Visual Path Following With Obstacle Avoidance for Quadcopters in Indoor Environments

Carlos A. Toro-Arcila[a,*], Héctor M. Becerra[b], Gustavo Arechavaleta[a]

[a] *Centro de Investigación y de Estudios Avanzados del IPN, Saltillo, Coahuila, México*
[b] *Centro de Investigación en Matemáticas (CIMAT), A.C., Jalisco S/N, Col. Valenciana, 36023, Guanajuato, Gto., México*

Abstract

In this paper, we propose a hierarchical visual servo control scheme for following visual paths with quadcopters in indoor environments. The control scheme is able to deal with three tasks simultaneously, considering as order of hierarchy: first, collision avoidance task, second visibility task (keep point features in the camera's field of view), and finally, visual servoing task. The strategy allows a commitment of active tasks (also the smooth deactivation of tasks) for avoiding conflicts between them. The proposed scheme uses a monocular RGB front-facing camera as single sensor, i.e., only 2D information is required for the control law computation. That information is used for online reactive visual path following, i.e., no 3D reconstruction of the environment is required, and no path or motion planning for obstacle avoidance is performed while the robot follows the visual path. The effectiveness of the proposed control scheme is supported by a stability analysis and experiments with a low-cost commercial quadcopter in scenarios with static obstacles.

Keywords: Visual servoing, image-based collision avoidance, visibility control, homography-based control, unmanned aerial vehicles, monocular vision.

1. Introduction

Nowadays, quadcopters are increasingly used for tasks such as surveillance, inspection of structures, inventories, video filming, and fire detection. It has caused greater interest in the research area in providing them with a higher degree of autonomy in their navigation.

The goal of this work is to drive a quadcopter to a target pose using visual feedback in indoor environments. However, in some situations, other tasks, such as collision avoidance and visibility (maintenance of the target in the camera's field of view), could appear, and the different tasks must be solved simultaneously without causing conflicts in the flight behavior of the robot. Moreover, the tasks must have an order of priority, for instance, if an obstacle appears, it is more important for the quadcopter not to collide than following the direction to the target. In this work, we adopt the strategy defined by Lee et al. (2012) to manage the commitment of active tasks (also the deactivation of tasks), which consists in defining a strict hierarchy among them.

The main control strategy used in this work relies on visual servoing. That strategy addresses the task of driving a robot to a target pose from an initial one using visual information feedback, which can be obtained through a camera mounted directly on the robot. Therefore, the movement of the camera is induced by the displacement of the robot (Chaumette and Hutchinson, 2006).

Frequently, during navigation with aerial robots in indoor spaces, the need to avoid static and moving obstacles arises. In Garcia et al. (2015), navigation based on vanishing points is used to move a quadcopter through corridors without hitting walls. In Park and Kim (2015); Iacono and Sgorbissa (2018); Mercado et al. (2018); Yang et al. (2021), different sensors are used to obtain 3D information about the environment, such as stereo, monocu-

*Corresponding author
Email address: carlos.toro@cinvestav.edu.mx (Carlos A. Toro-Arcila)

lar, and RGB-D cameras. In addition, they use various techniques to reconstruct a map of the environment, such as simultaneous localization and mapping (SLAM). These maps are used to perform navigation and obstacle avoidance tasks using motion planning and potential fields. In Loianno et al. (2017); Lin et al. (2020), stereo cameras are used as the main sensor. Then the visual, depth, and inertial information is combined to navigate and avoid obstacles using motion planning and model predictive control. In Lin and Peng (2021); Tang et al. (2021); Chiang et al. (2021) optical flow is used for obstacle detection and to estimate the velocity between the robot and the obstacles. Then path planning, visual and inertial information, and adaptive control are used for navigation and obstacle avoidance tasks.

Some works have addressed the vision-based navigation problem using a strategy called teach and repeat without regard to obstacle avoidance (Courbon et al., 2010; Do et al., 2015; Nguyen et al., 2016; Warren et al., 2019; Kozák et al., 2021). This strategy consists of capturing, storing, and ordering a set of key images of the environment (visual path), which is extracted from a visual memory (topological map) of the environment. Then, the robot navigation task is defined as a set of visual sub-tasks, whereby the quadcopter moves from the pose associated with the current image to the desired pose associated with each target image. In Nitsche et al. (2020), in addition to teach-and-repeat, they use visual information obtained from a stereo camera and inertial information to perform a 3D reconstruction of the environment. In Do et al. (2019), the teach and repeat technique is used, including visual and inertial information to generate the robot movement, and also to avoid obstacles. The teach and repeat strategy requires a learning stage to capture the topology of the environment by means of key images. Then, the quadcopter uses these key images to autonomously navigate in the same environment. Thus, the strategy can be applied for tasks in which the quadcopter must follow the same path several times, as it is demonstrated in the context of security and surveillance (Ayub et al., 2018), in logistic processes, particularly to make stock inventory in warehouses (Škrinjar et al., 2018), in industrial factory inspection (Fehr et al., 2018; Martinez-Carranza and Rojas-Perez, 2022), and as a backup navigation method in the event of failure of the main navigation system (Warren et al., 2019).

Usually, during the execution of visual servo control tasks, point features might be lost as they get out of the camera's field of view. This is a well-known issue of the visual servo control technique due to the controller's dependence on point features. Therefore, a strategy is needed to maintain the features in the field of view. In Gans et al. (2011), visual servo control is used to keep multiple targets in the field of view of a camera mounted on a mobile ground robot. In Zheng et al. (2019), an approach based on barrier functions is used to satisfy visibility constraints while performing visual servo control tasks with a quadcopter with a downward-facing camera. In Liu et al. (2020), an anti-disturbance back-stepping restricted attitude controller is used to keep an object in the camera's field of view while a simulated quadcopter executes a visual servo control task. In Lim et al. (2020), target tracking is performed and obstacle augmentation is used to perform the avoidance task and avoid occlusion of the target by the obstacle.

In this work, we propose a hierarchical visual servo control scheme for visual path following tasks of quadcopters in indoor environments. The control scheme is able to deal with three tasks, considering an order of hierarchy: first, collision avoidance task, second visibility task, and finally visual task. The proposed approach contributes to the field of vision-based quadcopters' navigation in the following aspects: first, for all tasks, information from a monocular RGB front-facing camera is used as a single sensor. Therefore, there is no limit to the sensing range of the sensor. Second, the controller is based only on 2D information, and it is not necessary to perform any 3D reconstruction of the environment. Thus, our approach is not computationally demanding to calculate the control law. Third, the controller is generic for any quadcopter equipped with a monocular RGB front-facing camera and a low-level controller whose inputs are velocity commands in four degrees of freedom (linear velocities on the x, y and z axes and angular velocity in yaw). Fourth, the controller has been directly implemented on a low-cost commercial quadcopter without any additional sensor. Finally, the controller allows the quadcopter to perform online reactive visual path following, i.e., path planning or motion planning are not required. As a consequence, the proposed control algorithm is able to run at 55 Hz, with low power consumption and minimal hardware requirements.

