y_2}\right). \end{aligned} \tag{38}$$

Considering the angular error $e_{\phi_1} = s_{\phi_1} - s_{\phi_{1,d}}$, its dynamics can be expressed as

$$\begin{aligned} \dot{e}_{\phi_1} &= \dot{s}_{\phi_1} - \dot{s}_{\phi_{1,d}} \\ \dot{e}_{\phi_1} &= c_{\phi_1} \phi_2 - \dot{s}_{\phi_{1,d}}. \end{aligned} \tag{39}$$

On the other hand, for virtual control, $\phi_{2,d}$ is used

$$\begin{aligned} -k_{\phi_1} e_{\phi_1} - k_{0\phi_1} I e_{\phi_1} &= c_{\phi_1} \phi_{2,d} - \dot{s}_{\phi_{1,d}} \\ \phi_{2,d} &= \tfrac{1}{c_\phi}\left(\dot{s}_{\phi_{1,d}} - k_{\phi_1} e_{\phi_1} - k_{0\phi_1} I e_{\phi_1}\right). \end{aligned} \tag{40}$$

Furthermore, with $e_{\phi_2} = \phi_2 - \phi_{2,d}$, the dynamic is defined by

$$\dot{e}_{\phi_2} = \dot{\phi}_2 - \dot{\phi}_{2,d}. \tag{41}$$

Now, using (33) in (41), the angular velocity error dynamic is expressed as

$$\dot{e}_{\phi_2} = \frac{J_y - J_z}{J_x} \theta_2 \psi_2 - \frac{J_R}{J_x} \theta_2 \omega_R + \frac{l}{J_x} u_2 - \dot{\phi}_{2,d}. \tag{42}$$

Finally, the control input u_2 proposed is

$$\begin{aligned} u_2 &= \tfrac{J_x}{l}\left(\dot{e}_{\phi_2} - \tfrac{J_y - J_z}{J_x} \theta_2 \psi_2 + \tfrac{J_R}{J_x} \theta_2 \omega_R + \dot{\phi}_{2,d}\right); \\ u_2 &= \tfrac{J_x}{l}\left(\dot{\phi}_{2,d} - k_{\phi_2} e_{\phi_2} - k_{0\phi_2} I e_{\phi_2} - \tfrac{J_y - J_z}{J_x} \theta_2 \psi_2 + \tfrac{J_R}{J_x} \theta_2 \omega_R\right). \end{aligned} \tag{43}$$

2.4.5. Yaw Motion Control

For rotation control, the subsystem S_4 is considered

$$\begin{aligned} \dot{\psi}_1 &= \psi_2; \\ \dot{\psi}_2 &= \left(\tfrac{J_x - J_y}{J_z}\right) \phi_2 \theta_2 + \tfrac{1}{J_z} u_4, \end{aligned} \tag{44}$$

where the error system is $e_{\psi_1} = \psi_1 - \psi_{1,ref}$, and its dynamic results are

$$\dot{e}_{\psi_1} = \psi_2 - \dot{\psi}_{1,ref}. \tag{45}$$

Defining the virtual control as in the previous sections, $\psi_{2,d}$ is:

$$\psi_{2,d} = \dot{\psi}_{1,ref} - k_{\psi 1} e_{\psi_1} - k_{0\psi_1} I e_{\psi_1}. \tag{46}$$

The differentiation of (46) with respect to time is obtained as

$$\dot{\psi}_{2,d} = \ddot{\psi}_{1,ref} - k_{\psi_1} \dot{e}_{\psi_1} - k_{0\psi_1} \dot{I} e_{\psi_1}. \tag{47}$$

Finally, with the error $e_{\psi 2} = \psi_2 - \psi_{2,d}$, its dynamic is defined as

$$\begin{aligned} \dot{e}_{\psi 2} &= \dot{\psi}_2 - \dot{\psi}_{2,d} \\ \dot{e}_{\psi_2} &= \left(\frac{J_x - J_y}{J_z}\right)\phi_2\theta_2 + \frac{1}{J_z}u_4 - \dot{\psi}_{2,d}, \end{aligned} \tag{48}$$

then the controller input u_4 is defined by

$$u_4 = J_z\left(\dot{\psi}_{2,d} - k_{\psi_2}e_{\psi_2} - k_{0\psi_2}Ie_{\psi_2} - \left(\frac{J_x - J_y}{J_z}\right)\phi_2\theta_2\right). \tag{49}$$

As in Section 2.4.2., the exponential stability can be inferred from the yaw error dynamics, i.e.:

$$\begin{aligned} \begin{pmatrix} \dot{Ie}_{\psi_1} \\ \dot{e}_{\psi_1} \end{pmatrix} &= A_{\psi,1}\begin{pmatrix} Ie_{\psi_1} \\ e_{\psi,1} \end{pmatrix}, \quad A_{\psi,1} = \begin{pmatrix} 0 & 1 \\ -k_{0\psi_1} & -k_{\psi 1} \end{pmatrix} \\ \begin{pmatrix} \dot{Ie}_{\psi_2} \\ \dot{e}_{\psi,2} \end{pmatrix} &= A_{\psi,2}\begin{pmatrix} Ie_{\psi_2} \\ e_{\psi,2} \end{pmatrix}, \quad A_{\psi,2} = \begin{pmatrix} 0 & 1 \\ -k_{0\psi_2} & -k_{\psi_2} \end{pmatrix} \end{aligned} \tag{50}$$

where the integral gains $k_{0\psi_1}$, $k_{0\psi_2}$ and proportional gains k_{ψ_1}, k_{ψ_2} are fixed so that the error dynamics are exponentially stable.

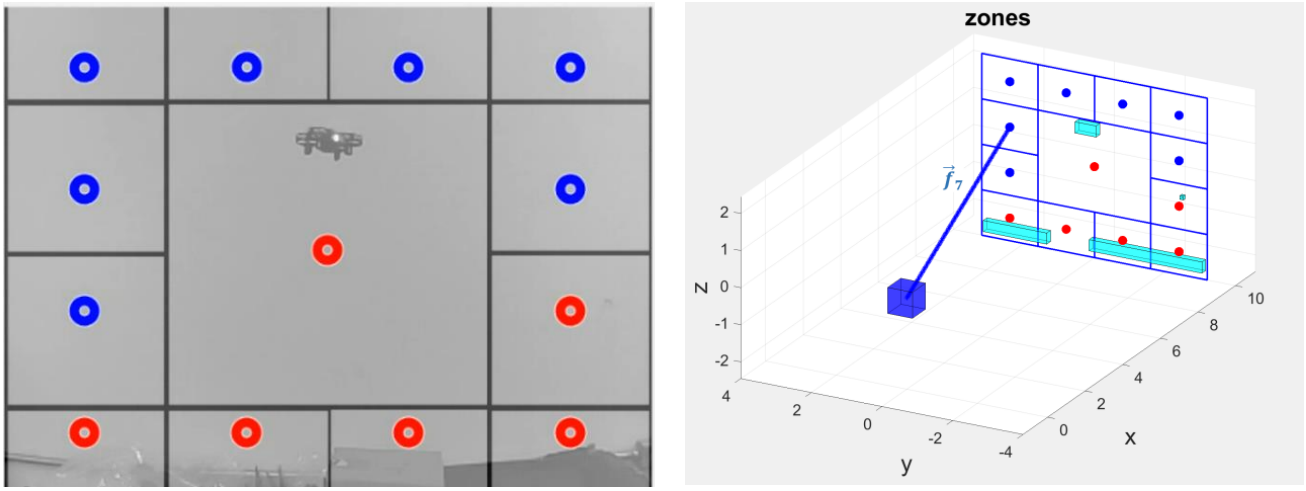
3. Results

Three experiments are documented to show the functionality of our proposal. For each case, the collision-free trajectories are calculated with one frame by the TGBOA algorithm in real-time. Subsequently, a trajectory is selected and employed as a reference for the PI-like controller. MATLAB 2022a is used for the controller simulation because of its compatibility with other platforms and its wide variety of tools and different uses in the industry, allowing the use of the algorithm in later works.

