machines

Article

An Advanced IBVS-Flatness Approach for Real-Time Quadrotor Navigation: A Full Control Scheme in the Image Plane

Ahmed Alshahir [1], Khaled Kaaniche [1,*], Mohammed Albekairi [1], Shahr Alshahr [1], Hassen Mekki [2], Anis Sahbani [3] and Meshari D. Alanazi [1]

[1] Department of Electrical Engineering, College of Engineering, Jouf University, Sakakah 72388, Saudi Arabia; aaalshahir@ju.edu.sa (A.A.); msalbekairi@ju.edu.sa (M.A.); saalshahr@ju.edu.sa (S.A.); mdalsayer@ju.edu.sa (M.D.A.)
[2] NOCCS Laboratory, National School of Engineering of Sousse, University of Sousse, Sousse 4054, Tunisia; hassen.mekki@eniso.u-sousse.tn
[3] Institute for Intelligent Systems and Robotics (ISIR), Centre National de la Recherche Scientifique, Sorbonne University, 75006 Paris, France; anis.sahbani@sorbonne-universite.fr
* Correspondence: kkaaniche@ju.edu.sa

Abstract: This article presents an innovative method for planning and tracking the trajectory in the image plane for the visual control of a quadrotor. The community of researchers working on 2D control widely recognizes this challenge as complex, because a trajectory defined in image space can lead to unpredictable movements of the robot in Cartesian space. While researchers have addressed this problem for mobile robots, quadrotors continue to face significant challenges. To tackle this issue, the adopted approach involves considering the separation of altitude control from the other variables, thus reducing the workspace. Furthermore, the movements of the quadrotor (pitch, roll, and yaw) are interdependent. Consequently, the connection between the inputs and outputs cannot be reversed. The task complexity becomes significant. To address this issue, we propose the following scenario: When the quadrotor is equipped with a downward-facing camera, flying at high altitude is sensible to spot a target. However, to minimize disturbances and conserve energy, the quadrotor needs to descend in altitude. This can result in the target being lost. The solution to this problem is a new methodology based on the principle of differential flatness, allowing the separation of altitude control from the other variables. The system first detects the target at high altitude, then plots a trajectory in the image coordinate system between the acquired image and the desired image. It is crucial to emphasize that this step is performed offline, ensuring that the image processing time does not affect the control frequency. Through the proposed trajectory planning, complying with the constraints of differential flatness, the quadrotor can follow the imposed dynamics. To ensure the tracking of the target while following the generated trajectory, the proposed control law takes the form of an Image Based Visual Servoing (IBVS) scheme. We validated this method using the RVCTOOLS environment in MATLAB. The DJI Phantom 1 quadrotor served as a testbed to evaluate, under real conditions, the effectiveness of the proposed control law. We specifically designed an electronic card to transfer calculated commands to the DJI Phantom 1 control joystick via Bluetooth. This card integrates a PIC18F2520 microcontroller, a DAC8564 digital-to-analogue converter, and an RN42 Bluetooth module. The experimental results demonstrate the effectiveness of this method, ensuring the precise tracking of the target as well as the accurate tracking of the path generated in the image coordinate system.

Keywords: advanced guidance and navigation; IBVS; path planning; path tracking; quadrotor; flatness control; DJI Phantom 1

check for updates

Citation: Alshahir, A.; Kaaniche, K.; Albekairi, M.; Alshahr, S.; Mekki, H.; Sahbani, A.; Alanazi, M.D. An Advanced IBVS-Flatness Approach for Real-Time Quadrotor Navigation: A Full Control Scheme in the Image Plane. *Machines* **2024**, *12*, 350. https://doi.org/10.3390/machines12050350

Academic Editor: Endrowednes Kuantama

Received: 22 March 2024
Revised: 15 May 2024
Accepted: 16 May 2024
Published: 19 May 2024

1. Introduction

Visual servoing, applied to drone navigation, is an approach that combines computer vision and control theory. This integration allows the quadrotor to navigate autonomously with high accuracy and efficiency. Through the utilization of visual feedback obtained from cameras installed onboard, drones are able to continuously adapt their position and orientation in relation to visual landmarks or objects of interest within their surroundings. This advanced technology enables unmanned aerial vehicles (UAVs), commonly known as drones, to execute intricate operations such as accurate object tracking, real-time obstacle avoidance, and precise maneuvering in environments where GPS signals are unavailable or obstructed by obstacles. Visual servoing algorithms facilitate the ability of drones to make instantaneous decisions by utilizing the visual data they perceive. This enables the drones to navigate in a flexible and adaptable manner in different situations. This capability has significant implications across various industries, such as surveillance, agriculture, infrastructure inspection, and search and rescue missions. Drones equipped with visual servoing technology can operate with improved autonomy and effectiveness, as evidenced by studies [1–4]. Drones equipped with state-of-the-art visual servoing techniques have the capability to monitor traffic patterns, thereby ensuring improved traffic flow and increased safety. Within the domain of cartography and geographical surveying [5,6], these drones possess the capability to traverse intricate landscapes, acquiring high-definition visual data for meticulous mapping purposes. This functionality is also utilized in the field of agriculture [7,8], where unmanned aerial vehicles (UAVs) aid in the process of crop monitoring and precision farming.

Generally, the methods in visual servoing are categorized into two families: 3D visual servoing or Position-Based Visual Servoing (PBVS) and 2D visual servoing or Image-Based Visual Servoing (IBVS) [9]. PBVS is a technique that aims to estimate the relative pose of visual targets in a three-dimensional (3D) space. This estimation allows for the accurate control of the drone's position. However, IBVS operates directly in the image coordinate system (the image plane) by utilizing image feature errors to calculate the control commands. The utilization of image feature errors in the IBVS-based approaches eliminates the necessity for any prior knowledge regarding the geometry of the target. IBVS-based approaches reduce the complexity of calculations and improve the resilience of the system. The ability to regulate the drone/camera system without reconstructing the target's relative pose gives the IBVS-based approaches a significant advantage.

Several works combining visual sensors with UAVs have been proposed. The authors of [10] employ perspective points as a means of directing a quadrotor through enclosed passageways while avoiding collisions with the surrounding walls. The authors of [11–14] used, respectively, monocular, stereo, and RGB-D cameras to acquire comprehensive 3D data pertaining to the surrounding environment. The "teach and repeat" approach, which involves capturing, storing, and organizing significant visual representations of the surroundings to facilitate robot navigation, has been the subject of several research studies [15–19]. In order to achieve the tracking of the visual trajectory of quadrotors in indoor environments, a recent study proposed a hierarchical visual servo control scheme. This scheme effectively handles collision avoidance, visibility, and visual tasks in a hierarchical manner, as outlined in [20].

Elementary geometric shapes in the 2D space of the image, such as points of interest, lines, ellipses, cylinders, and projective invariants are used as visual primitives in these methods. However, matching the primitives in images can be a complex task. For example, if we use the coordinates of four points of interest in the image as the primitives, it would be difficult to directly propose a desired trajectory, as this would require proposing four distinct trajectories. This does not necessarily reflect the user's concerns regarding the movements made in Cartesian space. The dynamics of a quadrotor (the UAV considered in this work) are very fast and nervous. In order to efficiently control this specific system, it is imperative to create a regulator with a high-frequency capability [21]. In contrast, visual servoing involves a preliminary stage of image processing that is designed to extract the object's distinctive features. This phenomenon may have a negative impact on the rate of progression of the control law. Furthermore, the complex nature of image processing may lead to temporal lags in motion control, potentially leading to the loss of targets. In the context of a conventional IBVS-based system, it should be noted that configuring the kinematic tensor of the quadrotor does not provide a guarantee of its successful execution in the physical domain. This is because of problems with controllability, especially when the system is not fully activated. In contrast to the quadrotor, which only has four inputs, the kinematic tensor has a dimension of six. It is extremely difficult to complete all six commands generated by the visual servo algorithm using only four control inputs.

The concept of differential flatness introduced and developed by Fliess [22] simplifies the dynamic behavior modeling of a system by identifying a set of fundamental variables called "flat outputs". As we will see in the rest of this paper, this approach has numerous interesting consequences regarding system control. Firstly, it redirects the process control towards the concept of a path that the system must adhere to. In other words, the motion required from a system has to fit the system's capabilities, avoiding many problems that control experts often face. Generating an appropriate desired trajectory is a key stage in flatness-based control. This trajectory takes into consideration the system's model in an implicit manner.

Typically, when a quadrotor is equipped with a downward-facing camera, conventional logic would suggest flying at a high altitude to detect a target. However, to minimize disturbances and save energy, the quadrotor is compelled to descend in altitude, potentially resulting in the loss of the target in the camera's field of view and the possibility of encountering obstacles. To overcome all the aforementioned issues, we propose a new trajectory planning and tracking method based on the concept of differential flatness. The key idea of this approach is to separate altitude control from the control of the other variables, allowing the quadrotor to initially fly at a high altitude to increase the camera's field of view. Once the target is detected, a path is planned in the image coordinate system between the detected image and the desired image at the same altitude. This trajectory is then converted into a trajectory in Cartesian space, independent of altitude. This planning, conducted offline, makes the quadrotor move in a way that gets it to its goal. It does this by setting dynamics like a smooth start, acceleration in the middle of the path, and a smooth arrival, which control the dynamics in the Cartesian plane. To reinforce this approach, we introduce a visual servoing method, demonstrating that a single point of the target to reach, representing, for example, the object's center of gravity, is sufficient. A trajectory generation block then converts the chosen trajectory in the image plane into a trajectory expressed in Cartesian space, independent of the altitude of the quadrotor. Furthermore, to get around problems with underactuated control and strong coupling of the quadrotor, we suggest a flatness-based control that would make the system controllable and make sure the generated trajectory converges asymptotically.

We include numerical simulations in order to validate the efficacy of this novel control approach. We also tested the proposed control laws under real-world conditions. We used the DJI Phantom 1 quadrotor as a testbed for this. Our team developed a specialized electronic card with the purpose of transferring calculated commands to the DJI Phantom 1 control joystick through the use of a Bluetooth connection. The experiments demonstrate the effectiveness of this method by ensuring accurate trajectory tracking in the image plane and precise target tracking. These advancements promise to significantly enhance the capabilities of camera-equipped quadrotors to reach fixed targets in complex environments, paving the way for new applications and developments in the field of aerial robotics.

This paper is organized as follows: Section 2 describes the quadrotor dynamic model. Section 3 explains the control strategy. The implementation of the control strategy takes place in two stages. Establishing the quadrotor's kinematic actions to track the trajectory in the image plane is the first step. This phase is offline. In the second phase, a differential flatness control strategy guarantees the asymptotic convergence of the Cartesian trajectory while maintaining a certain robustness. Section 4 shows the simulation results to verify the methodology's efficiency. The experimental tests performed using the DJI Phantom 1 are presented in Section 5.

2. Quadrotor Dynamic Model

The quadrotor dynamic model, which is widely employed [23–25], can be expressed mathematically as follows:

$$\begin{cases} \ddot{x} = u_1(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) - \frac{K_1\dot{x}}{m} \\ \ddot{y} = u_1(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) - \frac{K_2\dot{y}}{m} \\ \ddot{z} = u_1(\cos\theta\cos\phi) - g - \frac{K_3\dot{z}}{m} \\ \ddot{\theta} = u_2 - \frac{lK_4\dot{\theta}}{I_1} \\ \ddot{\phi} = u_3 - \frac{lK_5\dot{\phi}}{I_2} \\ \ddot{\psi} = u_4 - \frac{K_6\dot{\psi}}{I_3} \end{cases}. \tag{1}$$

Numerous experimental tests have supported this model. In Equation (1), (x, y, z) are the coordinates of the quadrotor; (θ, ϕ, ψ) represent the Euler angles for pitch, roll, and yaw; g is the gravitational acceleration; l represents the distance from the quadrotor's center of gravity to its rotors; m represents the whole mass of the quadrotor; (I_1, I_2, I_3) are the moments of inertia along the directions x, y, and z; $(K_1, K_2, K_3, K_4, K_5, K_6)$ are the drag coefficients (in this work, we make the assumption that the drag force is negligible, as it becomes insignificant at low velocities); and (u_1, u_2, u_3, u_4) are the command inputs defined by [24] as follows:

$$\begin{cases} u_1 = \frac{(T_1+T_2+T_3+T_4)}{m} \\ u_2 = \frac{l(-T_1-T_2+T_3+T_4)}{I_1} \\ u_3 = \frac{l(-T_1+T_2+T_3-T_4)}{I_2} \\ u_4 = \frac{C(T_1-T_2+T_3-T_4)}{I_3} \end{cases}. \tag{2}$$

In Equation (2), (T_1, T_2, T_3, T_4) represent the thrusts produced by the four rotors. These variables are the effective control inputs of the system. C denotes the force–moment scaling factor; u_1 represents the total thrust on the body according to the Z-axis; u_2 and u_3 are the pitch and roll inputs; and u_4 is the yaw input.

3. Control Strategy

The strategy of control allowing the quadrotor to follow a specific trajectory in the image plane is explained as follows: The quadrotor flies at a predefined altitude z_d, usually at a high altitude for a better view from its downward-facing camera. When the quadrotor locates the target, we can select any path in the image coordinate system that connects the first image the camera captures to the desired final image, both at the same altitude z_d. This trajectory takes into account the relative position of the target in relation to the quadrotor as well as the velocity, thus making it possible to specify a desired dynamic. For example, we can plan for a smooth start, acceleration in the middle of the path, and a smooth arrival. This also means that we can impose dynamics on the quadrotor itself in the Cartesian plane to reach the target.