To our knowledge, there is no hierarchical visual servo control for the visual path following that

involves collision avoidance, visibility, and visual tasks applied to quadcopters. The hierarchical control approach has been usually applied to manipulators; the proposed scheme contributes in the visual path following of quadcopters to avoid the switching between control laws each one dedicated to solve one task, avoiding discontinuous input velocities that can perturb the quadcopter motion. Given the challenge of using only a monocular camera and 2D information, the effective combination of tasks solutions in such a control scheme allows to ensure convergence of a visual servoing task, where the maintenance of image points in the field of view of the camera is crucial, but obstacle avoidance actions might jeopardize the task of reaching a destination using only visual feedback. Moreover, for the case where there are no obstacles in the predefined visual path, we take advantage of the *a priori* knowledge given by the predefined visual path to include a feed-forward component to the control law, which allows the quadcopter to complete the path in less time than just using a feedback component.

This paper presents an effective implementation of a visual control of quadcopters. However, some additional aspects could be addressed to improve the approach, in particular by considering modeling errors, uncertainties, and the influence of disturbances. To this end, an option is the use of learning-based methods such as those in Jiang et al. (2022); Song et al. (2022). Besides of the robustness provided by these kinds of methods, visual control could benefit from a better management of the convergence rate and the possibility to predefine the control performance.

The organization of the remaining sections is as follows. In Section 2, we describe a visual servo control applied to a quadcopter. In Section 3, we present the formulation of the hierarchical visual servo controller with three tasks and its respective stability analysis. In Section 4, we show the formulation of the controller for visual path following, with and without the hierarchical visual servo controller of Section 3. Finally, in Section 5, we present a series of experiments that validate the performance of the proposed control scheme.

2. Visual servo control of a quadcopter

In this work, we consider a conventional quadcopter that is a non-linear underactuated system, in which four flat outputs (three position coordinates and yaw angle) can be independently controlled

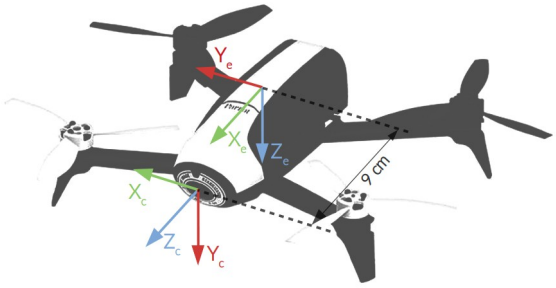


Figure 1: Parrot Bebop 2 quadcopter reference frames. Taken from Spindler and Gaumerais (2019).

(Mellinger and Kumar, 2011). In the literature, feedback linearization has been used to achieve decoupled control for each position coordinate of a quadcopter (Spitzer and Michael, 2021). Feedback linearization can be formulated in terms of velocities, accelerations, or thrusts as control inputs; in the first case, relying on a low-level velocity controller, the quadcopter dynamics for each position coordinate can be modeled as a decoupled single-integrator, as it has been done in Trujillo et al. (2021); Vargas et al. (2022). In this work, we consider modeling the position of the quadcopter and the angle of yaw $\boldsymbol{x} = [x\ y\ z\ \psi]^T$ through a single-integrator for each coordinate and velocities as control inputs $\boldsymbol{\nu} = [v_x\ v_y\ v_z\ \omega_z]^T$, such that

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\nu}(t) \in \mathbb{R}^4. \tag{1}$$

According to Chaumette and Hutchinson (2006), the visual servo control refers to the use of computer vision data to control the movement of a robot. We consider a quadcopter carrying a camera looking forward such that the camera movement is induced by the robot movement. This configuration is known as eye-in-hand.

Consider a pose-regulation problem where the quadcopter disposes of a target image $\mathcal{I}^*$ associated with a desired pose, which must be reached from the current quadcopter pose associated with the current image $\mathcal{I}$. Thus, the main objective of a visual servo controller is to minimize a task function $\boldsymbol{e}(t)$ defined in terms of visual information extracted from images $\mathcal{I}$ and $\mathcal{I}^*$. We use image points as visual information, which can be obtained through classical detection and description

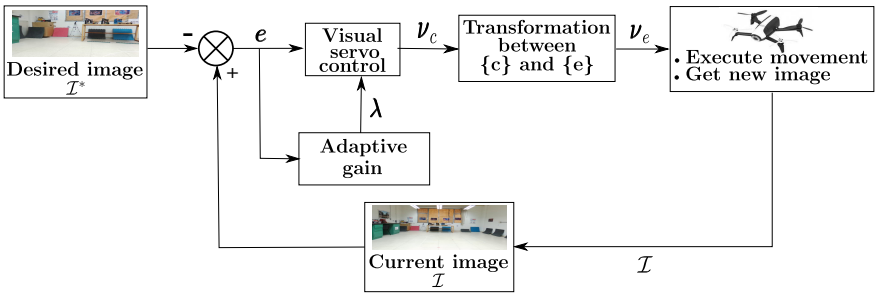


Figure 2: Visual servo control scheme for a quadcopter.

algorithms such as ORB, SIFT, SURF, etc. In this work, we propose to use a homography-based visual servo control (Benhimane and Malis, 2007) applicable to quadcopters, where the elements of the task function vector $\boldsymbol{e} = [\boldsymbol{e}_v^T \ \boldsymbol{e}_w^T]^T \in \mathbb{R}^6$ are defined as follows:

$$\begin{aligned} \boldsymbol{e}_v &= (\boldsymbol{H} - \boldsymbol{I})\boldsymbol{m}^*, \\ [\boldsymbol{e}_w]_\times &= \boldsymbol{H} - \boldsymbol{H}^T, \end{aligned} \tag{2}$$

where $\boldsymbol{H}$ is the Euclidean homography matrix relating images $\mathcal{I}$ and $\mathcal{I}^*$, $\boldsymbol{m}^* = [x \ y \ 1]^T = \boldsymbol{K}^{-1}\boldsymbol{p}^*$ with $\boldsymbol{p}^*$ any point in pixels homogeneous coordinates of the target image belonging to the plane that defines $\boldsymbol{H}$. Note that we assume that the parameters of the camera calibration matrix $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ are known. The operator $[\ \cdot\ ]_\times \in \mathbb{R}^{3\times3}$ represents the skew-symmetric matrix. Also, the time-derivative of the task function can be written as $\dot{\boldsymbol{e}}(t) = \boldsymbol{L}\boldsymbol{\nu}_c$ where $\boldsymbol{L} \in \mathbb{R}^{6\times6}$ is an interaction matrix that depends on 3D parameters (distance to the plane Z^* and normal of the plane $\boldsymbol{n}^*$). The camera twist is $\boldsymbol{\nu}_c = [\boldsymbol{v}^T \ \boldsymbol{w}^T]^T \in \mathbb{R}^6$, which is composed by the linear and angular velocity vectors $\boldsymbol{v} \in \mathbb{R}^3$ and $\boldsymbol{w} \in \mathbb{R}^3$, respectively.