3.1. Experiment 1

Figure 6a shows the video frame transmitted from the UAV to the workstation used for the first experiment. In the digital image, thirteen regions are defined as $R_{1–13}$, and objects are detected in regions R_1 and $R_{9–13}$ whose centers are marked in red. Consequently, collision-free regions are detected from region R_2 to region R_8 (centers are marked in blue). Subsequently, using the procedure described in Section 2.3, the collision-free trajectories between the object and UAV are calculated. The time required for the TGBOA algorithm is 26 ms in this frame. These trajectories are expressed using their vector equations, as seen in Table 3. Then, the trajectory $\vec{f}_7$ is selected, and it is marked in blue. Figure 6b illustrates the reference vector for the controller described in Section 2.4.

On the other hand, the simulation controller results are shown by a graphic in Figure 7: Figure 7a is the result for the position control, including horizontal, frontal, and altitude control; Figure 7b presents the results for the attitude control; Figure 7c shows the control input signal controller; and finally Figure 7d shows a 3D representation of the simulation. In each case, the generated reference is taken and discretized at five points, starting from the origin of the center of the selected free zone determined by the video analysis (see Figure 6a). The interval between each reference point is 5 s. Based on Figure 7, at all points, the PI-like controller reaches its stability in 4 s, and then 25 s is the total time required to complete the reference vector.



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

Figure 6. First experiment. (**a**) Frame acquired from the UAV video transmitted to the workstation, which is divided into regions: free regions marked in blue and collision regions marked in red; (**b**) graphical representation of vector reference $\vec{f}_7$ is selected for the controller, the reference originates from the drone and its endpoint is the central coordinates of the region R_7.

Table 3. Collision-free trajectories generated by object detection for frame shown in Figure 6a.

Region	Vector Equation	Parametric Equations	Symmetric Equations
R_k	$\vec{f}_k = (0,0,0) + t(x, y_k, z_k)$ (m)	$x = x_k t$ $y = y_k t$ (m) $z = z_k t$	$\frac{x}{x_k} = \frac{y}{y_k} = \frac{z}{z_k}$ (m)
R_2	$\vec{f}_2 = (0,0,0) + t(10, -2.4\ ,\ 0.613)$	$x_2 = 10t$ $y_2 = -2.4t$ $z_2 = 1.613t$	$\frac{x_2}{10} = \frac{y_2}{-2.4} = \frac{z_2}{0.613}$
R_3	$\vec{f}_3 = (0,0,0) + t(10, -2.4\ ,\ 1.838)$	$x_3 = 10t$ $y_3 = -2.4t$ $z_3 = 1.838t$	$\frac{x_3}{10} = \frac{y_3}{-2.4} = \frac{z_3}{1.838}$
R_4	$\vec{f}_4 = (0,0,0) + t(10, -0.8\ ,\ 1.838)$	$x_4 = 10t$ $y_4 = -0.8t$ $z_4 = 1.838t$	$\frac{x_4}{10} = \frac{y_4}{-0.8} = \frac{z_4}{1.838}$
R_5	$\vec{f}_5 = (0,0,0) + t(10,\ 0.8\ ,\ 1.838)$	$x_5 = 10t$ $y_5 = 0.8t$ $z_5 = 1.838t$	$\frac{x_5}{10} = \frac{y_5}{0.8} = \frac{z_5}{1.838}$
R_6	$\vec{f}_6 = (0,0,0) + t(10,\ 2.4\ ,\ 1.838)$	$x_6 = 10t$ $y_6 = 2.4t$ $z_6 = 1.838t$	$\frac{x_6}{10} = \frac{y_6}{2.4} = \frac{z_6}{1.838}$
R_7	* $\vec{f}_7 = (0,0,0) + t(10,\ 2.4\ ,\ 0.613)$	$x_7 = 10t$ $y_7 = 2.4t$ $z_7 = 0.613t$	$\frac{x_7}{10} = \frac{y_7}{2.4} = \frac{z_7}{0.613}$

* The vector $\vec{f}_7$ is marked in blue because this vector is used as a reference in the UAV controller.