In this study, we demonstrate that a single point of the target to be reached is sufficient to achieve visual control. A trajectory generation module converts the path chosen in the image plane into another expressed in Cartesian space. The trajectory thus generated will be independent of the altitude of the quadrotor. Thanks to the notion of differential flatness, we only need to know the positions (x, y, z), and the yaw orientation ψ to describe and control all the remaining variables of the system. Given that we previously imposed the altitude z independently of the other variables, the trajectory generated in Cartesian space provides the three other variables (x, y, ψ). The control strategy takes place in two stages. Firstly, in the initial stage, the focus is on developing the required movements that the quadrotor must execute to track the trajectory within the image plane. This phase occurs offline, meaning it is conducted prior to the actual execution of the trajectory. During this stage, intricate planning takes place, mapping out the sequence of movements necessary for the quadrotor to precisely follow the desired trajectory when projected onto the image plane. This planning ensures that when the quadrotor is in motion, it can adhere to the trajectory outlined in the image plane. The second stage consists of ensuring asymptotic convergence of the trajectory thus established in Cartesian space with a given level of robustness using a control based on differential flatness. Differential flatness, as a concept, becomes instrumental in orchestrating the quadrotor's movements in a way that not only follows the planned trajectory in the image plane but also guarantees the asymptotic convergence in three-dimensional Cartesian space. The term "asymptotic convergence" implies that the quadrotor progressively approaches and settles onto the desired trajectory over time.

3.1. Path Generation in Cartesian Space (Offline)

It should be emphasized that all the simplification hypotheses addressed in this paragraph are limited to the problem of generating the trajectory in Cartesian space from the desired trajectory chosen in the image plane. Subsequently, the dynamics of all system variables will be taken into account in the control problem. The principle of this step is illustrated in Figure 1. Once the quadrotor has been stabilized at a specific altitude, meaning the roll and pitch angles are very low, we proceed to acquire the image containing the target. Then, we choose an arbitrary trajectory that connects the initial image captured by the camera to the desired image to reach the same altitude. Since both images are captured at the same altitude, the trajectory connecting these two images can be independent of z altitude. In other words, we can consider an in-plane trajectory, as shown in Figure 2. Since the generated trajectory in Cartesian space is independent of the altitude of the quadrotor, we can choose a lower altitude for the quadrotor. This decision aims to save energy and reduce sources of disturbance that could affect the system. Therefore, we can select a trajectory in the image plane that bypasses obstacles, as illustrated in Figure 1.

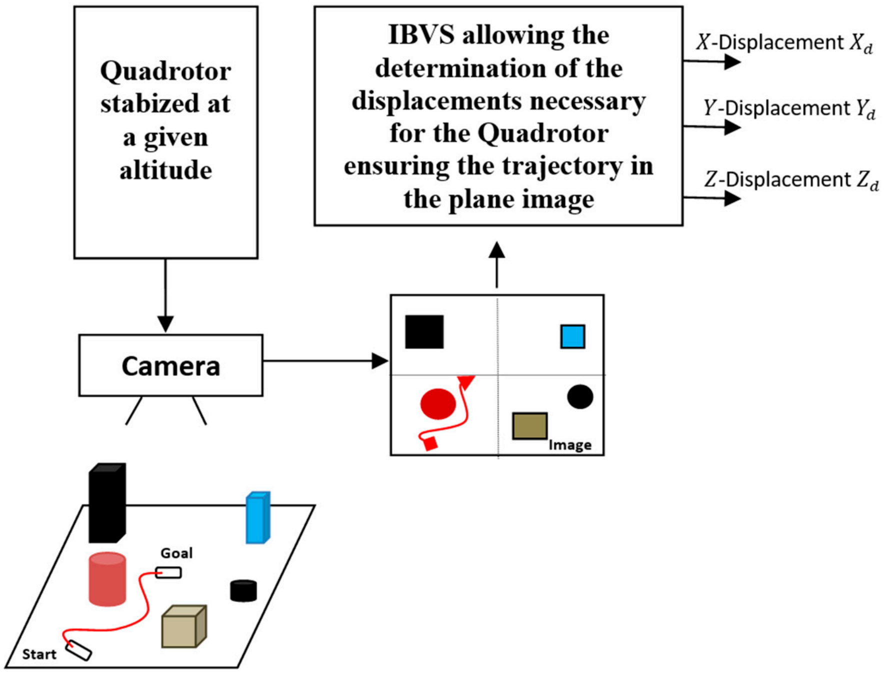

Quadrotor stabized at a given altitude
IBVS allowing the determination of the displacements necessary for the Quadrotor ensuring the trajectory in the plane image
X-Displacement X_d
Y-Displacement Y_d
Z-Displacement Z_d
Camera
Image
Goal
Start

Figure 1. Path generation in Cartesian space: the principle.

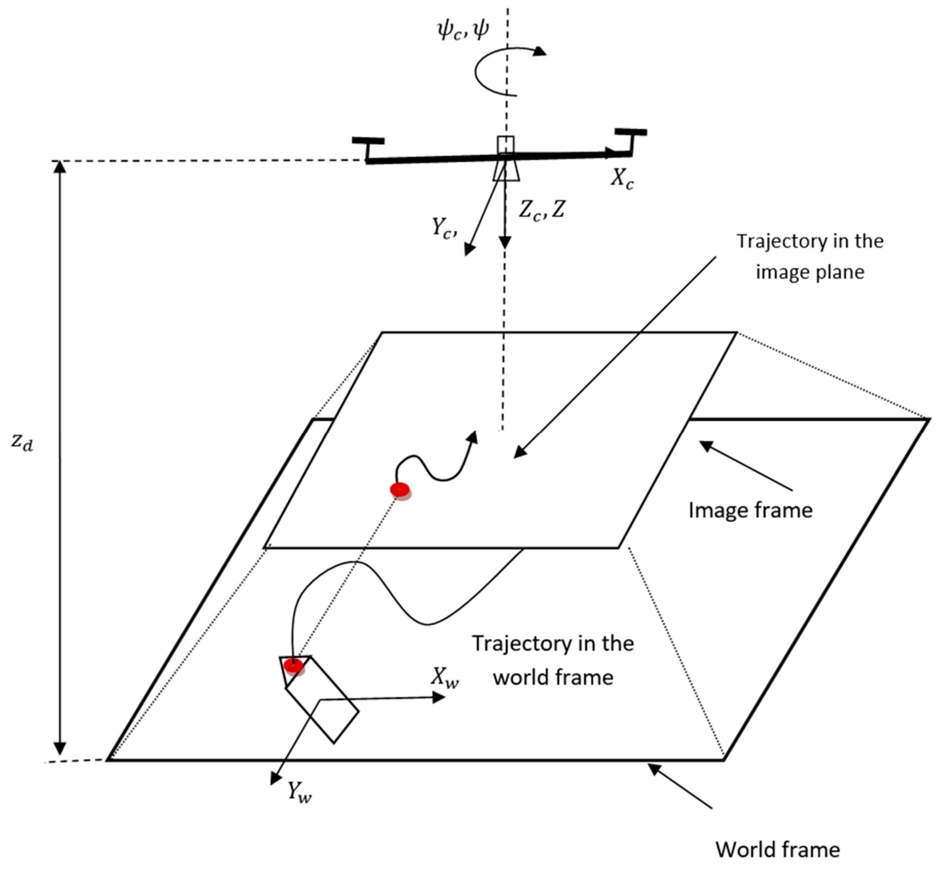


Figure 2. Trajectory in the world frame vs. trajectory in the image plane.

Due to the quadrotor's virtual displacement being limited to a certain plane and the dynamics of the desired trajectory, we can look at the onboard camera's behavior on the quadrotor in a way that is similar to that of a differential-based mobile robot. This robot operates within a plane that is parallel to the (X_w, Y_w) plane, and it maintains a constant distance of z_d from it. Moreover, the mobile robot executes a rotational movement around the Z_W-axis with an angle ψ (as shown in Figure 2), adhering to the following dynamics:

$$\begin{cases} \dot{x}_r = v_r \cos(\psi) \\ \dot{y}_r = v_r \sin(\psi) \\ \dot{\psi} = \omega_r \end{cases}, \tag{3}$$

where $\dot{x}_r$ and $\dot{y}_r$ denote the translational speeds along the X_w and Y_w axes of the robot, respectively, and v_r and ω_r represent the robot's linear and angular velocities, respectively.

3.1.1. Characteristics of the Descriptor

Based on the dynamics described in Equation (3), recognition of the translation velocities $(\dot{x}_r, \dot{y}_r)$ along the X_w and Y_w axes allows deduction of the orientation ψ along the Z_W-axis. Indeed:

$$\psi = \text{atan2}\left(\frac{\dot{y}_r}{\dot{x}_r}\right). \tag{4}$$

Since we know that $z_c = z = z_d = constant$, we can deduce the translation velocities $(\dot{x}_r, \dot{y}_r)$ by exploring the location of a single point P on the target. P can be arbitrarily chosen from the target, but it should not coincide with the camera's projection center. Indeed, the points belonging to the camera's projection center remain unchanged when subjected to rotations around the Z_W-axis. To simplify the detection process, we can choose the centroid of the target or the centroid of a part of the target, as depicted in Figure 3.

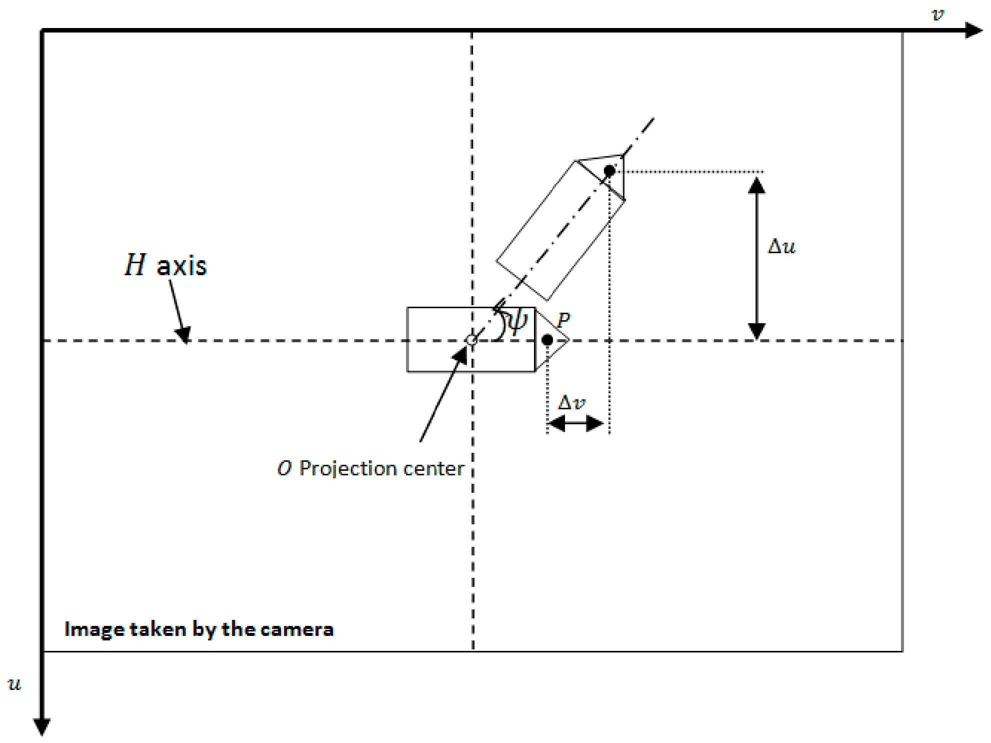


Figure 3. Characteristics of the descriptor.

3.1.2. Image-Based Visual Servoing

Consider P a point in the 3D Cartesian coordinate system, where its coordinates are represented by $X = (X_c, Y_c, Z_c = z_d = constant)$. Given the coordinates (x_m, y_m) expressed in millimeters, point p represents the projection of P onto the image coordinate system. In this work, the visual data considered are denoted by $S = (x_m, y_m)$. The following equations define the mathematical expressions for these coordinates:

$$\begin{cases} x_m = \frac{X_c}{Z_c} = \frac{(u-c_u)}{f.\alpha_u} \\ y_m = \frac{Y_c}{Z_c} = \frac{(v-c_v)}{f.\alpha_v} \end{cases}. \tag{5}$$

The pair (u,v) denotes the pixel-based coordinates of the image point p. The parameters $a = (c_u, c_v, f, \alpha_u, \alpha_v)$ refer to the camera's intrinsic attributes, where (c_u, c_v) represents the coordinates of the image's principal point, f denotes the focal length, and (α_u, α_v) represent the vertical and horizontal scale factors expressed in pixels per millimeter. By applying the operator of time derivative to the projection Equation (5), we get the following:

$$\dot{S} = l_s.\, V. \tag{6}$$

The kinematic tensor of the camera, denoted as V, is composed of the translational velocities v_c and the rotational velocities ω_c. L_s is the matrix that represents the interaction between variables, commonly known as the image Jacobian. It is derived from the following:

$$L_s = \begin{bmatrix} -\frac{1}{z_d} & 0 & \frac{x_m}{z_d} & x_m y_m & -\left(1 + {x_m}^2\right) & y_m \\ 0 & -\frac{1}{z_d} & \frac{y_m}{z_d} & 1 + {y_m}^2 & -x_m y_m & -x_m \end{bmatrix}. \tag{7}$$

Since the robot's movement is in the two-dimensional plane, using Equation (4), we can compute the translation velocity along the Y_c-axis given the translation velocity along the X_c-axis and the rotation velocity along the Z_c-axis. The interaction matrix in Equation (7) becomes the following:

$$L_s = \begin{bmatrix} -\frac{1}{z_d} & y_m \\ 0 & -x_m \end{bmatrix}. \tag{8}$$