In Benhimane and Malis (2007), it is proposed to linearize the relationship $\dot{\boldsymbol{e}}(t) = \boldsymbol{L}\boldsymbol{\nu}_c$ around $\boldsymbol{e} = 0$ to avoid calculating $\boldsymbol{L}$. Accordingly, an homography-based control law is as follows:

$$\boldsymbol{\nu}_c = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{w} \end{bmatrix} = - \begin{bmatrix} \lambda_v \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_w \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_v \\ \boldsymbol{e}_w \end{bmatrix}, \tag{3}$$

with $\lambda_v > 0$ and $\lambda_w > 0$. This control law depends only of the task function $\boldsymbol{e}$, and can be computed using the expressions in (2) given the pair of images $\mathcal{I}$ and $\mathcal{I}^*$. It is shown in Benhimane and Malis (2007) that the control law (3) yields a local exponential convergence of the task function $\boldsymbol{e}$ to zero. Moreover, the control law (3) has been successfully applied to control different robotic platforms equipped with monocular vision systems such a six-degree-of-freedom manipulator robot (Benhimane and Malis, 2007), and a humanoid robot (Delfin et al., 2014, 2016). The experimental results reported show that the region of attraction of (3) is large despite the theoretical local stability. In addition, there are practical benefits of the control law (3); in particular, it does not depend on the 3D structure of the scene (no depth measurements are required). Also, it can be applied to general scenes by calculating the homography from a virtual plane as suggested by Malis and Chaumette (2000). Unlike classical image-based visual servo control laws that use an interaction matrix (Chaumette and Hutchinson, 2006), the control law (3) does not require to be adapted when some points leave the camera's field of view. Thus, it is less sensitive to point loss during task execution; however, a minimum number of points must be kept in view.

The control gains λ_v and λ_w in (3) must be positive values, in many cases constant values are sufficient. However, in experiments with a quadcopter, we have found that as soon as the task error $\boldsymbol{e}$ approaches zero, the pose of the quadcopter oscillates because the velocities obtained from the control law are not sufficient to keep a error low. This makes it difficult for the quadcopter to complete the task with accuracy. According to Kermorgant and Chaumette (2013) a way to mitigate this issue is through the use of an adaptive gain, which depends on $\|\boldsymbol{e}\|$, and is given by:

$$\lambda(\|\boldsymbol{e}\|) = (\lambda_0 - \lambda_\infty) e^{-\frac{\lambda_0'}{\lambda_0 - \lambda_\infty}\|\boldsymbol{e}\|} + \lambda_\infty, \tag{4}$$

where $\boldsymbol{e}$ is the task error vector calculated from equation (2). $\|\boldsymbol{e}\|$ is the norm of the task error vector, λ_0 is the zero gain, to very low error values, $\lambda_\infty = \lim_{\|\boldsymbol{e}\|\to\infty} \lambda(\|\boldsymbol{e}\|)$, λ_0' is the slope of λ at $\|\boldsymbol{e}\| = 0$. In this work, we compute a single adaptive gain using the error vector $\boldsymbol{e}$, and apply this

gain to each coordinate. However, in some cases, a constant value or scaling can be added in order to obtain the desired behavior of the quadcopter.

It is worth noting that the output of the visual control scheme $\boldsymbol{\nu}_c$ obtained from the equation (3) is the instantaneous velocity of the camera, expressed in the reference frame of the camera $\{c\}$. We assume that the quadcopter has a low-level velocity control able to execute the commanded velocities expressed in the robot reference frame $\{e\}$. As defined in (1) and as happens in typical experimental platforms (Giernacki et al., 2020), only four velocities constitute the control inputs of a conventional quadcopter. The performance of the low-level velocity control is important for the accuracy of the visual servo control and we will detail some aspects of it in the experimental section. Figure 1 depicts the definition of the camera and quadcopter reference frames. Then, it is needed to transform the velocities given by the visual control from the camera reference frame to the robot reference frame. The following spatial motion transformation (6×6 matrix) allows expressing the vector $\boldsymbol{\nu}_c$ in the reference frame $\{e\}$:

$$ {}^{e}\boldsymbol{T}_{c} = \begin{bmatrix} {}^{e}\boldsymbol{R}_{c} & [{}^{e}\boldsymbol{t}_{c}]_{\times}{}^{e}\boldsymbol{R}_{c} \\ \boldsymbol{0}_{3\times 3} & {}^{e}\boldsymbol{R}_{c} \end{bmatrix}, \tag{5} $$

where ${}^{e}\boldsymbol{R}_{c} \in \mathbb{R}^{3\times 3}$ is the rotation matrix between the camera reference frame $\{c\}$ and the quadcopter reference frame $\{e\}$ (see Figure 1), which is defined as:

$$ {}^{e}\boldsymbol{R}_{c} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, $$

and ${}^{e}\boldsymbol{t}_{c} \in \mathbb{R}^{3}$ is the translation vector existing between the camera reference frame $\{c\}$ and the robot reference frame $\{e\}$, which is defined as ${}^{e}\boldsymbol{t}_{c} = \begin{bmatrix} 0.09 & 0 & 0 \end{bmatrix}^{T}$ for the Bebop 2 quadcopter.

Although the errors (2) and the control law (3) are formulated for 6 degrees of freedom, the quadcopter must be controlled by only four variables as expressed in (1) due to its underactuation. Then, the motions finally considered in the control are the decoupled motions of the quadcopter: x, y, z and yaw. We use the following selection matrix to have the velocities to control these motions:

$$ \boldsymbol{\nu}_{e} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} {}^{e}\boldsymbol{T}_{c}\boldsymbol{\nu}_{c} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ w_z \end{bmatrix}. \tag{6} $$

Therefore, $\boldsymbol{\nu}_{e} \in \mathbb{R}^{4}$ is the velocity vector commanded to the quadcopter.

Finally, a condition is defined to determine the achievement of the visual task. We selected a measurement that provides information directly from the image space. The mean square error between the l corresponding points per image pair measured in pixels, denoted by ε, is computed as follows:

$$ \varepsilon = \frac{1}{l}\sum_{j=1}^{l} \|\boldsymbol{p}_j - \boldsymbol{p}_j^{*}\| < T_{\varepsilon}, \tag{7} $$

where $\boldsymbol{p}_j$ and $\boldsymbol{p}_j^{*}$ are the corresponding image points in pixels of the current and target images, respectively. Then, when the error ε reaches a value less than a threshold T_{ε}, the visual task is finished. Figure 2 shows the complete visual servo control scheme of a quadcopter.

3. Hierarchical visual servo control

This section presents a control law that integrates three tasks: visual servo control of the previous section, collision avoidance, and visibility (maintenance of features in the field of view). The three tasks are defined in terms of visual information without requiring an additional sensor. In this work, we use the strategy defined by Lee et al. (2012) to manage the commitment of active tasks (also deactivation of tasks), which consists in defining a strict hierarchy among them (see Figure 3).

In the following subsections, each of the aforementioned tasks, and its corresponding individual controller, are detailed as well as their integration in a hierarchical visual servo control for a quadcopter.

3.1. Collision avoidance

The collision avoidance task allows the quadcopter to overcome obstacles in its path. In this work, we consider regular shaped obstacles placed vertically, large enough to stand out in front of the quadcopter. Their dimensions must be uniform, i.e. of constant width, and their two lateral edges must always be visible in the camera's field of view. To avoid a collision, the robot must keep a security distance from the obstacle. It is achieved by maintaining a reference distance between two points in the image plane. In this work, we use one point on each edge of the obstacle ($\boldsymbol{p}_1$ and $\boldsymbol{p}_2$) as shown in Figure 4. Obstacle detection could be done with simple image segmentation (details are given in Section 5).