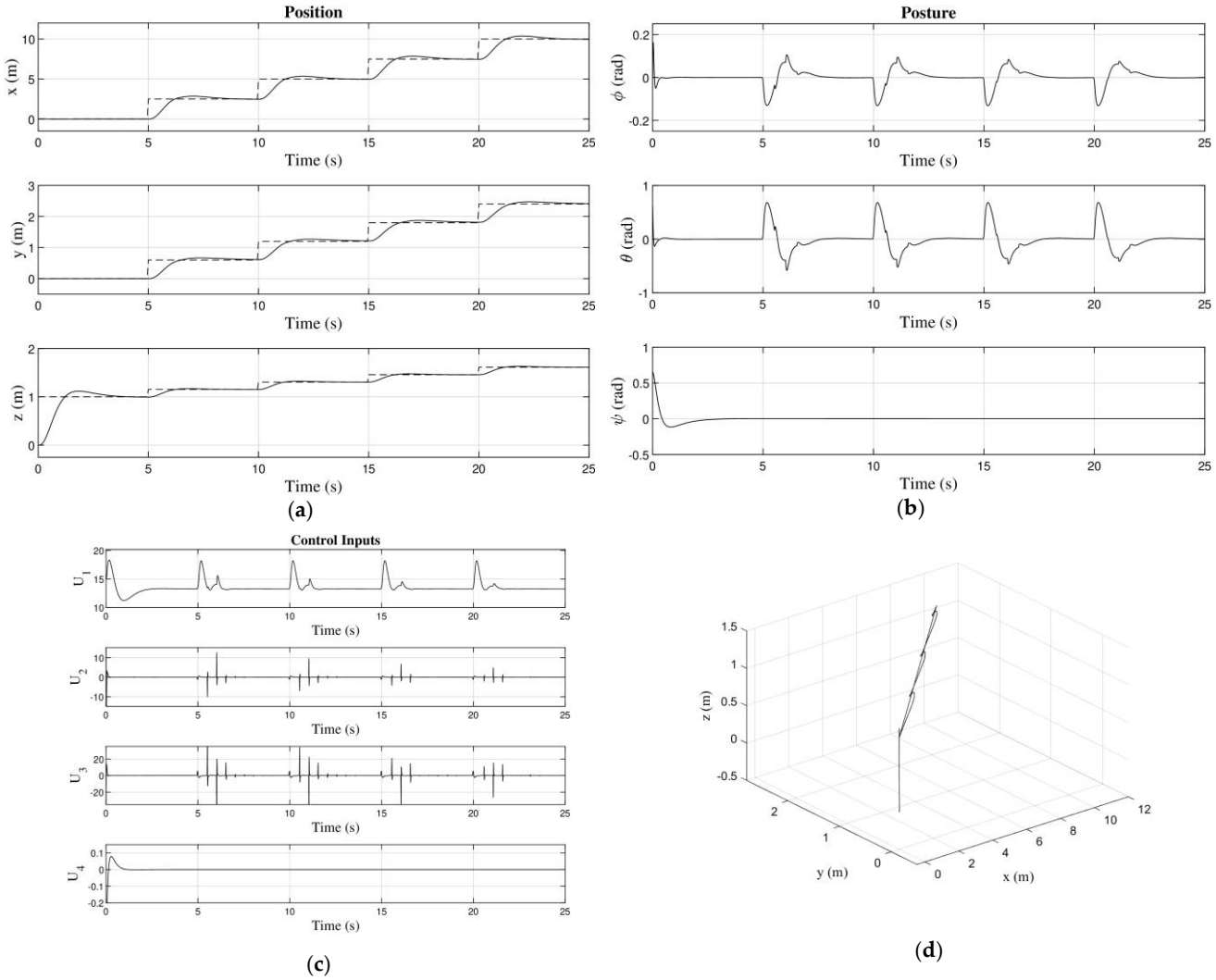
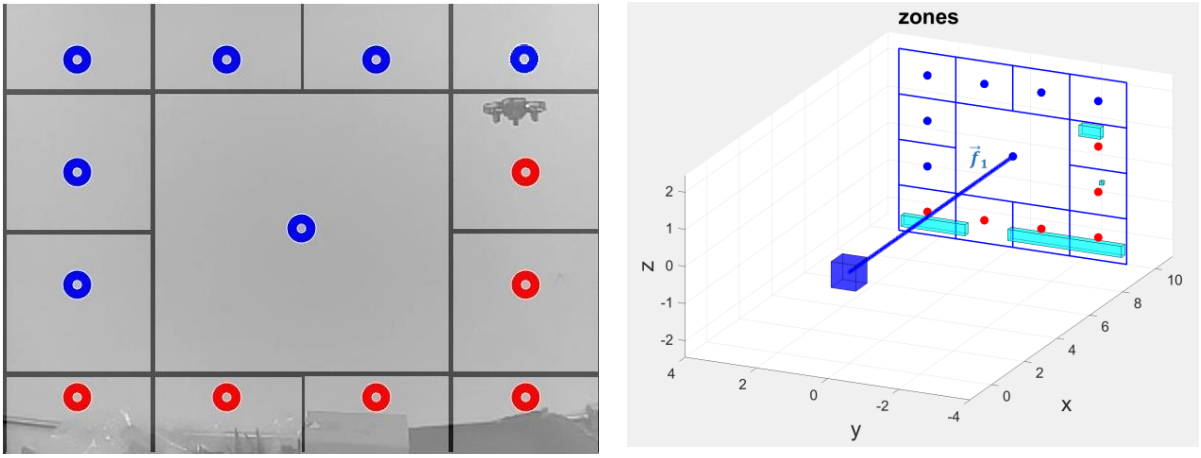


Figure 7. Experimental numerical results obtained when the vector $\vec{f}_7$ is used as a reference: (**a**) motion control; (**b**) attitude control; (**c**) control inputs; (**d**) 3D simulation.

3.2. Experiment 2

For this case, the video frame acquired from the UAV camera is shown in Figure 8a. As observed, there are six regions (R_2, $R_{9–13}$) with detected objects whose centers are marked with red dots, and the free regions found are R_1, $R_{3–8}$ (center marked with a blue dot). Using the UAV position coordinates and the central coordinates of the collision-free regions, trajectories are generated and represented through vector, parametric, and symmetric equations. These equations can be seen in Table 4. Next, the vector $\vec{f}_1$ is used as a reference for the UAV controller, and its graph is displayed in Figure 8b. In this case, the TGBOA algorithm displays the trajectory in 21 ms. On the other hand, Figure 9 shows the results obtained for the second experiment when the vector $\vec{f}_1$ is applied as a reference in the controller.



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

Figure 8. Second experiment. (**a**) Frame used during the second experiment: objects located in regions R_2, $R_{9–13}$ Such regions are identified in red and collision−free regions R_1, $R_{3–8}$ are marked with a blue color; (**b**) graphical representation of vector reference $\vec{f}_1$: the reference starts from the drone and its endpoint is the central coordinates of the region R_1.

Table 4. Collision-free trajectories generated by the TGBOA algorithm using the frame shown in Figure 8a.

Region	Vector Equation	Parametric Equations	Symmetric Equations
R_k	$\vec{f}_k = (0,0,0) + t(x, y_k, z_k)$ (m)	$x = x_k t$ $y = y_k t$ (m) $z = z_k t$	$\frac{x}{x_k} = \frac{y}{y_k} = \frac{z}{z_k}$ (m)
R_1	* $\vec{f}_1 =$ $(0,0,0) + t(10,\ 0\ ,\ 0)$	$x_1 = 10t$ $y_1 = 0t$ $z_1 = 0t$	$\frac{x_1}{10} = \frac{y_1}{0} = \frac{z_1}{0}$
R_3	$\vec{f}_3 = (0,0,0) + t(10, -2.4\ ,\ 1.838)$	$x_3 = 10t$ $y_3 = -2.4t$ $z_3 = 1.838t$	$\frac{x_3}{10} = \frac{y_3}{-2.4} = \frac{z_3}{1.838}$
R_4	$\vec{f}_4 = (0,0,0) + t(10, -0.8\ ,\ 1.838)$	$x_4 = 10t$ $y_4 = -0.8t$ $z_4 = 1.838t$	$\frac{x_4}{10} = \frac{y_4}{-0.8} = \frac{z_4}{1.838}$
R_5	$\vec{f}_5 = (0,0,0) + t(10,\ 0.8\ ,\ 1.838)$	$x_5 = 10t$ $y_5 = 0.8t$ $z_5 = 1.838t$	$\frac{x_5}{10} = \frac{y_5}{0.8} = \frac{z_5}{1.838}$
R_6	$\vec{f}_6 = (0,0,0) + t(10,\ 2.4\ ,\ 1.838)$	$x_6 = 10t$ $y_6 = 2.4t$ $z_6 = 1.838t$	$\frac{x_6}{10} = \frac{y_6}{2.4} = \frac{z_6}{1.838}$
R_7	$\vec{f}_7 = (0,0,0) + t(10,\ 2.4\ ,\ 0.613)$	$x_7 = 10t$ $y_7 = 2.4t$ $z_7 = 0.613t$	$\frac{x_7}{10} = \frac{y_7}{2.4} = \frac{z_7}{0.613}$
R_8	$\vec{f}_8 = (0,0,0) + t(10,\ 2.4\ , -0..613)$	$x_8 = 10t$ $y_8 = 2.4t$ $z_8 = 0.613t$	$\frac{x_8}{10} = \frac{y_8}{2.4} = \frac{z_8}{0.613}$

* The vector $\vec{f}_1$ is marked in blue due to this vector being used as a reference in the UAV controller.