Equation (6) can be reformulated in the subsequent format:

$$\begin{pmatrix} \dot{x}_m \\ \dot{y}_m \end{pmatrix} = \begin{pmatrix} -\frac{1}{z_d} & y_m \\ 0 & -x_m \end{pmatrix} \begin{pmatrix} v_r \\ \omega_r \end{pmatrix}. \tag{9}$$

3.1.3. Generation of the Trajectory in Cartesian Space

Let $(x^*(t),\ y^*(t))$ be the desired trajectory chosen in the image coordinate system. We generate a robot-executed trajectory in the image plane using a homographic transformation, ensuring asymptotic convergence to the desired trajectory. Using Equation (9), the two control inputs of the mobile robot are given by the following:

$$\begin{pmatrix} v_r \\ \omega_r \end{pmatrix} = \begin{pmatrix} -\frac{1}{z_d} & y_m \\ 0 & -x_m \end{pmatrix}^{-1} \begin{pmatrix} \dot{x}_m \\ \dot{y}_m \end{pmatrix}. \tag{10}$$

Since we assumed that point p does not coincide with the projection center, and by explicitly expressing the image Jacobian inverse (for $x_m \neq 0$), we obtain the following:

$$\begin{pmatrix} v_r \\ \omega_r \end{pmatrix} = \begin{pmatrix} -z_d & -\frac{z_d y_m}{x_m} \\ 0 & -\frac{1}{x_m} \end{pmatrix} \begin{pmatrix} \dot{x}_m \\ \dot{y}_m \end{pmatrix}. \tag{11}$$

In this situation, a reversible correlation is established between the outputs and inputs. Subsequently, we utilize a precise linearization technique as proposed in [26]. The linearized system that is obtained can be represented as a system that exhibits an integration as follows:

$$\begin{cases} \dot{x}_m = \vartheta_x \\ \dot{y}_m = \vartheta_y \end{cases}, \tag{12}$$

where ϑ_x and ϑ_y represent the two auxiliary control inputs that need to be determined in order to achieve asymptotic tracking of the intended path. The control law governing the behavior of the mobile robot is expressed as follows:

$$\begin{pmatrix} v_r \\ \omega_r \end{pmatrix} = \begin{pmatrix} -z_d & -\frac{z_d y_m}{x_m} \\ 0 & -\frac{1}{x_m} \end{pmatrix} \begin{pmatrix} \vartheta_x \\ \vartheta_y \end{pmatrix}, \tag{13}$$

with the following:

$$\begin{cases} \vartheta_x = \dot{x}^* + k_1(x^* - x_m) \\ \vartheta_y = \dot{y}^* + k_2(y^* - y_m) \end{cases}, \tag{14}$$

where k_1 and k_2 must have values guaranteeing asymptotically stable error dynamics. It is sufficient to take $k_1 > 0$ and $k_2 > 0$, ensuring asymptotic tracking of the desired trajectory $(x^*,\ y^*)$. Once we have synthesized the two controls (v_r, ω_r) that ensure asymptotic tracking of the specified path in the image coordinate system we can deduce, using Equation (3), the necessary displacements $(x_d,\ y_d,\ \psi_d)$ that the quadrotor must execute to ensure the pursuit of this trajectory.

3.2. Tracking of the Trajectory Generated in Cartesian Space

The proposed method for ensuring the tracking of the generated trajectory in Cartesian space is illustrated in Figure 4. This method consists of two distinct control loops. The first loop deals with altitude control, while the second loop manages the displacements and orientation in the plane using a flatness-based control.

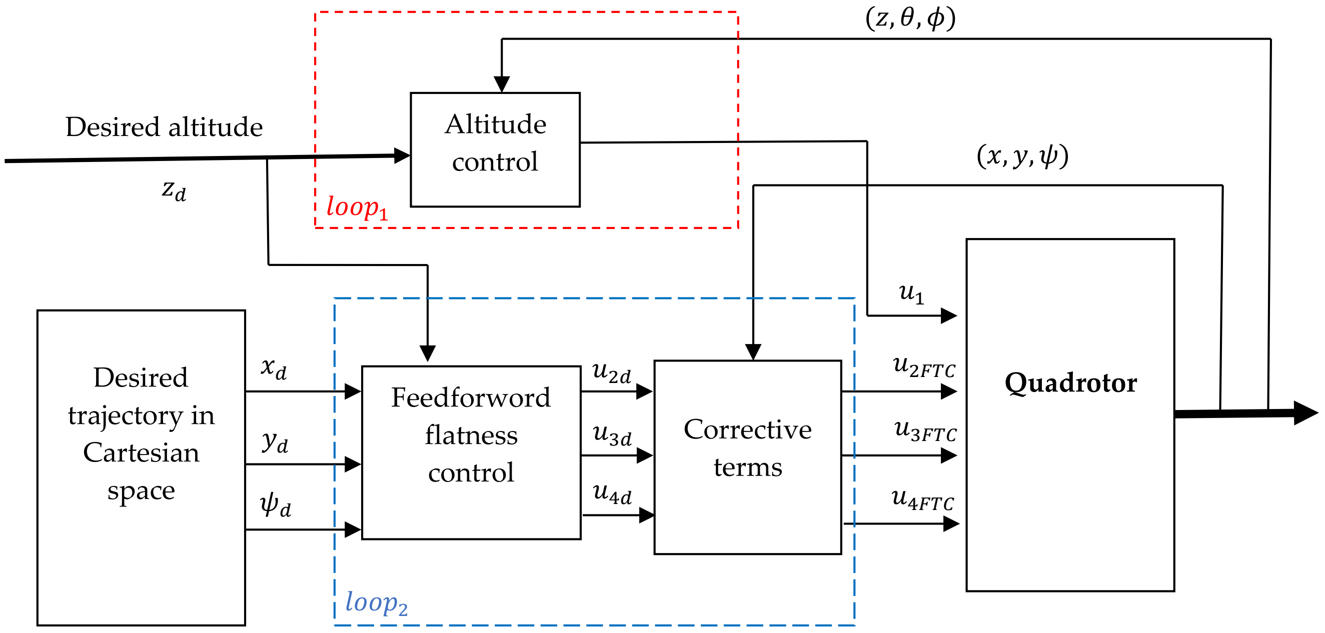


Figure 4. Tracking of the trajectory generated in Cartesian space (online).

3.2.1. Loop 1: Control of the Altitude

As stated in Section 2, the control input u_1 governs the displacement along Z. By employing the input–output feedback linearization method to the third equation of the system of Equation (1), the linearizing control will be given by the following:

$$u_1 = \frac{(Nu_z + g)}{\cos\theta\cos\varphi}, \text{ with } \cos\theta\cos\varphi \neq 0. \tag{15}$$

Note that Nu_z is the linearized system's new input:

$$\ddot{z} = Nu_z. \tag{16}$$

To ensure that the altitude $z(t)$ tracks the desired altitude $z_d(t)$, it is sufficient to take the new input as follows:

$$Nu_z = \ddot{z}_d + k_{11}(\dot{z}_d - \dot{z}) + k_{12}(z_d - z). \tag{17}$$

The coefficients k_{11} and k_{12} are chosen such that the polynomial $s^2 + k_{11}s + k_{11}$ is a Hurwitz-type polynomial.

3.2.2. Loop 2: Tracking Based on Differential Flatness

In this paragraph, we provide a flatness-based control approach aimed at achieving and guaranteeing the asymptotic convergence of the trajectory generated in Section 3.1.3. The study of flat systems is a complex field within the disciplines of differential geometry and algebra. The concept of differential flatness is introduced in [22]. Let us consider a nonlinear system that Equation (18) describes as follows:

$$\dot{x} = f(x, u), \ x \in \mathfrak{R}^n \text{ and } u \in \mathfrak{R}^m. \tag{18}$$

The system in (18) is considered differentially flat if a vector $F \in \mathfrak{R}^m$ that satisfies the given condition exists as follows:

$$F = \xi\left(x,\, u,\, \dot{u},\, \ldots,\, u^{(r-1)}\right), \tag{19}$$

where elements are differentially independent. Moreover, there should be the existence of $\eta(.)$ and $\Gamma(.)$ such that:

$$x = \eta\left(F,\, \dot{F},\, \ddot{F},\, \ldots,\, F^{(\alpha-1)}\right), \tag{20}$$

$$u = \Gamma\left(F, \dot{F},\, \ddot{F},\, \ldots,\, F^{(\alpha)}\right), \tag{21}$$

In the given context, α and r represent finite multi-indices, while ξ, η, and Γ denote vectors consisting of smooth functions. F is commonly known as the flat output of the system. A flat system is characterized by the ability to express the state and control variables in terms of the flat output and its derivatives. The open-loop flatness control, referred to as the Brunovosky control, is denoted by Equation (21) and is recognized for its capability to achieve an exact linearization of the system. In the case of a differentially flat system, the desired trajectory F_d can be determined. This allows for the definition of the desired state x_d and the desired open-loop control u_d in the following manner.

$$x_d = \eta\left(F_d,\, \dot{F}_d,\, \ddot{F}_d,\, \ldots,\, F_d^{(\alpha-1)}\right), \tag{22}$$

$$u_d = \Gamma\left(F_d,\, \dot{F}_d,\, \ddot{F}_d,\, \ldots,\, F_d^{(\alpha)}\right). \tag{23}$$

In the case that the system demonstrates inherent stability, it will exhibit satisfactory behavior and adhere to the planned trajectory. In order to enhance the convergence speed, unstable systems require the addition of a closed-loop correction term to the existing open-loop control. This correction term is crucial for ensuring accurate trajectory tracking. Thus, we propose a closed-loop flatness control system. The notation FTC, which stands for Flatness-Based Tracking Control, is used to represent this loop. The control system consists of two components: the open-loop control, represented by Equation (21) and a loop term ϑ. The loop term ϑ is a linear control that is designed to stabilize the linearized system. The FTC is provided in the following manner:

$$u_{FTC} = \Gamma\left(F_d,\, \dot{F}_d,\, \ddot{F}_d,\, \ldots,\, \vartheta\right), \tag{24}$$

where $\vartheta(t)$ indicates the newly introduced command. When the partial derivative of Γ with respect to $F^{(\alpha)}$ is locally invertible, it results in the following decoupled system:

$$F^{(\alpha)} = \vartheta, \tag{25}$$

with the following:

$$\vartheta = F_d^{(\alpha)} + \sum\nolimits_{i=0}^{\alpha-1} k_i\left(F_d^{(i)} - F^{(i)}\right). \tag{26}$$

Let $K(s) = s^\beta + \sum_{i=0}^{\beta-1} k_i s^i$. $K(s)$ is a diagonal matrix. The components of $K(s)$ are polynomials. The roots of these polynomials have a strictly negative real part. The proposed $K(s)$ enables the achievement of the asymptotic trajectory tracking $\lim\limits_{t\to\infty}(F_d - F) = 0$.

As illustrated in Figure 4, the control, based on differential flatness, ensures the necessary displacements of the UAV to follow the desired trajectory in the image plane. This command employs an open-loop method to make the system linear and a closed-loop correction term to make sure that the desired path converges asymptotically, even when there are disturbances. Replacing the control u_1 expressed by Equation (15) in the model describing the quadrotor's dynamics (Equation (1)), we obtain the following:

$$\begin{cases} \ddot{x} = (Nu_z + g)\tan\theta\cos\psi + (Nu_z + g)\frac{\tan(\phi)}{\cos\theta}\sin\psi \\ \ddot{y} = (Nu_z + g)\tan\theta\sin\psi - (Nu_z + g)\frac{\tan(\phi)}{\cos\theta}\cos\psi \\ \ddot{z} = Nu_z = \ddot{z}_d + k_{11}(\dot{z}_d - \dot{z}) + k_{12}(z_d - z) \\ \ddot{\theta} = u_2 \\ \ddot{\phi} = u_3 \\ \ddot{\psi} = u_4 \end{cases}. \tag{27}$$

We can demonstrate that this system is flat and has the following flat outputs: $F_1 = z$; $F_2 = x$; $F_3 = y$; $F_4 = \psi$. Via the first and second equations within system (27), we can express the variables θ and ϕ in terms of the flat outputs:

$$\begin{cases} \theta = \arctan\left(\frac{\cos(\psi)\ddot{x}+\sin(\psi)\ddot{y}}{\ddot{z}+g}\right) \\ \phi = \arcsin\left(\frac{\sin(\psi)\ddot{x}-\cos(\psi)\ddot{y}}{\sqrt{\ddot{x}^2+\ddot{y}^2+\left(\ddot{z}+g\right)^2}}\right) \end{cases}. \tag{28}$$

The control variables can be represented in relation to the flat outputs and their respective derivatives:

$$\begin{cases} u_2 = \ddot{\theta} = \frac{d^2}{dt}\left(\arctan\left(\frac{\cos(\psi)\ddot{x}+\sin(\psi)\ddot{y}}{\ddot{z}+g}\right)\right) \\ u_3 = \ddot{\phi} = \frac{d^2}{dt}\left(\arcsin\left(\frac{\sin(\psi)\ddot{x}-\cos(\psi)\ddot{y}}{\sqrt{\ddot{x}^2+\ddot{y}^2+\left(\ddot{z}+g\right)^2}}\right)\right) \\ u_4 = \ddot{\psi} \end{cases}. \tag{29}$$