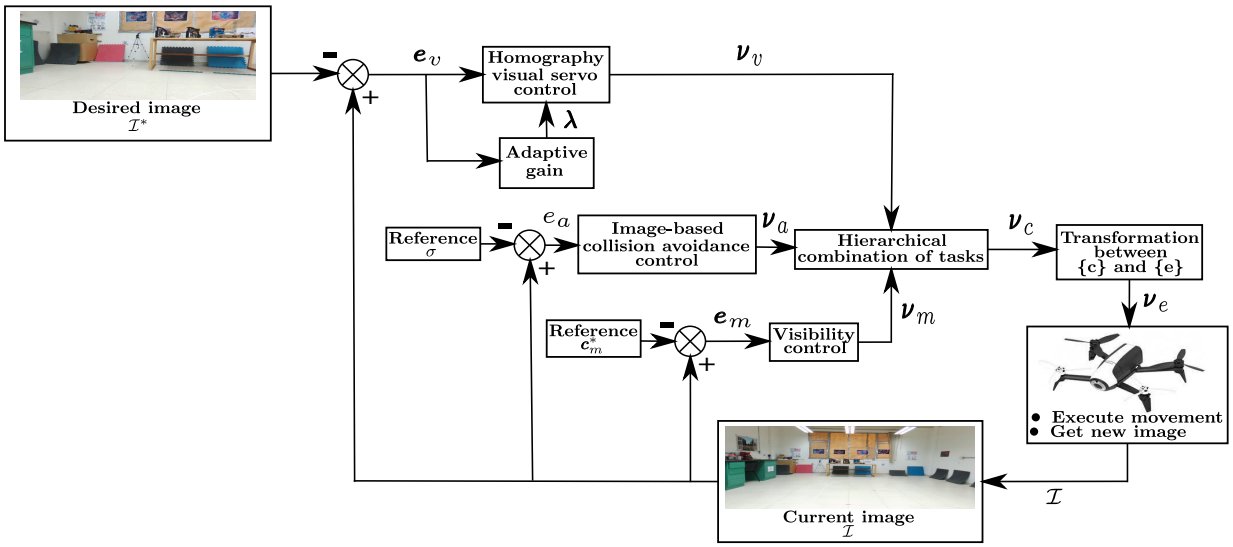


Figure 3: Hierarchical visual servo control scheme for a quadcopter.

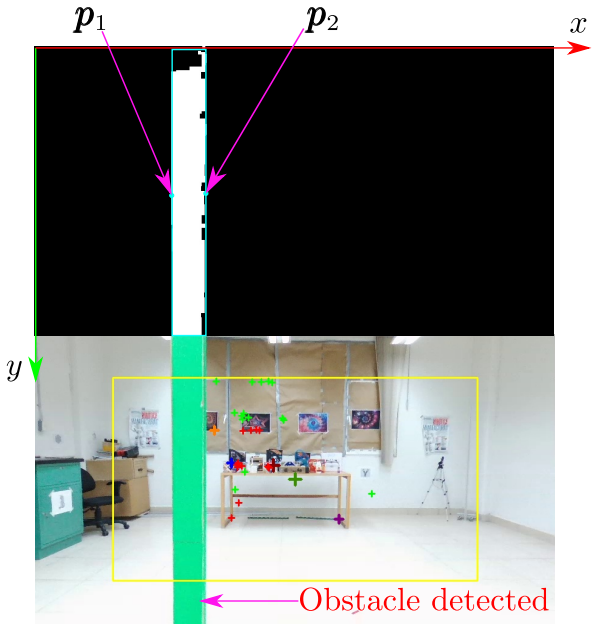


Figure 4: A green column obstacle detected by means of color segmentation. Above binary image with bounding box and below the image showing the detected obstacle.

The collision avoidance error is computed as follows:

$$e_a = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} - \sigma, \qquad (8)$$

where $\sigma \in \mathbb{R}$ is a constant reference distance. Similarly to the homography-based control law, we use normalized coordinates for other tasks, that is, $[x_1 \ y_1 \ 1]^T = \boldsymbol{K}^{-1}\boldsymbol{p}_1$ and $[x_2 \ y_2 \ 1]^T = \boldsymbol{K}^{-1}\boldsymbol{p}_2$. Deriving (8) with respect to time gives the following:

$$\dot{e}_a = \frac{1}{d}\begin{bmatrix}(x_1 - x_2)\\ (y_1 - y_2)\\ (x_2 - x_1)\\ (y_2 - y_1)\end{bmatrix}^T \begin{bmatrix}\dot{x}_1\\ \dot{y}_1\\ \dot{x}_2\\ \dot{y}_2\end{bmatrix},$$

where $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ and

$$\begin{bmatrix}\dot{x}_1\\ \dot{y}_1\\ \dot{x}_2\\ \dot{y}_2\end{bmatrix} = \begin{bmatrix}-\frac{1}{z_1} & 0 & \frac{x_1}{z_1} & x_1y_1 & -(1+x_1^2) & y_1\\ 0 & -\frac{1}{z_1} & \frac{y_1}{z_1} & (1+y_1^2) & -x_1y_1 & -x_1\\ -\frac{1}{z_2} & 0 & \frac{x_2}{z_2} & x_2y_2 & -(1+x_2^2) & y_2\\ 0 & -\frac{1}{z_2} & \frac{y_2}{z_2} & (1+y_2^2) & -x_2y_2 & -x_2\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ v_z\\ w_x\\ w_y\\ w_z\end{bmatrix},$$

where $\{z_1, z_2\} \in \mathbb{R}$ corresponds to the distance between the camera reference frame and the obstacle. Estimated values of z_1 and z_2 can be used, since they are unknown in our monocular framework.

The task function is then composed by the following Jacobians and velocity vector:

$$\boldsymbol{J}_1 = \begin{bmatrix}(x_1 - x_2) & (y_1 - y_2) & (x_2 - x_1) & (y_2 - y_1)\end{bmatrix},$$

$$\boldsymbol{J}_2 = \begin{bmatrix}-\frac{1}{z_1} & 0 & \frac{x_1}{z_1} & x_1y_1 & -(1+x_1^2) & y_1\\ 0 & -\frac{1}{z_1} & \frac{y_1}{z_1} & (1+y_1^2) & -x_1y_1 & -x_1\\ -\frac{1}{z_2} & 0 & \frac{x_2}{z_2} & x_2y_2 & -(1+x_2^2) & y_2\\ 0 & -\frac{1}{z_2} & \frac{y_2}{z_2} & (1+y_2^2) & -x_2y_2 & -x_2\end{bmatrix},$$

$$\boldsymbol{J}_a = \frac{1}{d}\boldsymbol{J}_1\boldsymbol{J}_2,$$

$$\boldsymbol{\nu}_a = \begin{bmatrix}v_x & v_y & v_z & w_x & w_y & w_z\end{bmatrix}^T,$$

such that the collision avoidance controller is:

$$\boldsymbol{\nu}_a = -\lambda_a \hat{\boldsymbol{J}}_a^{+} e_a. \tag{9}$$

The collision avoidance controller must be activated when the distance d is greater than σ, that is, when $e_a > 0$. This controller ensures the convergence to $e_a = 0$, which means that the quadcopter stops when the obstacle size in the image becomes σ. The combination of collision avoidance and visual servoing tasks produces an obstacle avoidance behavior that allows the quadcopter to follow the path towards the target pose.