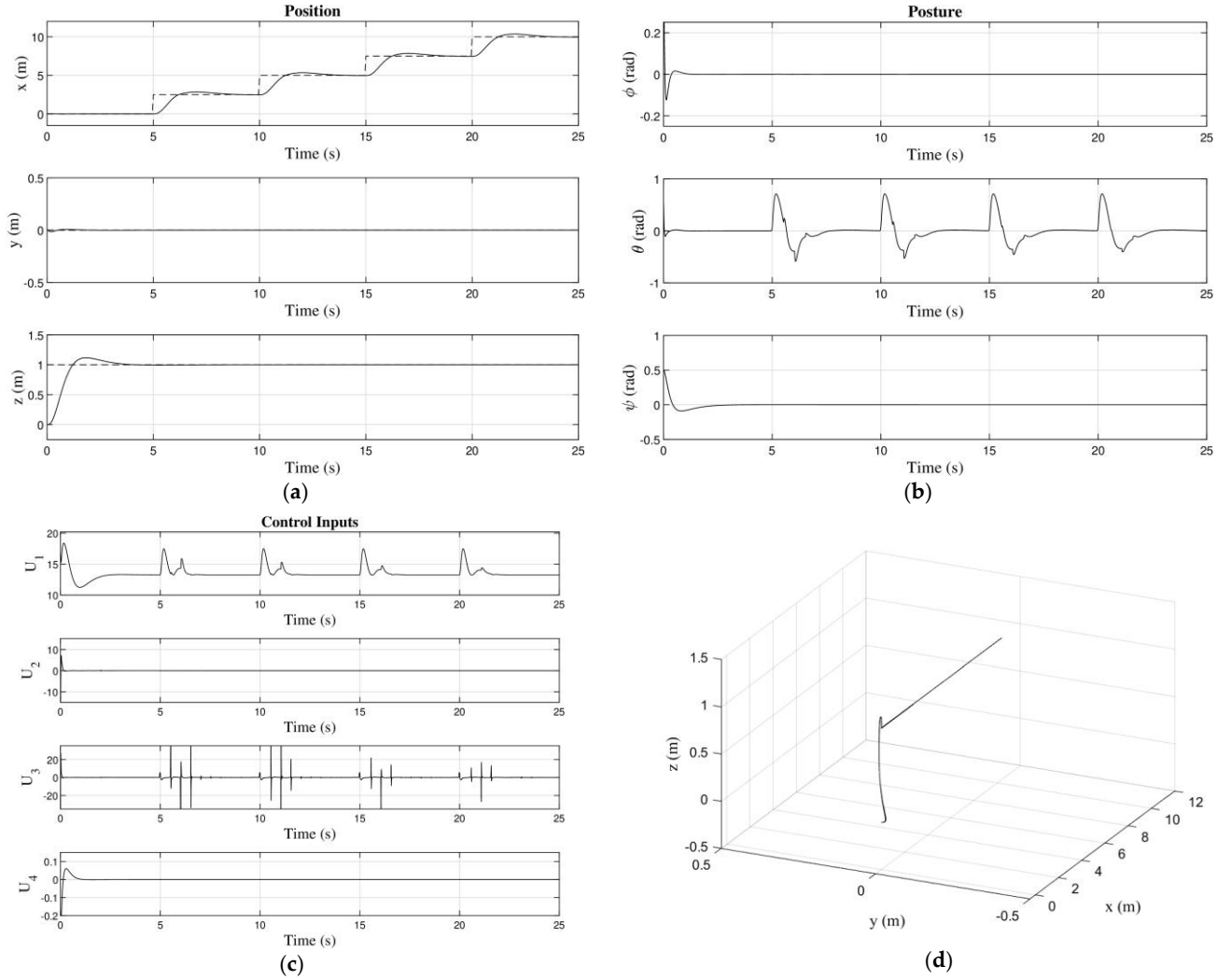
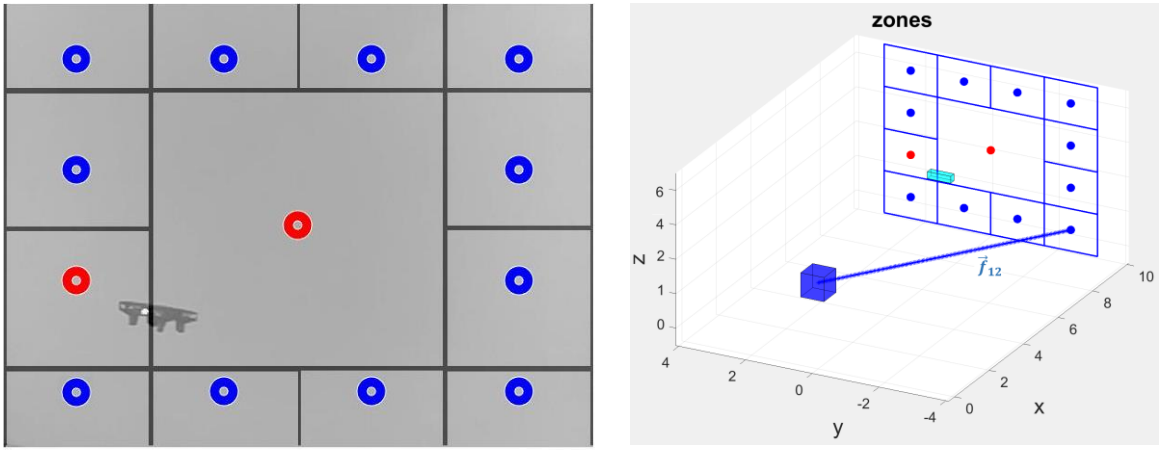


Figure 9. Results obtained when the vector $\vec{f}_1$ is used as a reference: (**a**) motion control; (**b**) attitude control; (**c**) control inputs; (**d**) 3D simulation.

For this experiment, five points are taken from the chosen trajectory with a duration of five seconds between them. Each reference point is reached in 4 s as observed in Figure 9a. The longitudinal movement retains the same behavior as in Experiment 1, while the lateral movement remains constant as the UAV follows a line forward. In addition, the altitude is maintained until the final point is reached. Again, the last point stabilizes after 24 s.

3.3. Experiment 3

This experiment uses the same methodology as the first by moving objects and taking a new frame for the TGBOA algorithm, detecting the position, finding collision-free zones, and generating the optimal trajectory. In Figure 10a, we can see the video frame for Experiment 3, where the objects are in the regions R_1 and R_8 with the centers marked in red. Therefore, the collision-free regions are $R_{2\text{–}7}$ and $R_{9\text{–}13}$, whose centers are marked in blue. Following the procedure described in Section 2.3, the possible trajectories for the UAV are calculated, and each trajectory is represented through a vector, parametric, and symmetric equation. Such equations are observable in Table 5. The references are obtained in 28 ms by the TGBOA algorithm. Subsequently, the trajectory $\vec{f}_{12}$ is used as a reference in the controller (see Figure 10b), and then the numerical results can be seen in Figure 11.