We have just expressed all the variables of the system in terms of the dynamics of $(z,\ x,\ y,\ \psi)$. Subsequently, system (27) is flat and has the following flat outputs: $F_1 = z$; $F_2 = x$; $F_3 = y$; and $F_4 = \psi$. To execute the desired trajectory $(x_d,\ y_d,\ \psi_d)$, created using Equation (29), it is possible to infer the open-loop control that will achieve this desired trajectory:

$$\begin{cases} u_{2d} = \frac{d^2}{dt}\left(\arctan\left(\frac{\cos(\psi_d)\ddot{x}_d+\sin(\psi_d)\ddot{y}_d}{\ddot{z}_d+g}\right)\right) \\ u_{3d} = \frac{d^2}{dt}\left(\arcsin\left(\frac{\sin(\psi_d)\ddot{x}_d-\cos(\psi_d)\ddot{y}_d}{\sqrt{\ddot{x}_d^2+\ddot{y}_d^2+\left(\ddot{z}_d+g\right)^2}}\right)\right) \\ u_{4d} = \ddot{\psi}_d \end{cases}. \tag{30}$$

At this stage, flatness has been utilized for the computing controls that match the open-loop trajectories of the system. Once the system reaches a state of stability, it will respond accordingly and adhere to the intended path. However, for non-stable systems or when one aims to expedite convergence, it is imperative to augment this open-loop command with a small closed-loop correction term to guarantee precise trajectory tracking. To generate these correction terms, we will make hypotheses. It is important to emphasize that these hypotheses only apply to the derivation of the correction terms and are considered solely in the vicinity of the desired trajectory. Once the quadrotor attains its desired trajectory, it is reasonable to hypothesize that the angles θ, ϕ, and ψ will decrease in magnitude. The expressions for the second derivative of θ and ϕ are provided as follows:

$$\begin{cases} \ddot{\theta} = \frac{F_2^{(4)}}{\ddot{F}_1+g} - 2\frac{F_2^{(3)}F_1^{(3)}}{\left(\ddot{F}_1+g\right)^2} + 2\frac{\ddot{F}_1\left(F_1^{(3)}\right)^2}{\left(\ddot{F}_1+g\right)^3} - \frac{\ddot{F}_2F_1^{(4)}}{\left(\ddot{F}_1+g\right)^3} \\ \ddot{\phi} = \frac{F_3^{(4)}}{\ddot{F}_1+g} + 2\frac{F_3^{(3)}F_1^{(3)}}{\left(\ddot{F}_1+g\right)^2} - 2\frac{\ddot{F}_3\left(F_1^{(3)}\right)^2}{\left(\ddot{F}_1+g\right)^3} + \frac{\ddot{F}_3F_1^{(4)}}{\left(\ddot{F}_1+g\right)^2} \end{cases}. \tag{31}$$

By employing the theorem provided by [27], which ignores all terms in the polynomial equation above the fourth degree, Equation (31) becomes the following:

$$\begin{cases} \ddot{\theta} = \frac{F_2^{(4)}}{\ddot{F}_1+g} \\ \ddot{\phi} = \frac{F_3^{(4)}}{\ddot{F}_1+g} \end{cases}. \tag{32}$$

Assuming that the quadrotor attains its desired altitude $(z - z_d = 0)$, the expressions for the commands can be written as follows:

$$\begin{cases} u_2 = \ddot{\theta} = \frac{F_2^{(4)}}{g} \\ u_3 = \ddot{\phi} = \frac{F_3^{(4)}}{g} \end{cases}. \tag{33}$$

Finally, here are the expressions for the closed-loop control laws that confirm asymptotic convergence (FTC: Flatness Tracking Control) toward the desired path, even when there are disturbances:

$$\begin{cases} u_{2FTC} = u_{2d} + k_{21}e_2^{(3)} + k_{22}e_2^{(2)} + k_{23}\dot{e}_2 + k_{24}e_2 \\ u_{3FTC} = u_{3d} + k_{31}e_3^{(3)} + k_{32}e_3^{(2)} + k_{33}\dot{e}_3 + k_{34}e_3, \\ \qquad u_{4FTC} = u_{4d} + k_{41}\dot{e}_4 + k_{42}e_4 \end{cases} \tag{34}$$

where $e_i = F_{id} - F_i$; (i = 2, 3, 4), and the k_{ij} terms are designated using the pole placement technique.

4. Simulation Results

As mentioned in Section 3, our method consists of two steps: a first step for generating the trajectory in Cartesian space, which is performed offline, and a second step involving the execution of the generated trajectory. The RVCTOOLS library was used for conducting the simulations.

4.1. Step 1: Trajectory Generation (Offline)

In order to evaluate the efficiency of the proposed tracking approach, we implemented the following procedure: The quadrotor initially reached a predefined altitude. At time $t = 10$ s (the time required for the quadrotor to stabilize at the altitude $z_d = 15$ m), we detected point P on the target. In our simulation, we considered the midpoint on one side of an object defined by four points. The next step involved choosing an arbitrary path in the image coordinate system that linked the starting and ending positions of point P on the target, both captured at the same altitude. It is crucial to consider that we can trace an arbitrary trajectory on the screen of an interface and extract an analytical expression for it. Given that we have the capability to define the dynamics of this trajectory (in terms of position and velocity), we opted for a polynomial-type trajectory, where $P_I = \left(x_i^*,\ y_i^*\right)$ was the initial position of point P on the object in the image coordinate system at time t_i and $P_f = \left(x_f^*,\ y_f^*\right)$ was its final position at time t_f. Let us consider the task of finding a path linking these two points and passing through a peak (maximum). To illustrate this, let us consider the example of a point with coordinates $\left(\frac{x_f^*+x_i^*}{2},\ 2y_f^* - y_i^*\right)$, which represents the maximum of the curve between y_i^* and y_f^*. We suggest the following dynamics: a gradual start, a sudden increase in speed during the path, and ultimately a smooth finish. The intended path $y^*(x^*)$ must meet the following constraints:

$$\begin{cases} y^*\left(x_i^*\right) = y_i^* \\ y^*\left(x_f^*\right) = y_f^* \\ y^*\left(\frac{x_f^*+x_i^*}{2}\right) = 2y_f^* - y_i^* \\ \frac{dy^*}{dx^*}\left(\frac{x_f^*+x_i^*}{2}\right) = 0 \\ \frac{d^2y^*}{d^2x^*}\left(\frac{x_f^*+x_i^*}{2}\right) < 0 \end{cases}. \quad (35)$$

We can use, for example, the polynomial equation in x^*, which fulfills the aforementioned constraints.

$$y^*(x^*) = y_i^* + \left(y_f^* - y_i^*\right)\left(\frac{x^* - x_i^*}{x_f^* - x_i^*}\right)\left(9 - 12\left(\frac{x^* - x_i^*}{x_f^* - x_i^*}\right) + 4\left(\frac{x^* - x_i^*}{x_f^* - x_i^*}\right)^2\right). \quad (36)$$

It remains to construct the evolution of $x^*(t)$. This must satisfy the specified limit conditions:

$$x^*(t_i) = x_i^*, \dot{x}^*(t_i) = 0, \cdots, x^{*(5)}(t_i) = 0, \quad (37)$$

$$x^*\left(t_f\right) = x_f^*, \dot{x}^*\left(t_f\right) = 0, \cdots, x^{*(5)}\left(t_f\right) = 0. \quad (38)$$

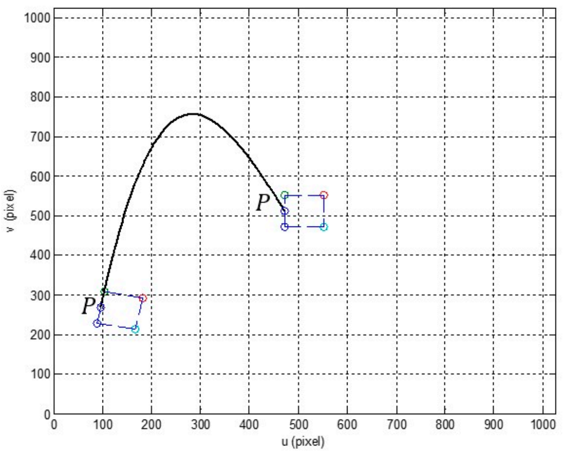
This translates to the 11th-degree polynomial as follows:

$$x^*(t) = x_i^* + \left(x_f^* - x_i^*\right)\sigma^6(t)\left(462 - 1980\sigma(t) + 3465\sigma^2(t) - 3080\sigma^3(t) + 1386\sigma^4(t) - 252\sigma^5(t)(t)\right), \quad (39)$$

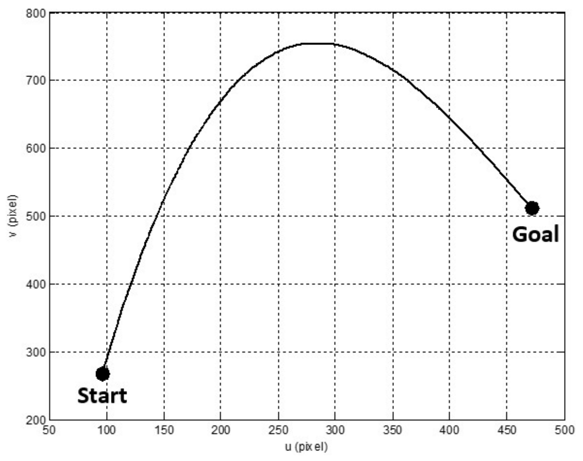
where:

$$\sigma(t) = \frac{t - t_i}{t_f - t_i}. \quad (40)$$

The dynamics of the desired trajectory are depicted in Figure 5. Figure 5a,b illustrate the desired position trajectory, connecting the two boundary points in the image plane and passing through a maximum. Figure 5c,d show the evolution of the trajectory along the axes (u and v), while Figure 5e,f present the velocity dynamics of these trajectories, reflecting the desired dynamics with a gradual start, acceleration in the middle, and smooth convergence. The image-based visual control is implemented using Equations (13) and (14). The coefficients of Equation (14) are defined as follows: $k_1 = 10$; $k_2 = 10$.

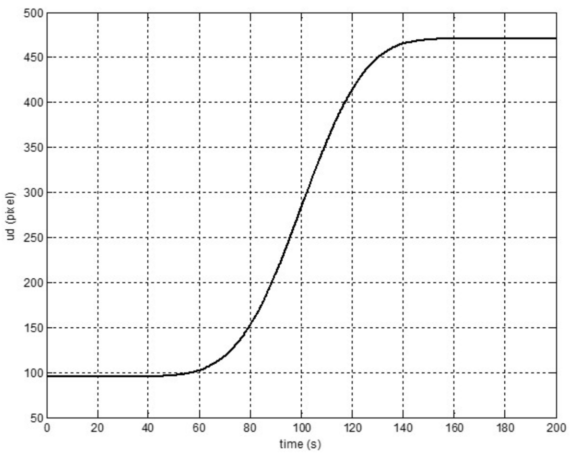


(**a**)

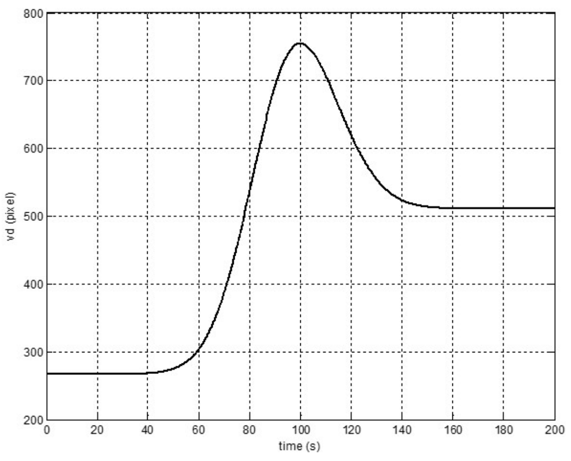


(**b**)

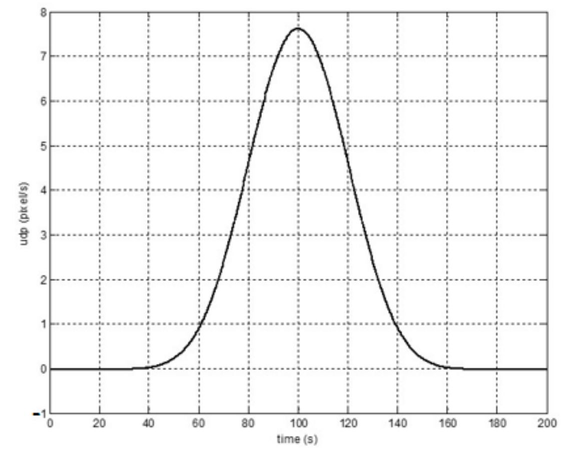
Figure 5. *Cont.*



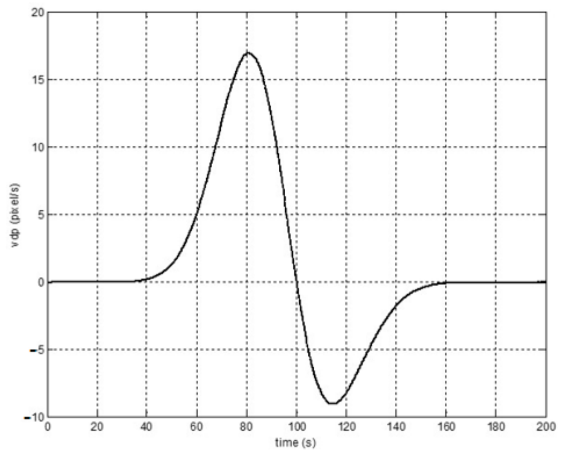
(**c**)



(**d**)



(**e**)



(**f**)

Figure 5. The dynamics of the desired trajectory: (**a**,**b**) the desired position trajectory of a point P; (**c**) the evolution of the trajectory along the u-axis; (**d**) the evolution of the trajectory along the v-axis; (**e**) the velocity dynamics of the trajectory along the u-axis in pixel/s; and (**f**) the velocity dynamics of the trajectory along the v-axis pixel/s.