3.2. Visibility control

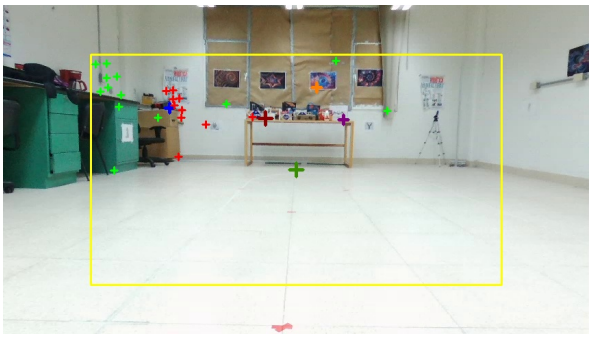


Figure 5: Visibility task.

Figure 5 describes the components of the visibility task. The light-green crosses represent the feature points in the target image, and the light-red crosses represent the feature points in the current image (which is the image shown). The yellow rectangle defines the safety area for the current points (light-red crosses). If any current point leaves the yellow rectangle, the visibility task is triggered. Thus, the objective of the visibility task is to keep all current points in the camera's field of view by maintaining them centered on the image plane. For this purpose, the controller takes the mean of the cloud of current points, represented by the large dark red cross, to the desired mean, represented by the large dark green cross. The visibility task is performed through the control of a virtual point (large dark red cross) calculated from the mean of the points in the current image (light-red crosses) as follows:

$$\boldsymbol{c}_m(t) = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{m}_i, \tag{10}$$

where k is the number of normalized feature points in the current image. The visibility error is calculated as follows:

$$\boldsymbol{e}_m = \boldsymbol{c}_m(t) - \boldsymbol{c}_m^{*}, \tag{11}$$

where $\boldsymbol{c}_m^{*}$ contains the coordinates x and y of the desired mean in the normalized image plane.

The visibility control law in charge of keeping the points in the camera's field of view is calculated as follows:

$$\boldsymbol{\nu}_m = -\lambda_m \hat{\boldsymbol{J}}_m^{+} \boldsymbol{e}_m, \tag{12}$$

where $\hat{\boldsymbol{J}}_m$ is an approximation of the interaction matrix, which stands for:

$$\boldsymbol{J}_m = \begin{bmatrix} -\frac{1}{z} & 0 & \frac{x}{z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{z} & \frac{y}{z} & (1+y^2) & -xy & -x \end{bmatrix},$$

with $z \in \mathbb{R}$ as the distance between the camera reference frame and the scene captured in the target image. Then, an estimated value of z is used.

3.3. Hierarchical control law

We propose using a hierarchical combination of the task functions defined in the previous sections, aiming for the quadcopter to reach a visual target while obstacles are avoided when is needed, and keeping the image points in the camera's field of view. Clearly, the visual servoing task always remains active.

The hierarchy of tasks has been defined as: first, the collision avoidance task, second, the visibility task, and third, the visual task. As a consequence, the control law for computing the velocity vector of the camera is defined as follows:

$$\boldsymbol{\nu}_{c_h} = \boldsymbol{\nu}_a + \boldsymbol{\nu}_m + \boldsymbol{\nu}_v, \tag{13}$$

where $\boldsymbol{\nu}_a$, $\boldsymbol{\nu}_m$ and $\boldsymbol{\nu}_v$ are collision avoidance, visibility maintenance, and visual servo-control components, respectively. They stand for:

$$\begin{aligned} \boldsymbol{\nu}_a &= \hat{\boldsymbol{J}}_a^{+} \dot{e}_a^i, & (14)\\ \boldsymbol{\nu}_m &= (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^{+} (\dot{\boldsymbol{e}}_m^i - \hat{\boldsymbol{J}}_m \boldsymbol{\nu}_a), & (15)\\ \boldsymbol{\nu}_v &= (\hat{\boldsymbol{J}}_v \boldsymbol{N}_m)^{+} (\dot{\boldsymbol{e}}_v^i - \hat{\boldsymbol{J}}_v (\boldsymbol{\nu}_a + \boldsymbol{\nu}_m)). & (16) \end{aligned}$$

The corresponding projectors to the null space of the higher-priority tasks are as follows:

$$\begin{aligned} \boldsymbol{N}_a &= \boldsymbol{I}_{6\times 6} - \hat{\boldsymbol{J}}_a^{+} \hat{\boldsymbol{J}}_a, \\ \boldsymbol{N}_m &= \boldsymbol{N}_a - (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^{+} (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a). \end{aligned}$$

The intermediate values of the combined task functions are given by (Lee et al. (2012)):

$$\begin{aligned}\dot{e}_a^i =& -h_a(t)\lambda_a e_a + (1-h_a(t))\hat{\boldsymbol{J}}_a(-h_m(t)\lambda_m\hat{\boldsymbol{J}}_m^+\boldsymbol{e}_m \\ & - h_v(t)\lambda_v\hat{\boldsymbol{J}}_v^+\boldsymbol{e}_v),\end{aligned} \tag{17}$$

$$\begin{aligned}\dot{\boldsymbol{e}}_m^i =& -h_m(t)\lambda_m\boldsymbol{e}_m + (1-h_m(t))\hat{\boldsymbol{J}}_m(-h_a(t)\lambda_a\hat{\boldsymbol{J}}_a^+e_a \\ & - h_v(t)\lambda_v\hat{\boldsymbol{J}}_v^+\boldsymbol{e}_v),\end{aligned} \tag{18}$$

$$\begin{aligned}\dot{\boldsymbol{e}}_v^i =& -h_v(t)\lambda_v\boldsymbol{e}_v + (1-h_v(t))\hat{\boldsymbol{J}}_v(-h_a(t)\lambda_a\hat{\boldsymbol{J}}_a^+e_a \\ & - h_m(t)\lambda_m\hat{\boldsymbol{J}}_m^+\boldsymbol{e}_m),\end{aligned} \tag{19}$$

where $h_a(t)$, $h_m(t)$, and $h_v(t)$ are time-dependent functions in the closed interval $[0,1]$. They are calculated as follows:

$$h(t) = \begin{cases} \frac{1}{2}\big(1-\cos\big(\frac{\pi(t-t_0)}{t_f-t_0}\big)\big), & t_o \le t \le t_f, \\ 1, & t > t_f, \end{cases} \tag{20}$$

where t_0 is the initial time and t_f is the final time. $t_f - t_0$ is the duration of the task activation. The deactivation of tasks is done by computing $1-h(t)$, and the time that each task is active may be different. The collision avoidance task remains active as long as the distance d is greater than σ, that is, when $e_a > 0$. The visibility task remains active as long as any of the current points are outside the safety area, and the visual task remains always active.