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

Figure 10. Third experiment. (**a**) Frame used in the third experiment: the object is in the regions $R_{2–7}$, and $R_{9–13}$ identified with a red circle and R_1 collision−free regions and are identified R_8 with a blue circle; (**b**) graphical representation of vector reference $\vec{f}_{12}$.

Table 5. Collision-free trajectories generated by the TGBOA algorithm using the frame shown in Figure 10a.

Region	Vector Equation	Parametric Equations	Symmetric Equations
R_k	$\vec{f}_k = (0,0,0) + t(x, y_k, z_k)$ (m)	$x = x_k t$ $y = y_k t$ (m) $z = z_k t$	$\frac{x}{x_k} = \frac{y}{y_k} = \frac{z}{z_k}$ (m)
R_2	$\vec{f}_2 = (0,0,0) + t(10, -2.4\ ,\ 0.613)$	$x_2 = 10t$ $y_2 = -2.4t$ $z_2 = 0.613t$	$\frac{x_2}{10} = \frac{y_2}{-2.4} = \frac{z_2}{0.613}$
R_3	$\vec{f}_3 = (0,0,0) + t(10, -2.4\ ,\ 1.838)$	$x_3 = 10t$ $y_3 = -2.4t$ $z_3 = 1.838t$	$\frac{x_3}{10} = \frac{y_3}{-2.4} = \frac{z_3}{1.838}$
R_4	$\vec{f}_4 = (0,0,0) + t(10, -0.8\ ,\ 1.838)$	$x_4 = 10t$ $y_4 = -0.8t$ $z_4 = 1.838t$	$\frac{x_4}{10} = \frac{y_4}{-0.8} = \frac{z_4}{1.838}$
R_5	$\vec{f}_5 = (0,0,0) + t(10,\ 0.8\ ,\ 1.838)$	$x_5 = 10t$ $y_5 = 0.8t$ $z_5 = 1.838t$	$\frac{x_5}{10} = \frac{y_5}{0.8} = \frac{z_5}{1.838}$
R_6	$\vec{f}_6 = (0,0,0) + t(10,\ 2.4\ ,\ 1.838)$	$x_6 = 10t$ $y_6 = 2.4t$ $z_6 = 1.838t$	$\frac{x_6}{10} = \frac{y_6}{2.4} = \frac{z_6}{1.838}$
R_7	$\vec{f}_7 = (0,0,0) + t(10,\ 2.4\ ,\ 0.613)$	$x_7 = 10t$ $y_7 = 2.4t$ $z_7 = 0.613t$	$\frac{x_7}{10} = \frac{y_7}{2.4} = \frac{z_7}{0.613}$
R_9	$\vec{f}_9 = (0,0,0) + t(10\ ,\ 2.4, -1.838)$	$x_9 = 10t$ $y_9 = 2.4t$ $z_9 = -1.838t$	$\frac{x_9}{10} = \frac{y_9}{2.4} = \frac{z_9}{-1.838}$

Table 5. *Cont.*

Region	Vector Equation	Parametric Equations	Symmetric Equations
R_{10}	$\vec{f}_{10} = (0,0,0) + t(10,\ 0.8, -1.838)$	$x_{10} = 10t$ $y_{10} = 0.8t$ $z_{10} = -1.838t$	$\frac{x_{10}}{10} = \frac{y_{10}}{0.8} = \frac{z_{10}}{-1.838}$
R_{11}	$\vec{f}_{11} = (0,0,0) + t(10, -0.8, -1.838)$	$x = 10t$ $y_2 = -0.8t$ $z_{11} = -1.838t$	$\frac{x_{11}}{10} = \frac{y_{12}}{-0.8} = \frac{z_{12}}{-1.838}$
R_{12}	$* \vec{f}_{12} = (0,0,0) + t(10, -2.4, -1.838)$	$x_{12} = 10t$ $y_{12} = -2.4t$ $z_{12} = -1.838$	$\frac{x_{12}}{10} = \frac{y_{12}}{-2.4} = \frac{z_{12}}{-1.838}$
R_{13}	$\vec{f}_{13} = (0,0,0) + t(10, -2.4, -0.613)$	$x_{13} = 10t$ $y_{13} = -2.4t$ $z_{13} = -0.613t$	$\frac{x_{13}}{10} = \frac{y_{13}}{-2.4} = \frac{z_{13}}{-0.613}$

* The vector $\vec{f}_{12}$ is marked in blue due to this vector being used as a reference in the UAV controller.