The simulation results are presented in Figure 6. Figure 6a,b show the evolution of the desired trajectory and the trajectory obtained through visual control. It is evident that the visual control algorithm operates effectively in generating a trajectory faithful to the desired path. Figure 6c–e represent the necessary displacements generated by the visual control algorithm. These displacements indicate the movements that the quadrotor must execute to track the desired trajectory chosen in the image plane.

4.2. Step 2: Generated Trajectory Performed by the Quadrotor (Online Tracking)

In the previous step, we generated desired trajectories for the variables x, y, and Ψ. Given that the proposed method separates altitude control (the displacement along the z-axis) from the other variables, we have the ability to independently choose the desired trajectory for the altitude. To evaluate the effectiveness of our control strategy, although we considered a constant altitude in the trajectory generation problem, we introduced a variable altitude as a source of disturbance, as follows: The quadrotor stabilizes at an altitude $z_d = 15$ m, then undergoes a gradual takeoff, acceleration in the middle, and smooth convergence towards the altitude $z_d = 1$ m. The parameters of Equation (17) ensuring the pursuit of this desired trajectory are chosen as follows: $k_{11} = 10$ and $k_{12} = 25$. Taking into account the physical characteristics of the quadrotor, the control input for altitude must be bounded by $|u_1| < 15$. Figure 7a,b illustrate the evolution of the desired altitude and the altitude achieved by the quadrotor. It is clear that the quadrotor stabilizes around the altitude z = 15 m in less than 4 s and follows the desired trajectory. The evolution of the control law ensuring the desired altitude is presented in Figure 7c,d. It is evident that this is a continuous, smooth, and physically achievable evolution.

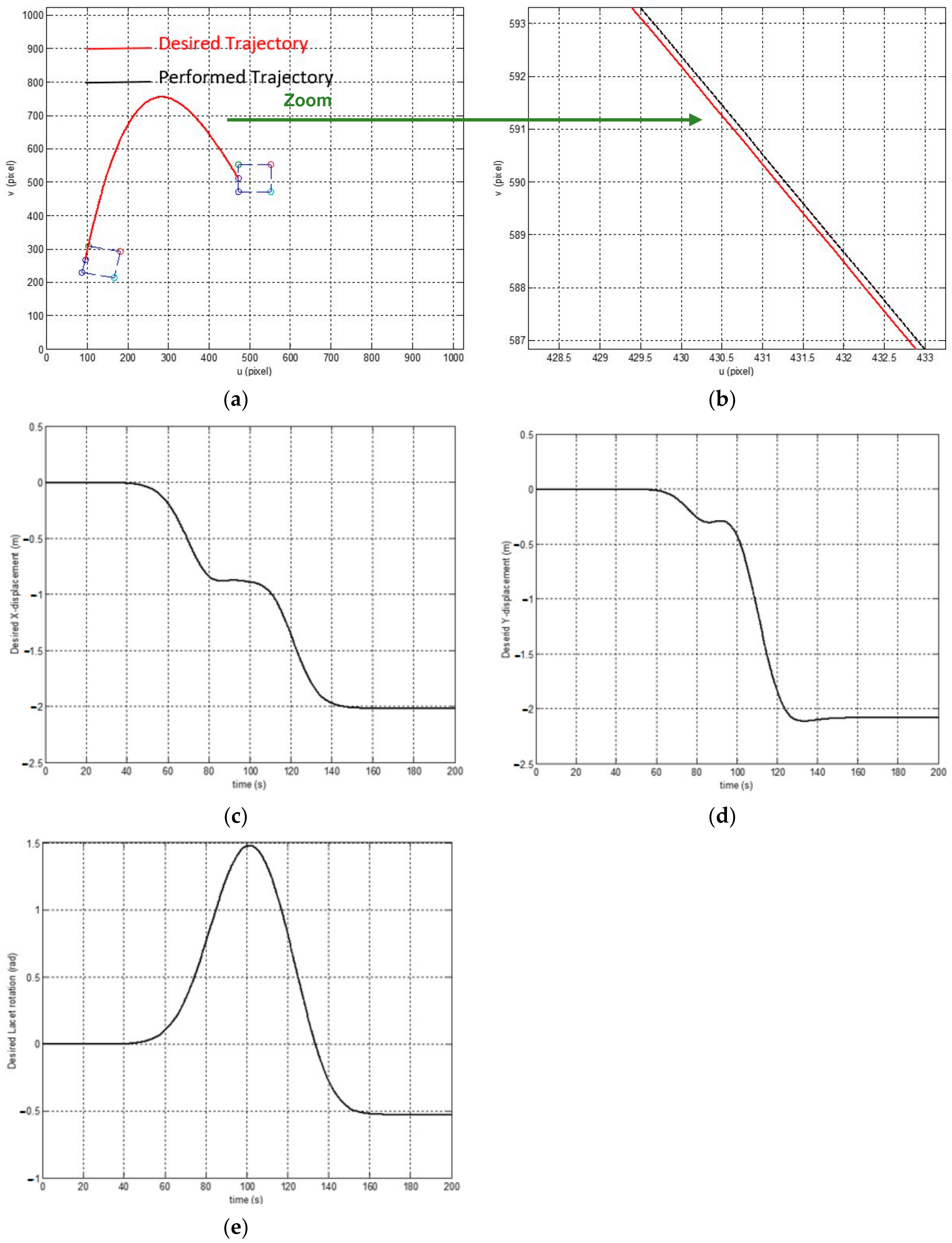


Figure 6. Trajectories generation and tracking in the image plane: (**a**,**b**) the desired trajectory and the performed trajectory in the image plane; (**c**) the desired displacement in Cartesian space along the X-axis; (**d**) the desired displacement in Cartesian space along the Y-axis; and (**e**) the desired rotation (yaw angle) along the Z-axis.

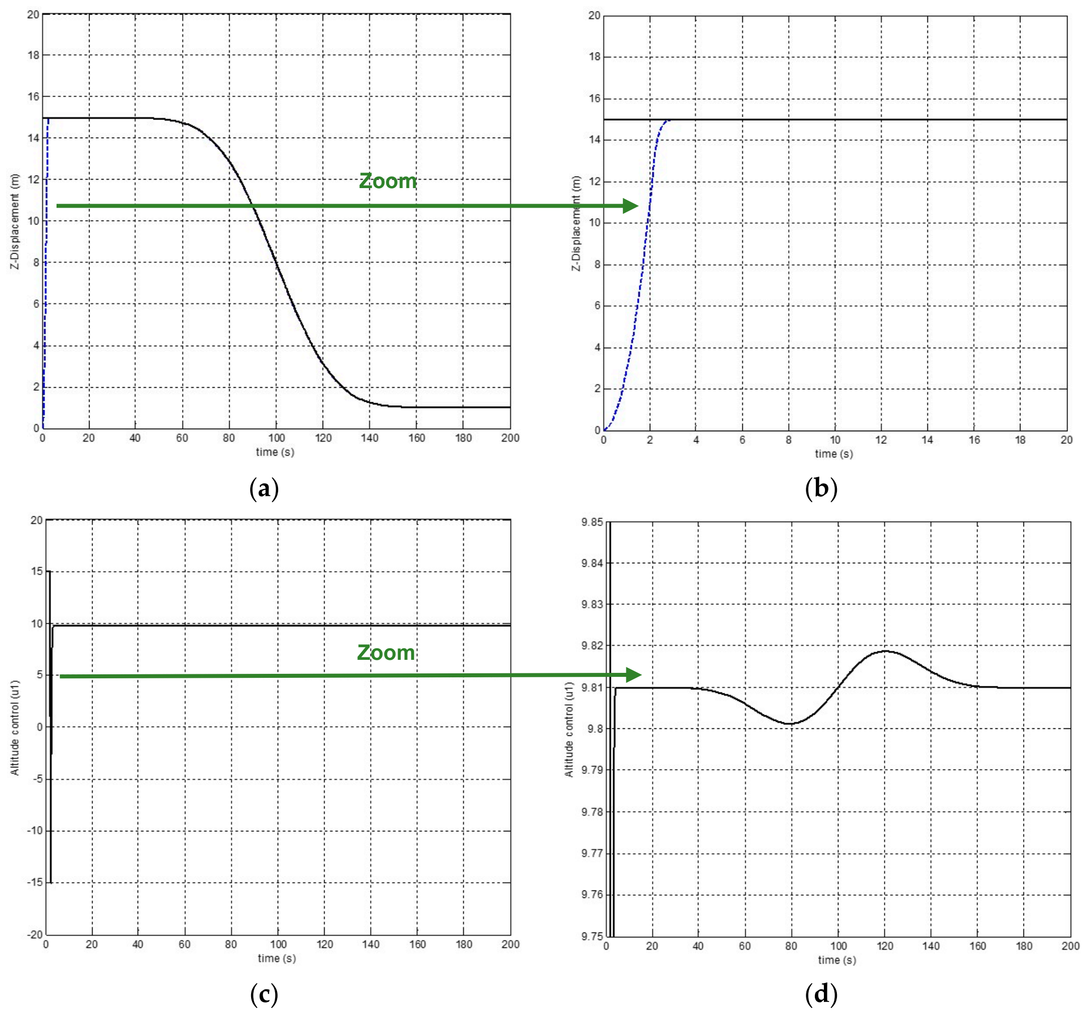


Figure 7. Control of the quadrotor altitude: (**a**,**b**) evolution of the desired altitude and the performed altitude of the quadrotor; and (**c**,**d**) evolution of the altitude control law.

To achieve the generated trajectory for the variables x, y, and Ψ, a flatness-based control was proposed. The gain parameters, which define the dynamics of the errors, are listed in Table 1.

Table 1. The gain parameters.

Gain	k_{21}	k_{22}	k_{23}	k_{24}	k_{31}	k_{32}	k_{33}	k_{34}	k_{41}	k_{42}
Value	4	6	4	1	4	6	4	1	2	1

Figure 8a–d illustrate the desired trajectories (the displacement along x, displacement along y, and orientation along the z-axis) as well as the trajectories actually executed by the quadrotor. It is clear that the proposed method effectively ensures the pursuit of the generated trajectory. Figure 8e–g present the evolution of the control laws that ensure the pursuit of the desired trajectories. Figure 8h,i represent the variation of the roll angle and pitch angle, respectively. It is evident that these two variables remain sufficiently small during the trajectory tracking.

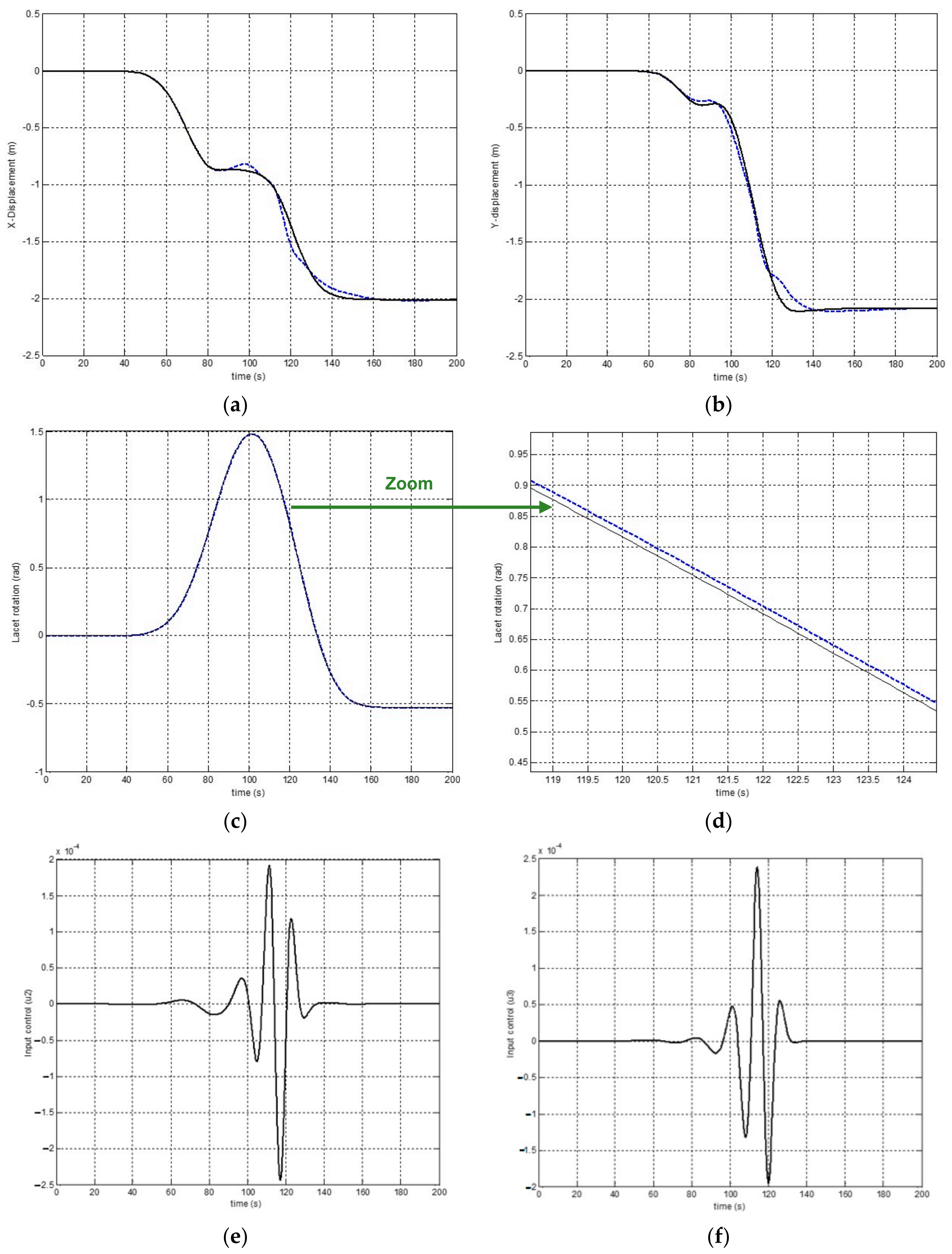
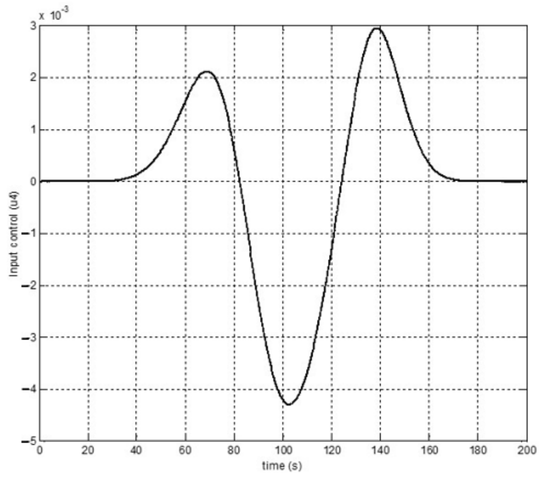
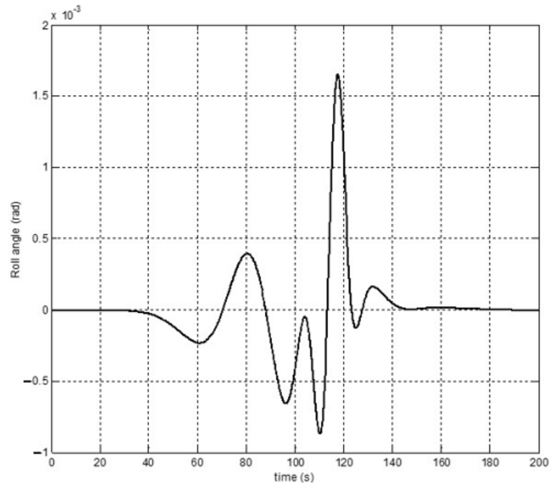