The following theorem analyzes the convergence of the hierarchical control law with intermediate values to combine tasks as in (13).

Theorem 1. *Consider the three-task stack: collision avoidance, visibility, and visual servo control, which can be represented by the following extended kinematics:*

$$\dot{\boldsymbol{e}}' = \begin{bmatrix}\boldsymbol{J}_a \\ \boldsymbol{J}_m \\ \boldsymbol{J}_v\end{bmatrix}\boldsymbol{\nu}_{c_h}, \tag{21}$$

where $\boldsymbol{e}' = \begin{bmatrix}e_a & \boldsymbol{e}_m^T & \boldsymbol{e}_v^T\end{bmatrix}^T$*, with* e_a*,* $\boldsymbol{e}_m$ *and* $\boldsymbol{e}_v$ *given by* (8)*,* (11)*, and* (2) *respectively. The control law* (13) *ensures asymptotic convergence to the origin* $\boldsymbol{e}' = \boldsymbol{0}$*.*

Proof. A candidate Lyapunov function considering the three tasks with intermediate values (13) is:

$$V(\boldsymbol{e}') = \frac{1}{2}\boldsymbol{e}'^T\boldsymbol{e}'. \tag{22}$$

The derivative of (22) with respect to time is as follows:

$$\dot{V}(\boldsymbol{e}') = \boldsymbol{e}'^T\dot{\boldsymbol{e}}' = \boldsymbol{e}'^T\begin{bmatrix}\boldsymbol{J}_a \\ \boldsymbol{J}_m \\ \boldsymbol{J}_v\end{bmatrix}\boldsymbol{\nu}_{c_h}. \tag{23}$$

By substituting (13) into (23) and organizing the terms in matrix form, we obtain the following:

$$\dot{V}(\boldsymbol{e}') = -\boldsymbol{e}'^T\boldsymbol{M}\boldsymbol{e}', \tag{24}$$

where

$$\boldsymbol{M} = \begin{bmatrix}\boldsymbol{M}_a & \boldsymbol{M}_b & \boldsymbol{M}_c \\ \boldsymbol{M}_d & \boldsymbol{M}_e & \boldsymbol{M}_f \\ \boldsymbol{M}_g & \boldsymbol{M}_h & \boldsymbol{M}_i\end{bmatrix}. \tag{25}$$

The mathematical development to obtain the sub-matrices of the matrix $\boldsymbol{M}$ can be found in Appendix A. Using the following properties of the null space projectors defined in Obregón-Flores et al. (2021), we get:

$$\begin{aligned}\boldsymbol{J}_a(\hat{\boldsymbol{J}}_m\boldsymbol{N}_a)^+ &\approx \boldsymbol{0}, \\ \boldsymbol{J}_a(\hat{\boldsymbol{J}}_v\boldsymbol{N}_m)^+ &\approx \boldsymbol{0}, \\ \boldsymbol{J}_m(\hat{\boldsymbol{J}}_m\boldsymbol{N}_a)^+ &\approx \boldsymbol{I}, \\ \boldsymbol{J}_m(\hat{\boldsymbol{J}}_v\boldsymbol{N}_m)^+ &\approx \boldsymbol{0}, \\ \boldsymbol{J}_v(\hat{\boldsymbol{J}}_v\boldsymbol{N}_m)^+ &\approx \boldsymbol{I},\end{aligned} \tag{26}$$

and by considering $\hat{\boldsymbol{J}}_a \approx \boldsymbol{J}_a$, $\hat{\boldsymbol{J}}_m \approx \boldsymbol{J}_m$, $\hat{\boldsymbol{J}}_v \approx \boldsymbol{J}_v$, and $\boldsymbol{J}_a\hat{\boldsymbol{J}}_a^+ \approx 1$, we get the following:

$$\begin{aligned}\boldsymbol{M}_a &= h_a(t)\lambda_a, \\ \boldsymbol{M}_b &= (1-h_a(t))h_m(t)\lambda_m\hat{\boldsymbol{J}}_a\hat{\boldsymbol{J}}_m^+, \\ \boldsymbol{M}_c &= (1-h_a(t))h_v(t)\lambda_v\hat{\boldsymbol{J}}_a\hat{\boldsymbol{J}}_v^+, \\ \boldsymbol{M}_d &= (1-h_m(t))h_a(t)\lambda_a\hat{\boldsymbol{J}}_m\hat{\boldsymbol{J}}_a^+, \\ \boldsymbol{M}_e &= h_m(t)\lambda_m, \\ \boldsymbol{M}_f &= (1-h_m(t))h_v(t)\lambda_v\hat{\boldsymbol{J}}_m\hat{\boldsymbol{J}}_v^+, \\ \boldsymbol{M}_g &= (1-h_v(t))h_a(t)\lambda_a\hat{\boldsymbol{J}}_v\hat{\boldsymbol{J}}_a^+, \\ \boldsymbol{M}_h &= (1-h_v(t))h_m(t)\lambda_m\hat{\boldsymbol{J}}_v\hat{\boldsymbol{J}}_m^+, \\ \boldsymbol{M}_i &= h_v(t)\lambda_v.\end{aligned} \tag{27}$$

On the one hand, when $h_a = 1$, $h_m = 1$ and $h_v = 1$, the matrix $\boldsymbol{M}$ becomes a diagonal matrix with eigenvalues that depend on $\lambda_a > 0$, $\lambda_m > 0$ and $\lambda_v > 0$. In that condition, the error $\boldsymbol{e}'$ converges to the origin asymptotically.

On the other hand, the continuity during the transition of hierarchical tasks is defined in Lee

et al. (2012). In this case, it is worth noting that the visual servoing task is active all the time, and consequently $h_v = 1, \forall t$. Then, the submatrices $\boldsymbol{M}_g = \boldsymbol{0}$ and $\boldsymbol{M}_h = \boldsymbol{0}$. This allows us to express the time-derivative of the Lyapunov function as $\dot{V} = -\boldsymbol{e'}^T \boldsymbol{M}_{trg} \boldsymbol{e'} - \boldsymbol{e}_m^T \boldsymbol{M}_d e_a$, where $\boldsymbol{M}_{trg}$ is an upper triangular matrix with the same blocks on the diagonal as the matrix $\boldsymbol{M}$. Thus, the closed-loop system can be seen as a perturbed system subject to a bounded perturbation $\boldsymbol{e}_m^T \boldsymbol{M}_d e_a$. The effect of this perturbation constrains the convergence of the system to a region around the origin $\boldsymbol{e'} = 0$ (Khalil, 2002)(Chapter 9). However, this condition is bounded to a relatively short finite time $t_f - t_0$, and the system eventually resumes its asymptotic convergence condition at the end of the transition. Hence, convergence of the hierarchical control law with smooth transitions is ensured. □

The proposed control law (13) is valid for 6 degrees of freedom, since it is formulated to yield the vector of velocities in $\mathbb{R}^6$, therefore, it might be applied for fully actuated quadrotors as the ones described in Zheng et al. (2020). However, in the case of conventional quadrotors, as the one that we consider, it is not possible to control them in the 3D space including roll-pitch-yaw, and we have to remove the two velocities not required. Then, it is necessary to transform the velocity vector $\boldsymbol{\nu}_{c_h}$ from the camera to the quadcopter reference frame and apply the selection matrix as in the equation (6). Moreover, given the underactuation of the quadcopter considered, it must initiate flying in hover, i.e., with null pitch and roll angles to start static.