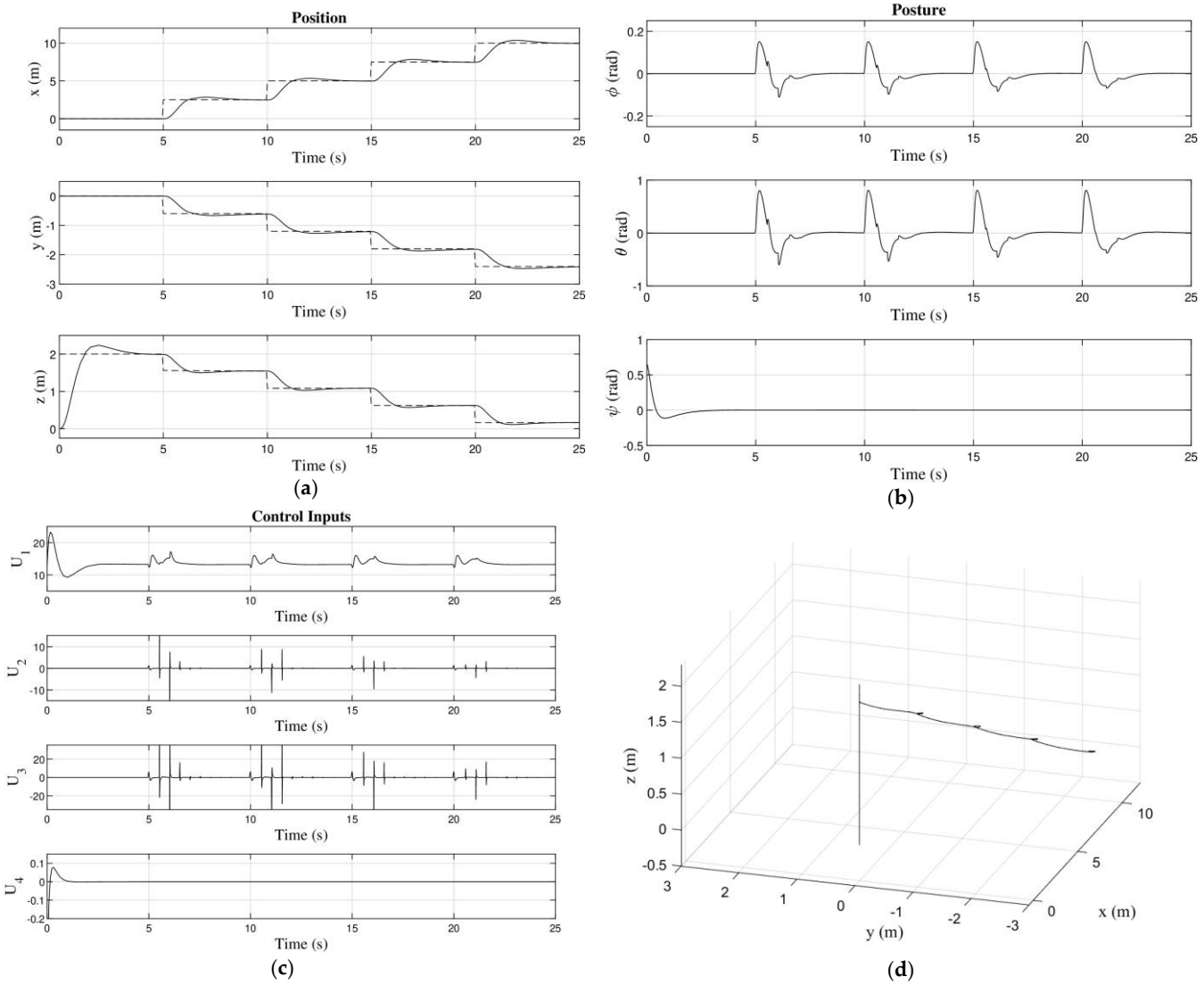


Figure 11. Results obtained by the third experiment when the vector $\vec{f}_{12}$ is used as a reference: (**a**) motion control; (**b**) attitude control; (**c**) control inputs; (**d**) 3D simulation.

For the last experiment presented in this paper, the trajectory pointing towards zone number 12 is used, and the simulation controller results are shown in Figure 11, where the dotted line represents the reference, and the line represents the numerical results. It is observed that the PI-like controller is stable since it reaches each point of the trajectory in approximately 4 s and it is maintained until the next point, reaching the center of the collision-free zone in 25 s.

3.4. Object Avoidance

With the results obtained in the three experiments (Section 3.1–Section 3.3), the behavior graph of object avoidance is shown in Figure 12.

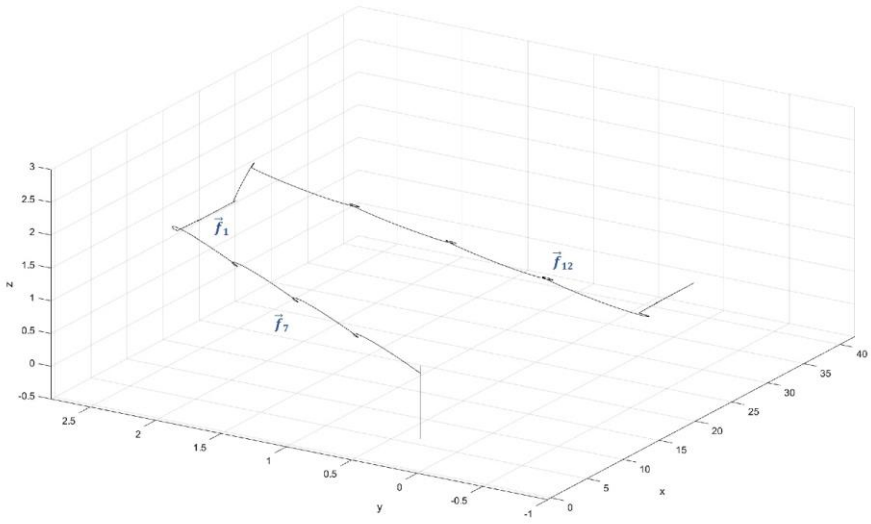


Figure 12. Trajectory generated by TGBOA algorithm results of three experiments and their use in the PI-like controller.

Analyzing Figure 12 reveals the UAV travels in an initial charged trajectory in the controller, until an object is detected with the vision sensing system using the TGBOA algorithm to create the new trajectory. Then, the new trajectory is added to the PI-like controller. Note that the new trajectory $\vec{f}_7$ (Experiment 1) evades the object located during the flight, as observed in Figure 6. This new trajectory $\vec{f}_7$ is followed until the visual sensing systems detect other objects in the UAV trajectory, and as a consequence, the TGBOA algorithm creates a new trajectory again $\vec{f}_1$(Experiment 2), and then is added to the PI-like controller, updating the UAV with the new objects in the scenario. The $\vec{f}_1$ makes the drone to evade in accordance with the new values added to the algorithm. Notice that the UAV follows the $\vec{f}_1$ until the vision sensing system detects objects in the scenario, and then it is necessary to create a new trajectory with the TGBOA algorithm; in this case, the trajectory vector $\vec{f}_{12}$ (Experiment 3) is created. Please refer to Figure 11 and Table 5. As in the other cases, $\vec{f}_{12}$ allows the drone to evade objects thanks to a change in the direction or trajectory.

Therefore, considering the results shown in Figure 12, the TGBOA algorithm creates new trajectories in real-time using the evading of objects. These are detected with the visual sensing system. The trajectory vectors are then added to the controller, and with this new information the drone moves to a new position.

4. Discussion

In this work, a vision system was applied for object detection to generate real-time trajectories for drones and was probed as references in a newly designed PI-like controller.

A new algorithm called trajectory generation based on object avoidance (TGBOA) was reported. In the TGBOA algorithm, the trajectories were created based on object evasion and using the drone position and the central coordinates of collision-free regions. Its mathematical representation was calculated and shown on vectors, parametric, and symmetric equations. In each experiment, one vector was selected and used as a reference in the PI-like controller (see Section 2.4) by a simulation software. The numerical results obtained verify the operation of our proposal since the drone always reached the reference generated by the TGBOA algorithm. Based on the results, the following points can be highlighted:

1. The new trajectory generation based on object avoidance (TGBOA) algorithm is proposed to generate real-time trajectories for drones. The TGBOA algorithm is based on evading detected objects using a vision sensing system.
2. The TGBOA algorithm is experimentally checked since the trajectory vectors were generated from a video analysis. Then, they are applied as references in the designated PI-like controller.
3. Our new proposal generates new trajectories in real-time for drone controllers, taking into consideration changing scenarios because of its movement or random natural scenarios.
4. The TGBOA algorithm is more efficient in the creation of new drones' trajectories.
5. The TGBOA algorithm is easy to implement because it is based on the detection of objects in a changing scenario, and it is not based on the optic flow concept [35]. As a consequence, the number of computational operations and the mathematical complexity are reduced.
6. With the objects evading model from the TGBOA algorithm, the drone can be more autonomous when it travels from an initial point f_A to a final point f_B, even though this is a changing scenario.
7. The TGBOA algorithm can be implemented in specialized development boards.

However, due to the applied software, the data transmission, and the controller, time limitations take place in the execution such as:

1. For each frame, the TGBOA algorithm creates a new trajectory in 30 ms.
2. Including the transmission time between the UAV and the workstation, the total time to produce and update a trajectory is up to 390 ms, due to the delay in the Wi-Fi network. Then, the data transmission time is 360 ms.
3. Experimentally, the execution time is long since the drone's controller requires up to 25 s to reach the center of the chosen free-collision zone.
4. The total execution time is experimentally 25.360 s.
5. MATLAB 2022a is used due to its compatibility with other platforms and its wide variety of tools and different uses in the industry, allowing the use of the algorithm in future projects.