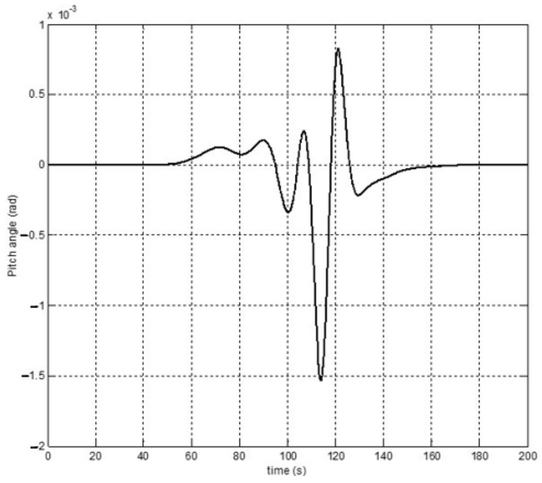
Figure 8. *Cont*.



(**g**)



(**h**)



(**i**)

Figure 8. Generated trajectory performed by the quadrotor: (**a**,**b**) desired (in black) and performed (in blue) trajectories along the x-axis and y-axis, respectively; (**c**,**d**) desired (in black) and performed (in blue) of the yaw angle; (**e**) input control u_2; (**f**) input control u_3; (**g**) input control u_4; (**h**) roll angle variation; and (**i**) pitch angle variation.

5. Experimental Results

The strategy adopted in this section is similar to that addressed in the simulation section. Starting from the very conclusive simulation results presented in Section 4, we explore in this section the performance of our proposed control law on a real platform. We use the DJI Phantom 1 quadrotor for this. We configure the onboard camera to continuously send a 240 $\times$ 320 pixel stream on the 2.4 GHz frequency with a rate of 15 frames per second. The desire to reduce the calculations related to image segmentation drove the choice of this minimal configuration. This segmentation must provide, in real time, the position of the center of gravity of the target. The target is a black mobile robot, which contrasts significantly with the color of the navigation space (bare terrain). The extraction of the target is carried out thanks to a simple binarization of the image followed by a morphological opening operation (erosion and dilation). This last operation is necessary to marginalize the shadow of the quadrotor as best as possible. Finally, a selection based on the proportion of extracted regions allows the identification of the target and the calculation of its center of gravity. Figure 9 shows the binarization and selection operations of the target based on the size criterion.

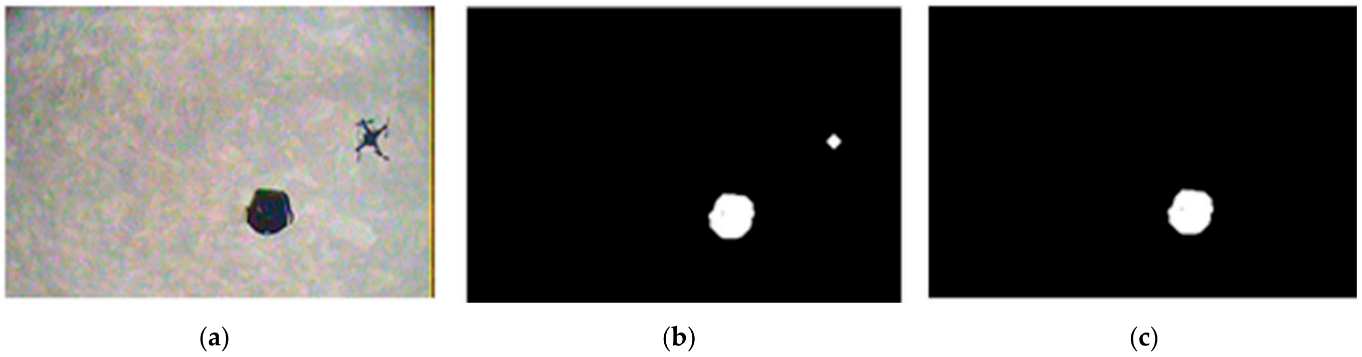
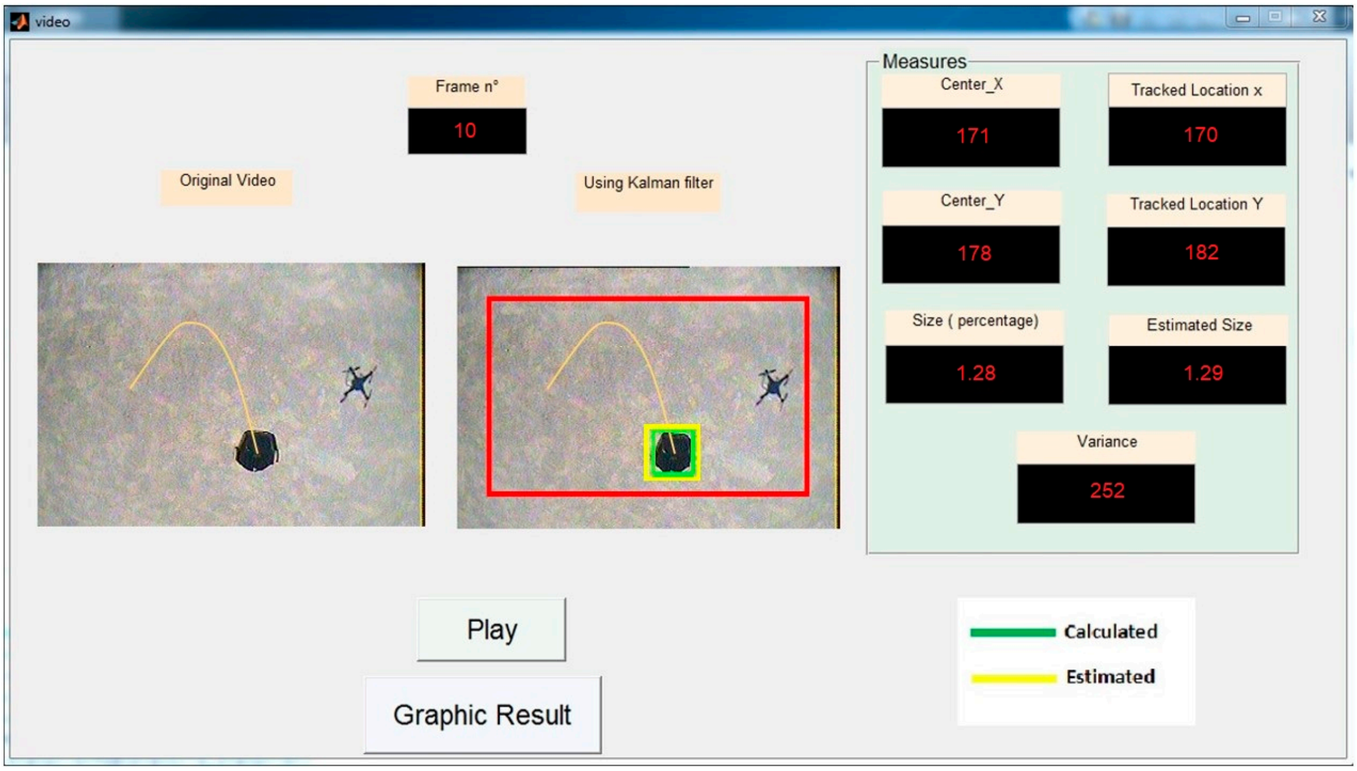


Figure 9. Target recognition: (**a**) original image; (**b**) binarization and opening; and (**c**) target selection based on the size criterion.

To control the quadrotor with the command values calculated using the proposed algorithm, we designed an electronic card that was integrated into the control joystick. This card receives the necessary thrusts via a Bluetooth connection and transforms them into four voltages to power the four motors of the quadrotor. This design includes a PIC18F2520 microcontroller, a DAC8564 digital-to-analogue converter, and a RN42 Bluetooth module. The generated thrusts are deduced based on Equation (2), with $m = 0.670$ kg, $l = 0.175$ m, $I1 = I2 = 0.0137$ kg.m^2, $I3 = 0.0231$ kg.m^2, and $C = 0.4$. Equation (41) enables the conversion of the generated commands for each motor into thrust values.

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{pmatrix} 0.1675 & -0.0196 & -0.0196 & 0.0144 \\ 0.1675 & -0.0196 & 0.0196 & -0.0144 \\ 0.1675 & 0.0196 & 0.0196 & 0.0144 \\ 0.1675 & 0.0196 & -0.0196 & -0.0144 \end{pmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}. \tag{41}$$

The desired trajectory in the image plane is built using Equations (36)–(40) with $x_i^* = 171$, $y_i^* = 178$, $t_i = 0.66s$, $x_f^* = 108, y_f^* = 77$, and $t_f = 2$. Figure 10 shows the results of an experiment in which the DJI Phantom 1 takes off to reach an altitude of 15 m. Once the target has been detected (which will give us a starting point of $x_i^* = 171$, $y_i^* = 178$), a trajectory is generated in the image plane (the yellow curve). The mobile robot playing the role of the target is static throughout the experience. Its orientation, as well as its apparent size in successive images, are tracked by the movement of the quadrotor. The task of the quadrotor during this experience is to make the necessary movements to ensure that the target remains on the desired trajectory. In this phase, we introduced a Kalman filter to estimate the position and size of the target. This filter has proven to be very useful because the analogue transmission of the live streaming is sometimes very noisy. This makes the target detection phase impossible. Figure 10b perfectly illustrates this case. Despite the total absence of the target, our algorithm continues to generate the necessary commands based entirely on the estimated position (the yellow square). In Figure 10a,c–e, the green square shows the success of the target identification phase. We notice that the size of the target is almost constant during the first half of the flight (Figure 10a–c). This is completely normal since we have separated the u_1 command responsible for altitude from the rest of the commands. However, the size of the target suddenly changes in the second half of the flight (it became 10 to 15% smaller; see Figure 10d,e). This is due to a gust of wind. Despite this, the quadrotor continues to follow the desired trajectory by manipulating the u_2, u_3, and u_4 commands. This result shows in an experimental manner the benefit of the separation of commands proposed in the diagram of Figure 4. Finally, Figure 11 shows the desired and achieved trajectories. If we take into account the DJI Phantom 1's experience with wind gusts and the target's intermittent absence, the superposition of the two trajectories is almost perfect.