4. Visual path following

The previous sections are dedicated to cope with pose-regulation of a quadcopter considering a single reference image. In this section, we propose to extend the visual servo-controllers for visual path following, where the desired path is represented by a set of images previously taken by the robot in a teach-by-showing fashion. In that case, the quadcopter is asked to solve a sequence of visual servoing tasks, which implies moving from one target image to the next. It is then assumed that the visual path is given and that the robot starts near the first target image of the path.

4.1. Visual path following using hierarchical visual servo control

Besides that the quadcopter has to follow a visual path, collision avoidance and visibility tasks can be activated if needed. When a visual servo-control is used to solve a visual path following problem, it is important to take into account the sudden change of control objectives for different reference images. In other words, when the quadcopter completes a visual servoing task, just at the moment when the next target image is given (from $\mathcal{I}_k^*$ to $\mathcal{I}_{k+1}^*$), the visual servoing error $\boldsymbol{e}$ given in (2) changes abruptly. The consequence is that discontinuous velocities are generated and undesired oscillatory movements of the quadcopter appear. To avoid this issue, we propose to introduce a smooth transition function $h_t(t)$ in the control scheme (13), so that the camera velocity vector is computed as:

$$\boldsymbol{\nu}_{c_{t_1}} = h_t(t) \boldsymbol{\nu}_{c_h}, \tag{28}$$

where $h_t(t)$ is a transition function, which is calculated according to Delfin et al. (2014) as follows:

$$h_t(t) = \begin{cases} h_0 + \frac{1}{2}(1 - h_0)\left(1 - \cos\left(\frac{\pi(t - t_0)}{t_f - t_0}\right)\right), & t_o \leq t \leq t_f, \\ 1, & t > t_f. \end{cases} \tag{29}$$

The function $h_t(t)$ has a minimum value h_0, from which it increases smoothly up to the unity. The value h_0 allows the robot to keep its motion when changing the reference image, while the discontinuities in the velocities are mitigated.

The algorithm 1 describes the sequence of steps by which the quadcopter solves the visual task while avoiding obstacles and keeping the key points in the camera's field of view. The function **matchingKeyPoints** of Algorithm 1 matches the features obtained with the SIFT-type feature detector and descriptor based on Lowe (2004), which are invariant to scale and rotation, so that the matching process is robust to the quadcopter orientation. Then, both the features and the descriptors of the features of both images ($\mathcal{I}_i$, $\mathcal{I}_k^*$) are passed to a robust matcher, which makes use of the RANSAC algorithm (Fischler and Bolles, 1981). The matched key points are ordered to obtain three points, which maximize the area of a triangle. These points are used to build a virtual plane, needed to estimate the generic homography matrix $\boldsymbol{H}$ as proposed by Malis and Chaumette (2000). The function **sortPoints** orders the matched keypoints according to their coordinates (x, y). That is, the first and third points have the highest values on the axes x (purple) and

y (orange), respectively (see Figure 5). The second is the one with the smallest value on the x-axis (blue). The other points remain in the same order as they come out of the matcher (red). The function **tracking** tracks paired feature points through an iterative version of the Lucas-Kanade pyramid optical flow method, based on (Bouguet et al., 2001). The function **checkPoints** verifies that all points in the current image $\mathcal{I}$ (red crosses in Figure 5) are within the safety area (yellow rectangle), based on their image coordinates u and v. The function **lowLevelControl** refers to a low-level velocity control, assumed existing in the quadcopter to execute the velocities v_x, v_y, v_z and w_z given by the hierarchical visual servo control scheme. More details will be given in the experimental section. The function **imgChVerif** is in charge of verifying the conditions to perform the target image change. If $k \neq n$ and error $\varepsilon_k < T_\varepsilon$, then $partialTarget = true$, $i = 0$, and $k = k + 1$. If $k = n$ and $\varepsilon_k < T_\varepsilon$, then $partialTarget = true$, $k = k+1$ and the quadcopter is stopped finishing the navigation. It is worth noting that the threshold T_ε allows to control the accuracy to follow the visual path but also the smoothness of the quadcopter motion. A low T_ε yields good accuracy but no constant motion, since the quadcopter reduces its speed when reaching each target image. A high T_ε reduces the accuracy, but yields a smooth motion. Since the ultimate goal is the safe navigation of the quadcopter towards the position associated with the last reference image, the accuracy can be slightly sacrificed in order to get a fluent motion.

Algorithm 1 : **hierarchicalVisualPathFollowing** allows the robot to solve the n visual subtasks to execute the visual path following while solving the obstacle avoidance task and keeping the key points in the camera's field of view.

Require: visual path $\boldsymbol{I}^* = \{\mathcal{I}_1^*, \mathcal{I}_2^*, \mathcal{I}_{n-1}^*, \mathcal{I}_n^*\}$.
Ensure: visual path following.
$i \leftarrow 0$
$k \leftarrow 1$
$t \leftarrow 0$
$h_a(t) \leftarrow 0$
$h_m(t) \leftarrow 0$
$h_v(t) \leftarrow 1$
$partialTarget \leftarrow true$
while $k \leq n$ **do**
  $\mathcal{I}_i \leftarrow$ **getNewImage**
  if $partialTarget = true$ **then**
    $\mathcal{I}_k^* \leftarrow \boldsymbol{I}^*[k]$
    $keyPoints \leftarrow$ **matchingKeyPoints**$(\mathcal{I}_k^*, \mathcal{I}_i)$
    $sortedMat \leftarrow$ **sortPoints**$(keyPoints)$
    $partialTarget \leftarrow false$
    $t \leftarrow 0$
  else
    $sortedMat \leftarrow$ **tracking**$(\mathcal{I}_i, \mathcal{I}_{i-1}, sortedMat)$
  end if
  $\boldsymbol{H}_k \leftarrow$ **computeHomography**$(sortedMat)$
  $\boldsymbol{e}_{v_k} \leftarrow$ **computeVisualTaskError**$(\boldsymbol{H}_k)$ ▷ (2)
  $\lambda\left(\|\boldsymbol{e}_{v_k}\|\right) \leftarrow$ **computeAdaptiveGain**$(\boldsymbol{e}_{v_k})$ ▷ (4)
  $\boldsymbol{\nu}_v \leftarrow$ **visualTaskVelocities**$(\boldsymbol{e}_{v_k})$ ▷ (16)
  $e_a \leftarrow$ **computeAvoidanceTaskError**$(\boldsymbol{p}_1, \boldsymbol{p}_2)$ ▷ (8)
  if $e_a > 0$ **then**
    $h_a(t) \leftarrow$ **activationFunction**(t) ▷ (20)
  else
    $h_a(t) \leftarrow 1 -$ **activationFunction**(t)
  end if
  $\boldsymbol{\nu}_a \leftarrow$ **avoidanceTaskVelocities**(e_a) ▷ (14)
  $\boldsymbol{e}_m \leftarrow$ **computeMeanTaskError**$(\boldsymbol{c}_m(t))$ ▷ (11)
  $outsideBB \leftarrow$ **checkPoints**$(sortedMat)$
  if $outsideBB > 0$ **then**
    $h_m(t) \leftarrow$ **activationFunction**(t) ▷ (20)
  else
    $h_m(t) \leftarrow 1 -$ **activationFunction**(t)
  end if
  $\boldsymbol{\nu}_m \leftarrow$ **meanTaskVelocities**$(\boldsymbol{e}_m)$ ▷ (15)
  $h_t(t) \leftarrow$ **smoothTransitionFunction**(t) ▷ (29)
  if $n > 1$ **then**
    $\boldsymbol{\nu}_c \leftarrow$ **cameraVelocities**$(\boldsymbol{\nu}_a, \boldsymbol{\nu}_m, \boldsymbol{\nu}_v, h_t(t))$ ▷ (28)
  else
    $\boldsymbol{\nu}_{c_h} \leftarrow$ **cameraVelocities**$(\boldsymbol{\nu}_a, \boldsymbol{\nu}_m, \boldsymbol{\nu}_v)$ ▷ (13)
  end if
  $\boldsymbol{\nu}_e \leftarrow$ **robotVelocities**$(\boldsymbol{\nu}_c)$ ▷ (6)
  robotMovement $\leftarrow$ **lowLevelControl**$(\boldsymbol{\nu}_e)$
  $\varepsilon_k \leftarrow$ **meanSquareError**$(sortedMat)$ ▷ (7)
  $i \leftarrow i + 1$
  $k \leftarrow$ **imgChVerif**$(\varepsilon_k, T_\varepsilon, k, n, partialTarget, i)$
  $t \leftarrow t + cycleTime$
end while
return