Comparing our trajectory generation proposal (TGBOA algorithm) with those reported in the references, the TGBOA algorithm is functional for random and dynamic scenarios, is based on object evasion, operates in real-time, and can work online. However, in the reported techniques, the surrounding environment cannot change, the controller references must be preloaded, and the trajectory works offline. Based on the above, the TGBOA algorithm offers advantages compared to the previously described techniques, due to how it operates in changing and random situations.

Comparing the TGBOA algorithm with the proposals reported in the references [7,19,24,25], the TGBOA algorithm reduces complexity because our work is based on the detection of moving objects, while the indicated references are based on the calculation of optical flow, which is calculated through the frequency components of the image or optimization algorithms. Table 6 shows the comparison between the TGBOA algorithm and the references where the trajectory of the drone is determined by applying the optical flow technique.

Table 6. Comparison between related work and TGBOA algorithm.

	Relevant Aspect	Our Proposal
[19]	High image sampling and position control signal rate. Expensive hardware and requires prior knowledge or estimation of the objects in its spatial environment.	Only needs a vision sensor, and online trajectory references are generated and updated in real-time.
[23]	Large amount of computational resources used for CNN. Three vision sensors needed. High performance boards used to improve the performance.	Improves efficiency by focusing the algorithm on object detection and drone position, thus reducing computational complexity.
[24]	Updating environment requires a lot of time and computer resources. Dynamic scenarios are not considered.	Focuses on the position of moving objects and dynamic scenarios without relying on full scenario or feature extraction.
[25]	The optical flow constructed from the image sequence makes it difficult to achieve the real-time performance on the low-cost hardware. The presence of moving obstacles is not considered.	Easy to implement because it is based on the detection of objects in a changing scenario and not based on the optic flow concept with Fourier transform [38].

Based on Tables 3–5 and Figures 5–12, the TGBOA algorithm works correctly in the generation of trajectories in real-time only if the vision system is correctly applied in the evasion of objects. However, the runtime is high due to the software used in our experiments. Therefore, our future work has the following directions: the first is to reduce the execution time using electronic development cards, the second is to improve the transmission data rate, and the third is to apply the TGBOA algorithm when the UAV will move through random and changing scenarios.

5. Conclusions

The new algorithm called trajectory generation based on object avoidance (TGBOA) is capable of flight planning in real-time from object detection by a vision sensor and with low computational resources by processing a video transmitted from a UAV to a workstation. The algorithm can generate collision-free trajectories that can be used as a reference in the UAV controller to avoid objects during the flight. However, the optimal trajectory will depend on the detected objects, UAV positions, and the endpoint. Thirteen regions were defined in the image for the video analysis. This number can be increased to have more trajectories generated in different directions; however, as a consequence, the execution time increases due to the number of computational operations required. This problem can be solved by programming the TGBOA algorithm on electronic cards with greater computational power and using different platforms dedicated to robotic systems. A PI-like controller was developed for a quadcopter position and yaw control. The dynamics of the UAV have been divided into four subsystems with a control input and an output variable each. Using the references obtained from the TGBOA algorithm, the flight controller led to satisfactory results in the experiments, due to where the objects were located and evaded in different areas and choosing the appropriate trajectory for each case. The numerical simulations of the proposed controllers have been implemented using the mathematical software MATLAB 2022a, and the results show a good performance of the proposed control law, reaching the proposed references in 4 s. Future work will concern the implementation of the algorithm and different controllers using other simulation platforms and development boards.

Author Contributions: Conceptualization, J.T.G.B., C.A.L., S.D.G. and G.G.-T.; methodology, J.T.G.B., C.A.L. and S.D.G.; software, L.F.M.M.; validation, L.F.M.M., C.A.L. and J.T.G.B.; formal analysis, J.T.G.B. and C.A.L.; investigation, L.F.M.M. and J.T.G.B.; resources, J.T.G.B. and C.A.L.; data curation, L.F.M.M.; writing—original draft preparation, L.F.M.M.; writing—review and editing, J.T.G.B.,

C.A.L., S.D.G. and G.G.-T.; methodology, J.T.G.B., C.A.L., S.D.G. and G.G.-T.; visualization, L.F.M.M.; supervision, G.G.-T., J.T.G.B. and C.A.L.; project administration, J.T.G.B. and C.A.L.; funding acquisition, L.F.M.M., J.T.G.B. and C.A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The authors thank Mexico's National Council of Science and Technology (CONACyT) and the University of Guadalajara for the support granted. Luis Felipe Muñoz Mendoza also thanks CONACyT for the scholarship.

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

References

1. Alzahrani, B.; Oubbati, O.S.; Barnawi, A.; Atiquzzaman, M.; Alghazzawi, D. UAV assistance paradigm: State-of-the-art in applications and challenges. *J. Netw. Comput. Appl.* **2020**, *166*, 102706. [CrossRef]
2. Tsouros, D.C.; Bibi, S.; Sarigiannidis, P.G. A review on UAV-based applications for precision agriculture. *Information* **2019**, *10*, 349. [CrossRef]
3. Woo, J.W.; An, J.-Y.; Cho, M.G.; Kim, C.-J. Integration of path planning, trajectory generation and trajectory tracking control for aircraft mission autonomy. *Aerosp. Sci. Technol.* **2021**, *118*, 107014. [CrossRef]
4. Ahmed, F.; Jenihhin, M. A Survey on UAV Computing Platforms: A Hardware Reliability Perspective. *Sensors* **2022**, *22*, 6286. [CrossRef]
5. Ahmed, F.; Mohanta, J.C.; Keshari, A.; Yadav, P.S. Recent Advances in Unmanned Aerial Vehicles: A Review. *Arab. J. Sci. Eng.* **2022**, *47*, 7963–7984. [CrossRef] [PubMed]
6. Gupta, A.; Fernando, X. Simultaneous Localization and Mapping (SLAM) and Data Fusion in Unmanned Aerial Vehicles: Recent Advances and Challenges. *Drones* **2022**, *6*, 85. [CrossRef]
7. Chen, Y.; Yu, J.; Mei, Y.; Zhang, S.; Ai, X.; Jia, Z. Trajectory optimization of multiple quad-rotor UAVs in collaborative assembling task. *Chin. J. Aeronaut.* **2016**, *29*, 184–201. [CrossRef]
8. Li, L.; Sun, L.; Jin, J. Survey of advances in control algorithms of quadrotor unmanned aerial vehicle. In Proceedings of the 2015 IEEE 16th International Conference on Communication Technology (ICCT), Hangzhou, China, 18–21 August 2015; pp. 107–111. [CrossRef]
9. Li, Y.; Song, S. A survey of control algorithms for Quadrotor Unmanned Helicopter. In Proceedings of the 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), Nanjing, China, 18–20 October 2012; pp. 365–369. [CrossRef]
10. Antonio-Toledo, M.E.; Sanchez, E.N.; Alanis, A.Y. Neural Inverse Optimal Control Applied to Quadrotor UAV. In Proceedings of the 2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Guadalajara, Mexico, 7–9 November 2018; pp. 1–8. [CrossRef]
11. Guillén-Bonilla, J.T.; García, C.C.V.; Di Gennaro, S.; Morales, M.E.S.; Lúa, C.A. Vision-Based Nonlinear Control of Quadrotors Using the Photogrammetric Technique. *Math. Probl. Eng.* **2020**, *2020*, 5146291. [CrossRef]
12. Rahman, Y.A.A.; Hajibeigy, M.T.; Al-Obaidi, A.S.M.; Cheah, K.H. Design and Fabrication of Small Vertical-Take-Off-Landing Unmanned Aerial Vehicle. *MATEC Web Conf.* **2018**, *152*, 02023. [CrossRef]
13. Gupta, N.; Chauhan, R.; Chadha, S. Unmanned Aerial Vehicle (UAV) for Parcel Delivery. *Int. J. Eng. Res. Technol.* **2020**, *13*, 2824–2830. [CrossRef]
14. Yang, H.; Lee, Y.; Jeon, S.; Lee, D. Multi-rotor drone tutorial: Systems, mechanics, control and state estimation. *Intell. Serv. Robot.* **2017**, *10*, 79–93. [CrossRef]
15. Eslamiat, H.; Li, Y.; Wang, N.; Sanyal, A.K.; Qiu, Q. Autonomous Waypoint Planning, Optimal Trajectory Generation and Nonlinear Tracking Control for Multi-rotor UAVs. In Proceedings of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019. [CrossRef]
16. Shiller, Z. Off-Line and On-Line Trajectory Planning. In *Motion and Operation Planning of Robotic Systems. Mechanisms and Machine Science*; Springer: Cham, Switzerland, 2015; Volume 29, pp. 29–62.
17. Muñoz, J.; López, B.; Quevedo, F.; Monje, C.A.; Garrido, S.; Moreno, L.E. Multi UAV Coverage Path Planning in Urban Environments. *Sensors* **2021**, *21*, 7365. [CrossRef]
18. Youn, W.; Ko, H.; Choi, H.S.; Choi, I.H.; Baek, J.-H.; Myung, H. Collision-free Autonomous Navigation of A Small UAV Using Low-cost Sensors in GPS-denied Environments. *Int. J. Control Autom. Syst.* **2021**, *19*, 953–968. [CrossRef]
19. Chuang, H.-M.; He, D.; Namiki, A. Autonomous Target Tracking of UAV Using High-Speed Visual Feedback. *Appl. Sci.* **2019**, *9*, 4552. [CrossRef]
20. Causa, F.; Fasano, G. Multiple UAVs trajectory generation and waypoint assignment in urban environment based on DOP maps. *Aerosp. Sci. Technol.* **2021**, *110*, 106507. [CrossRef]