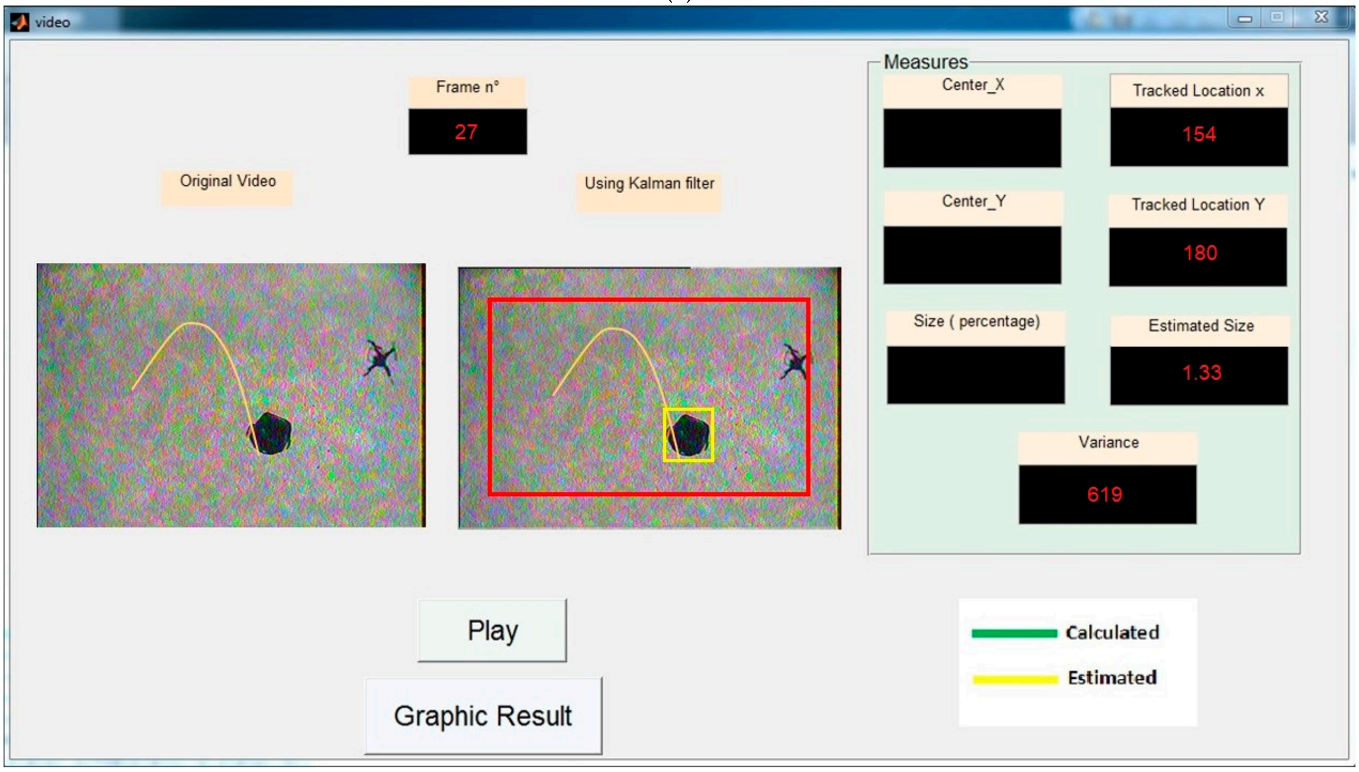
Measures

Center_X	Tracked Location x
171	170
Center_Y	**Tracked Location Y**
178	182
Size (percentage)	**Estimated Size**
1.28	1.29

Variance: 252

🟩 Calculated
🟨 Estimated

(**a**)



Measures

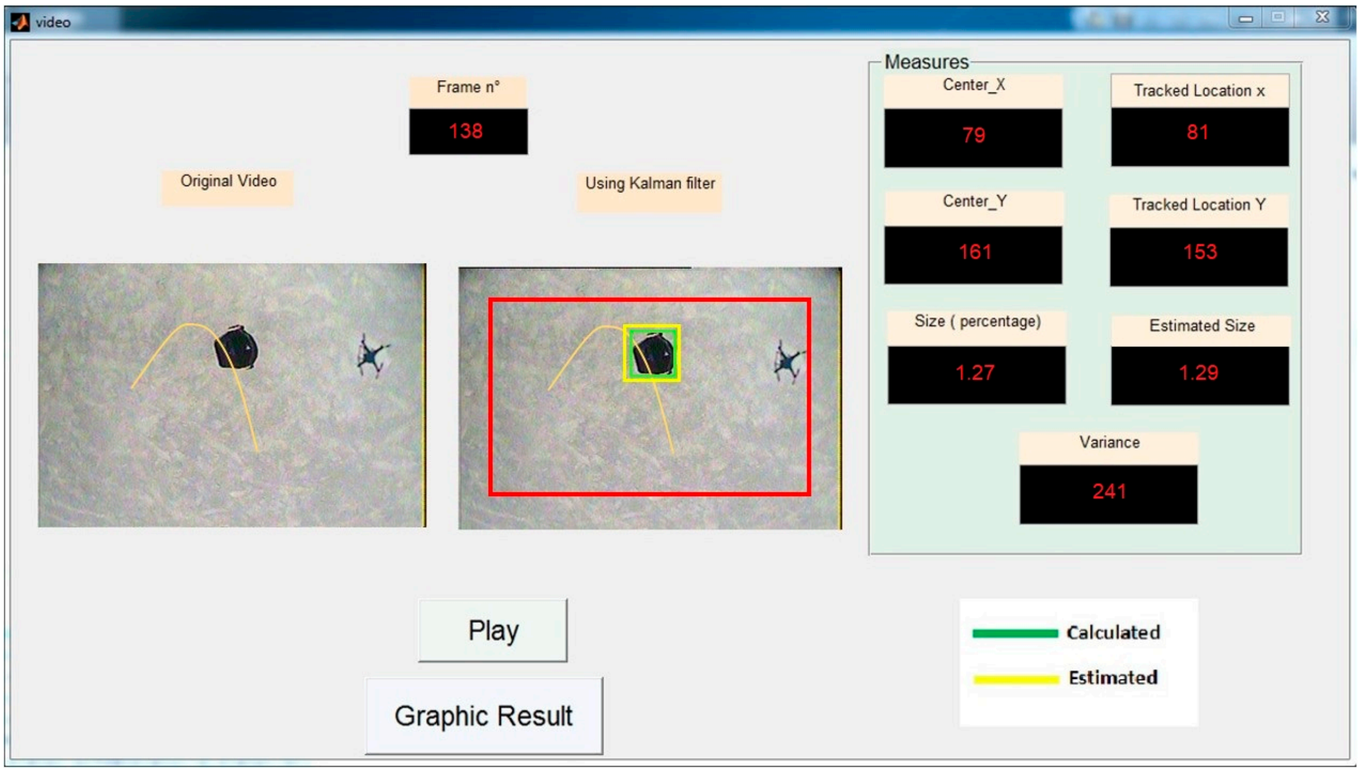
Center_X	Tracked Location x
	154
Center_Y	**Tracked Location Y**
	180
Size (percentage)	**Estimated Size**
	1.33

Variance: 619

🟩 Calculated
🟨 Estimated

(**b**)

Figure 10. *Cont.*

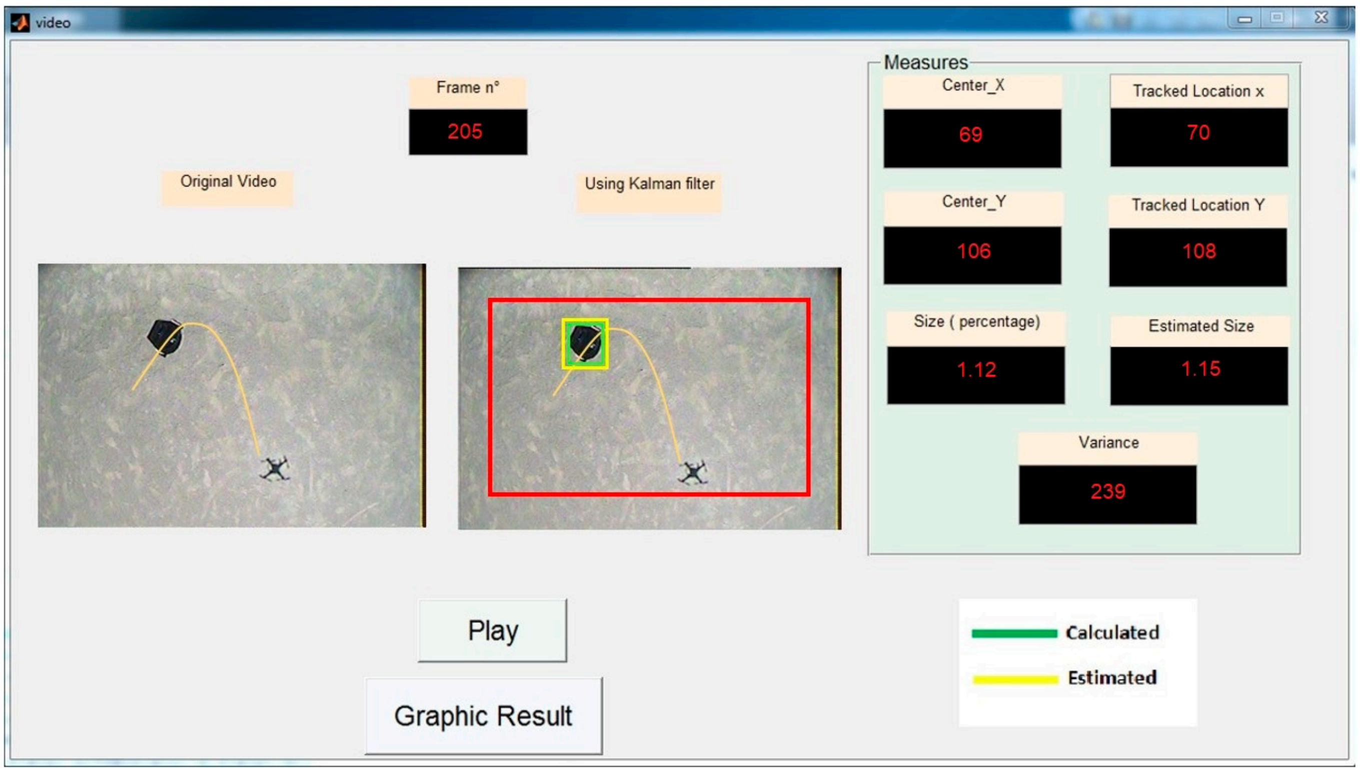


Frame n° 138

Measures

Center_X	79
Tracked Location x	81
Center_Y	161
Tracked Location Y	153
Size (percentage)	1.27
Estimated Size	1.29
Variance	241

(**c**)



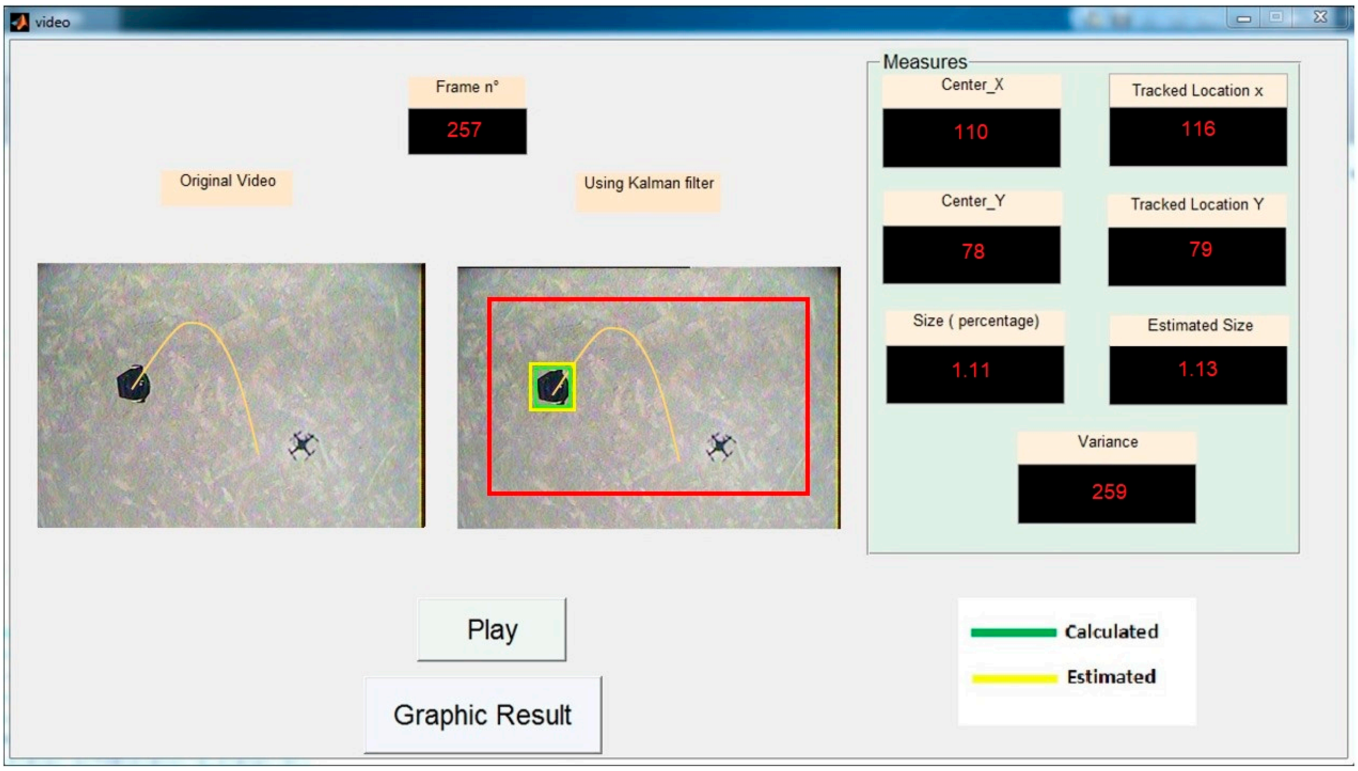
Frame n° 205

Measures

Center_X	69
Tracked Location x	70
Center_Y	106
Tracked Location Y	108
Size (percentage)	1.12
Estimated Size	1.15
Variance	239

(**d**)

Figure 10. *Cont*.



Frame n°
257

Original Video

Using Kalman filter

Measures

Center_X	Tracked Location x
110	116
Center_Y	Tracked Location Y
78	79
Size (percentage)	Estimated Size
1.11	1.13

Variance
259

🟩	Calculated
🟨	Estimated

(**e**)

Figure 10. IBVS-Based Image Plan Path planning and tracking with the DJI Phantom 1: (**a**) tracking start; (**b**) continuation of the tracking without target recognition; (**c**) tracking reaching halfway through the flight (frame 138); (**d**) tracking under wind gusts (target size decreases); and (**e**) the achievement of the desired trajectory. For all sub-figures, the red rectangle delimits the zone of interest in which the target recognition operation is carried out. The yellow curve represents the desired trajectory. The green square demarcates the recognized target. The yellow square represents the estimated size and position of the target.

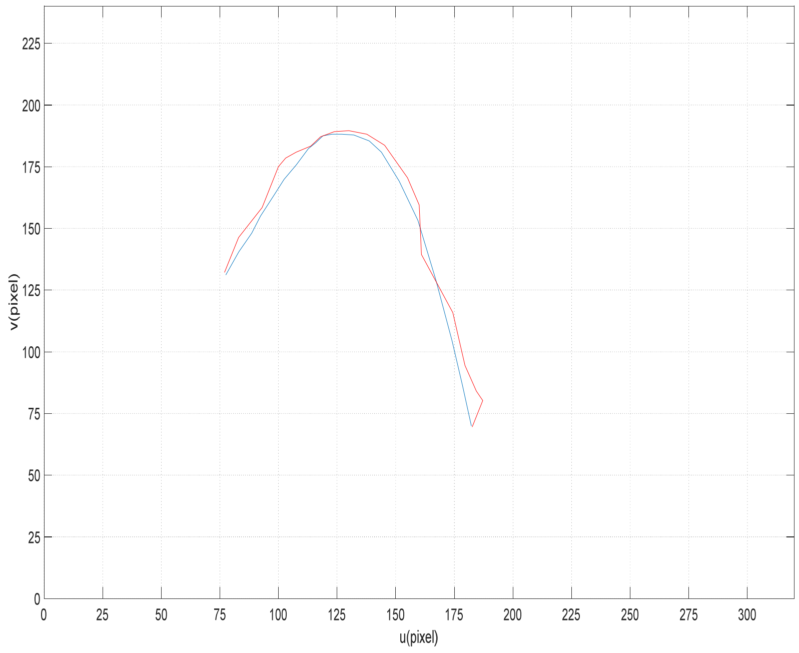


Figure 11. Desired (in blue) and performed (in red) trajectories.

6. Discussion

Our work primarily focuses on planning and implementing a specific trajectory from the image plane (a 2D space) to a 3D space. The research community specializing in 2D control widely acknowledges this issue as intricate, given that a specific path in the image space can lead to unpredictable robot motion in Cartesian space.

Although we have recently resolved this issue for a mobile robot [28], it continues to pose a significant challenge for a quadrotor. Indeed, our previous approach was based on the assumption that the coordinates of an image point (u and v) could serve as a flat output for a two-wheeled robot. This allowed us to establish a relationship between the flat output space and the workspace (2D/2D). Nevertheless, this methodology is not suitable for the scenario involving a quadrotor.

We employ a methodology to tackle this problem, which involves investigating the feasibility of decoupling altitude control from the other variables, thereby reducing the workspace. Furthermore, due to the interconnection of the quadrotor's movements (pitch, roll, and yaw), it is not feasible to establish a reversible relationship between the outputs and inputs. This highlights the intricate nature of the task. We would like to emphasize that trajectory planning entails applying dynamics to the trajectory.

In order to address these challenges, we opted to utilize a virtual dynamics system that is based on the kinematic model of a two-wheeled mobile robot for the trajectory generation. We chose this approach because existing evidence suggests that this framework can establish a reversible relationship between the outputs and inputs. Following that, we were able to successfully generate the required motions in 2D space to accomplish this trajectory in the 2D image plane. Subsequently, it became imperative to establish the mapping between the motions produced in the two-dimensional (2D) realm of the virtual robot and the motions occurring in the three-dimensional (3D) realm of the quadrotor.

Our methodology effectively addresses the issue of energy conservation and mitigates disturbances that may impact the quadrotor, such as wind gusts. Once we have executed the required maneuvers to reach the desired destination, we were able to initiate a descent in altitude.

Another notable contribution to this work is the use of a camera as an information source for guiding the quadrotor's control decisions. Nevertheless, this presents a practical obstacle in terms of control frequency. Quadrotor control necessitates a high control frequency, whereas image processing necessitates a lower control frequency, resulting in potential synchronization issues. In our control approach, we addressed this problem by implementing an offline planning strategy, ensuring that the time required for image processing does not impact the control frequency.

As previously stated, we have the ability to choose a trajectory within the image plane. In a practical implementation, the control tower must manually specify the trajectory. The task involves tracing the trajectory to avoid obstacles shown on the interface screen and then formulating an analytical expression to integrate into the control loop. Given our system's outdoor operation and susceptibility to external disturbances, it is imperative to improve the control algorithm in order to guarantee its robustness.

7. Conclusions

In this paper, we present a novel method for trajectory planning and tracking in the image plane for the visual control of a quadrotor. This approach relies on the concept of differential flatness, allowing the separation of altitude control from the control of the other variables. The control strategy is implemented in two steps. The first step involves determining the necessary displacements that the quadrotor must execute to follow the trajectory in the image plane, conducted offline. To facilitate the trajectory planning, a new visual servoing method was proposed, demonstrating that a single point of the target to be reached is sufficient for the visual control. The second step aims to ensure the asymptotic convergence of the trajectory generated in Cartesian space with a given level of robustness. To overcome the challenges related to underactuated control and strong coupling of the quadrotor, a flatness-based control was introduced. This approach ensures the controllability of the system and guarantees the asymptotic convergence of the generated trajectory. The simulations performed on MATLAB using RVCTOOLS library and experiments performed with the DJI Phantom 1 demonstrate the effectiveness of the proposed method.

Author Contributions: Conceptualization, H.M., A.A. and K.K.; methodology, H.M. and K.K.; software, A.A. and S.A.; validation, M.A., S.A. and A.A.; formal analysis, M.A.; investigation, A.S.; resources, A.A.; data curation, K.K.; writing—original draft preparation, H.M., A.A. and K.K.; writing—review and editing, M.D.A. and K.K.; visualization, A.S.; supervision, K.K.; project administration, A.A.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia for funding this research work through project number 223202.

Data Availability Statement: Data are contained within the article.

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

References

1. Wei, Z.; Zhu, M.; Zhang, N.; Wang, L.; Zou, Y.; Meng, Z.; Wu, H.; Feng, Z. UAV-Assisted Data Collection for Internet of Things: A Survey. *IEEE Internet Things J.* **2022**, *9*, 15460–15483. [CrossRef]
2. Ceren, Z.; Altuğ, E. Image Based and Hybrid Visual Servo Control of an Unmanned Aerial Vehicle. *J. Intell. Robot. Syst.* **2011**, *65*, 325–344. [CrossRef]
3. Metni, N.; Hamel, T. A UAV for Bridge Inspection: Visual Servoing Control Law with Orientation Limits. *Autom. Constr.* **2007**, *17*, 3–10. [CrossRef]
4. Arafat, M.Y.; Moh, S. Routing Protocols for Unmanned Aerial Vehicle Networks: A Survey. *IEEE Access* **2019**, *7*, 99694–99720. [CrossRef]
5. Fraundorfer, F.; Heng, L.; Honegger, D.; Lee, G.H.; Meier, L.; Tanskanen, P.; Pollefeys, M. Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 October 2012. [CrossRef]
6. Bi, R.; Gan, S.; Yuan, X.; Li, R.; Gao, S.; Yang, M.; Luo, W.; Hu, L. Multi-View Analysis of High-Resolution Geomorphic Features in Complex Mountains Based on UAV–LiDAR and SfM–MVS: A Case Study of the Northern Pit Rim Structure of the Mountains of Lufeng, China. *Appl. Sci.* **2023**, *13*, 738. [CrossRef]
7. Tokekar, P.; Hook, J.V.; Mulla, D.; Isler, V. Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. *IEEE Trans. Robot.* **2016**, *32*, 1498–1511. [CrossRef]
8. Costello, B.; Osunkoya, O.O.; Sandino, J.; Marinic, W.; Trotter, P.; Shi, B.; Gonzalez, F.; Dhileepan, K. Detection of Parthenium Weed (*Parthenium hysterophorus* L.) and Its Growth Stages Using Artificial Intelligence. *Agriculture* **2022**, *12*, 1838. [CrossRef]
9. Chaumette, F.; Hutchinson, S. Visual Servo Control. I. Basic Approaches. *IEEE Robot. Autom. Mag.* **2006**, *13*, 82–90. [CrossRef]
10. Garcia, A.; Mattison, E.; Ghose, K. High-Speed Vision-Based Autonomous Indoor Navigation of a Quadcopter. In Proceedings of the 2015 International Conference on Unmanned Aircraft Systems (ICUAS), Denver, CO, USA, 9–12 June 2015. [CrossRef]
11. Iacono, M.; Sgorbissa, A. Path Following and Obstacle Avoidance for an Autonomous UAV Using a Depth Camera. *Robot. Auton. Syst.* **2018**, *106*, 38–46. [CrossRef]
12. Mercado, D.; Castillo, P.; Lozano, R. Sliding Mode Collision-Free Navigation for Quadrotors Using Monocular Vision. *Robotica* **2018**, *36*, 1493–1509. [CrossRef]
13. Park, J.; Kim, Y. Collision Avoidance for Quadrotor Using Stereo Vision Depth Maps. *IEEE Trans. Aerosp. Electron. Syst.* **2015**, *51*, 3226–3241. [CrossRef]
14. Yang, X.; Chen, J.; Dang, Y.; Luo, H.; Tang, Y.; Liao, C.; Chen, P.; Cheng, K.-T. Fast Depth Prediction and Obstacle Avoidance 417 on a Monocular Drone Using Probabilistic Convolutional Neural Network. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 156–167. [CrossRef]
15. Courbon, J.; Mezouar, Y.; Guénard, N.; Martinet, P. Vision-Based Navigation of Unmanned Aerial Vehicles. *Control Eng. Ing Pract.* **2010**, *18*, 789–799. [CrossRef]
16. Do, T.; Carrillo-Arce, L.C.; Roumeliotis, S.I. Autonomous Flights through Image-Defined Paths. *Springer Proc. Adv. Robot.* **2017**, *1*, 39–55. [CrossRef] [PubMed]
17. Kozak, V.; Pivonka, T.; Avgoustinakis, P.; Majer, L.; Kulich, M.; Preucil, L.; Camara, L.G. Robust Visual Teach and Repeat Navigation for Unmanned Aerial Vehicles. In Proceedings of the 2021 European Conference on Mobile Robots (ECMR), Virtual, 31 August–3 September 2021. [CrossRef]
18. Nguyen, T.; Mann, G.K.I.; Gosine, R.G.; Vardy, A. Appearance-Based Visual-Teach-And-Repeat Navigation Technique for Micro Aerial Vehicle. *J. Intell. Robot. Syst.* **2016**, *84*, 217–240. [CrossRef]

19. Warren, M.; Greeff, M.; Patel, B.; Collier, J.; Schoellig, A.P.; Barfoot, T.D. There's No Place Like Home: Visual Teach and Repeat for Emergency Return of Multirotor UAVs During GPS Failure. *IEEE Robot. Autom. Lett.* **2019**, *4*, 161–168. [CrossRef]
20. Toro-Arcila, C.A.; Becerra, H.M.; Arechavaleta, G. Visual Path Following with Obstacle Avoidance for Quadrotors in Indoor Environments. *Control Eng. Pract.* **2023**, *135*, 105493. [CrossRef]
21. López, J.; Dormido, R.; Dormido, S.; Gómez, J.P. A RobustH∞Controller for an UAV Flight Control System. *Sci. World J.* **2015**, *2015*, 403236. [CrossRef] [PubMed]
22. Fliess, M.; Lévine, J.; Martin, P.; Rouchon, P. On Differentially Flat Nonlinear Systems. *IFAC Proc. Vol.* **1992**, *25*, 159–163. [CrossRef]
23. Chamseddine, A.; Zhang, Y.; Rabbath, C.A.; Join, C.; Theilliol, D. Flatness-Based Trajectory Planning/Replanning for a Quadrotor Unmanned Aerial Vehicle. *IEEE Trans. Aerosp. Electron. Syst.* **2012**, *48*, 2832–2848. [CrossRef]
24. Li, T.; Xia, Y.; Ma, D. Flatness-Based Target Tracking for a Quadrotor Unmanned Aerial Vehicle. *IFAC-PapersOnLine* **2015**, *48*, 874–879. [CrossRef]
25. Abadi, A.; El Amraoui, A.; Mekki, H.; Ramdani, N. Guaranteed Trajectory Tracking Control Based on Interval Observer for Quadrotors. *Int. J. Control* **2019**, *93*, 2743–2759. [CrossRef]
26. Hagenmeyer, V.; Delaleau, E. Exact Feedforward Linearization Based on Differential Flatness. *Int. J. Control* **2003**, *76*, 537–556. [CrossRef]
27. Chamseddine, A.; Zhang, Y.; Rabbath, C.A.; Theilliol, D. Trajectory Planning and Replanning Strategies Applied to a Quadrotor Unmanned Aerial Vehicle. *J. Guid. Control Dyn.* **2012**, *35*, 1667–1671. [CrossRef]
28. Albekairi, M.; Mekki, H.; Kaaniche, K.; Yousef, A. An Innovative Collision-Free Image-Based Visual Servoing Method for Mobile Robot Navigation Based on the Path Planning in the Image Plan. *Sensors* **2023**, *23*, 9667. [CrossRef]