21. Farid, G.; Mo, H.; Zahoor, M.I.; Liwei, Q. Computationally efficient algorithm to generate a waypoints-based trajectory for a quadrotor UAV. In Proceedings of the 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 9–11 June 2018; pp. 4414–4419. [CrossRef]
22. Shirzadeh, M.; Asl, H.J.; Amirkhani, A.; Jalali, A.A. Vision-based control of a quadrotor utilizing artificial neural networks for tracking of moving targets. *Eng. Appl. Artif. Intell.* **2017**, *58*, 34–48. [CrossRef]
23. Zhilenkov, A.A.; Epifantsev, I.R. The use of convolution artificial neural networks for drones autonomous trajectory planning. In Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow, Russia, 29 January–1 February 2018; pp. 1044–1047. [CrossRef]
24. Alzugaray, I.; Teixeira, L.; Chli, M. Short-term UAV path-planning with monocular-inertial SLAM in the loop. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2017), Singapore, 29 May–3 June 2017; pp. 2739–2746. [CrossRef]
25. Lin, H.-Y.; Peng, X.-Z. Autonomous Quadrotor Navigation with Vision Based Obstacle Avoidance and Path Planning. *IEEE Access* **2021**, *9*, 102450–102459. [CrossRef]
26. Mo, H.; Farid, G. Nonlinear and Adaptive Intelligent Control Techniques for Quadrotor UAV–A Survey. *Asian J. Control.* **2018**, *21*, 989–1008. [CrossRef]
27. Leon, J.A.R.; Lua, C.A.; Morales, M.E.S.; Di Gennaro, S.; Guzman, A.N. Altitude and attitude non linear controller applied to a quadrotor with a slung load. In Proceedings of the IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), Ixtapa, Mexico, 13–15 November 2019; pp. 1–6. [CrossRef]
28. Raiesdana, S. Control of quadrotor trajectory tracking with sliding mode control optimized by neural networks. *Proc. Inst. Mech. Eng. Part I: J. Syst. Control. Eng.* **2020**, *234*, 1101–1119. [CrossRef]
29. Wu, W.; Jin, X.; Tang, Y. Vision-based trajectory tracking control of quadrotors using super twisting sliding mode control. *Cyber-Phys. Syst.* **2020**, *6*, 207–230. [CrossRef]
30. Shankaran, V.P.; Azid, S.I.; Mehta, U.; Fagiolini, A. Improved Performance in Quadrotor Trajectory Tracking Using MIMO PI$^{\lambda}$-D Control. *IEEE Access* **2022**, *10*, 110646–110660. [CrossRef]
31. Kumar, R.; Bhargavapuri, M.; Deshpande, A.M.; Sridhar, S.; Cohen, K.; Kumar, M. Quaternion Feedback Based Autonomous Control of a Quadcopter UAV with Thrust Vectoring Rotors. In Proceedings of the American Control Conference (ACC), Denver, CO, USA, 1–3 July 2020. [CrossRef]
32. Benhadhria, S.; Mansouri, M.; Benkhlifa, A.; Gharbi, I.; Jlili, N. VAGADRONE: Intelligent and Fully Automatic Drone Based on Raspberry Pi and Android. *Appl. Sci.* **2021**, *11*, 3153. [CrossRef]
33. Xing, H.; Zhu, L.; Chen, B.; Liu, C.; Niu, J.; Li, X.; Feng, Y.; Fang, W. A comparative study of threshold selection methods for change detection from very high-resolution remote sensing images. *Earth Sci. Informatics* **2022**, *15*, 369–381. [CrossRef]
34. Hughes, P.C. *Spacecraft Attitude Dynamics*; Dover Publications, Inc.: Mineola, NY, USA, 1986.
35. Nagaty, A.; Saeedi, S.; Thibault, C.; Seto, M.; Li, H. Control and Navigation Framework for Quadrotor Helicopters. *J. Intell. Robot. Syst.* **2012**, *70*, 1–12. [CrossRef]
36. Tayebi, A.; McGilvray, S. Attitude stabilization of a VTOL quadrotor aircraft. *IEEE Trans. Control. Syst. Technol.* **2006**, *14*, 562–571. [CrossRef]
37. Guzman, A.N.; Di Gennaro, S.; Dominguez, J.R.; Lua, C.A.; Loukianov, A.G.; Castillo-Toledo, B. Enhanced Discrete-Time Modeling via Variational Integrators and Digital Controller Design for Ground Vehicles. *IEEE Trans. Ind. Electron.* **2016**, *63*, 6375–6385. [CrossRef]
38. Cheng, H.-W.; Chen, T.-L.; Tien, C.-H. Motion Estimation by Hybrid Optical Flow Technology for UAV Landing in an Unvisited Area. *Sensors* **2019**, *19*, 1380. [CrossRef]