4.2. Visual path following using visual servo control

In this subsection, we consider that only the visual servoing task is used, the collision avoidance task, and the visibility task will not be present. In that situation, we describe two control schemes for visual path following. The first uses the visual servo control scheme (3), which, due to the reasons described in Section 4.1, a smooth transition function must be included, as follows:

$$\boldsymbol{\nu}_{c_{t_2}} = h_t(t)\boldsymbol{\nu}_c. \tag{30}$$

Given that the visual path contains valuable information about the route that the quadcopter has to follow, we propose a second scheme that uses the concept of feed-forward, which is integrated to the visual servo control (30), as follows:

$$\boldsymbol{\nu}_{e_f} = p\boldsymbol{\nu}_e + (1-p)\boldsymbol{\nu}_p, \tag{31}$$

where $p \in [0, 1]$, is a weight that allows one to determine the percentage contribution of the feed-forward velocity vector $\boldsymbol{\nu}_p$ in the control scheme. $\boldsymbol{\nu}_e$ and $\boldsymbol{\nu}_p$ are calculated using the i-th target image. $\boldsymbol{\nu}_e$ is the velocity vector obtained from the visual servo control law given in (30), and it is transformed to the robot reference frame $\{e\}$ by means of equation (6). $\boldsymbol{\nu}_p$ is the feed-forward velocity vector of

dimension (4×1), which is defined as:

$$\boldsymbol{\nu}_p = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \gamma {}^{e}\boldsymbol{T}_c \, \boldsymbol{\nu}_c, \qquad (32)$$

where $\boldsymbol{\nu}_c$ is calculated between target images ($\mathcal{I}_i^*$ and $\mathcal{I}_{i+1}^*$) using the control law (3). $\gamma \in [0, 1]$, is a scaling factor that allows the values of the components of the feed-forward velocity vector $\boldsymbol{\nu}_p$ to be adjusted, since the magnitude of the velocities calculated from visual errors, in general, does not have an adequate scale. The feed-forward velocities vectors $\boldsymbol{\nu}_p$ are calculated only one time at the beginning, as can be seen in Algorithm 2, lines 2-15.

The algorithm 2 presents the procedure for implementing the two visual path-following strategies, with and without the feed-forward terms. Unlike Algorithm 1 the hierarchical control is not used, i.e., there is no collision avoidance or visibility task, only the visual task is used. By including the feed-forward term, the quadcopter performs a more continuous motion during the visual path following, that is, the quadcopter does not reduce its speed every time it reaches the pose associated with each target image, as it does during navigation when the feed-forward term is not included. The **imgChVerif** function is in charge of verifying the conditions to perform the target image change. If feed-forward is not used, the strategy becomes similar to that described at the end of Section 4.1. For the strategy with feed-forward, the logic in the **imgChVerif** function is as follows: if $k \neq n$ and if $\varepsilon_k < T_\varepsilon$ or ε_k has an increasing behavior in the last five iterations, then $pTarg = true$, $k = k+1$, $i = 0$, $vVec = vVec + 1$ and the ***mnSqErVec*** data are deleted. If $k = n$ and $\varepsilon_k < T_\varepsilon$, then $k = k + 1$ and the robot stops its motion, finishing the navigation.

Notice that, under no disturbance conditions, it is expected that the feed-forward velocities $\boldsymbol{\nu}_p$, applied in open-loop on the corresponding segment of reference image, will achieve a motion close to the reference visual path, but no correction can be realized. For this reason, the applicability of the control scheme with feed-forward term as in (31) is for situations where the quadcopter has not to perform obstacle avoidance tasks. It would be possible to formulate a hybrid switching scheme where the feed-forward component is used whenever there are no obstacles; however, discontinuous velocities would be generated, which is not desirable.

Algorithm 2 : **visualPathFollowing** allows the robot to solve the n visual sub-tasks to execute the visual path following.

Require: visual path $\boldsymbol{I}^* = \{\mathcal{I}_1^*, \mathcal{I}_2^*, \mathcal{I}_{n-1}^*, \mathcal{I}_n^*\}$, $feedForward$.
Ensure: visual path following.
1: $k \leftarrow 2$
2: **if** $feedForward = true$ **then**
3: **while** $k \leq n$ **do**
4: $\mathcal{I}_k^* \leftarrow \boldsymbol{I}^*[k]$
5: $\mathcal{I}_{k-1}^* \leftarrow \boldsymbol{I}^*[k-1]$
6: $keyPoints \leftarrow$ **matchingKeyPoints**$(\mathcal{I}_k^*, \mathcal{I}_{k-1}^*)$
7: $sortedMat \leftarrow$ **sortPoints**$(keyPoints)$
8: $\boldsymbol{H}_k \leftarrow$ **computeHomography**$(sortedMat)$
9: $\boldsymbol{e}_k \leftarrow$ **computeSubtaskError**$(\boldsymbol{H}_k)$ ▷ (2)
10: $\boldsymbol{\nu}_c \leftarrow$ **homographyBasedControl**$(\boldsymbol{e}_k)$ ▷ (3)
11: $\boldsymbol{\nu}_p \leftarrow$ **feedForwardVelocities**$(\boldsymbol{\nu}_c)$ ▷ (32)
12: $\boldsymbol{V}_p \leftarrow \boldsymbol{\nu}_p^T$
13: $k \leftarrow k + 1$
14: **end while**
15: **end if**
16: $i \leftarrow 0$
17: $k \leftarrow 1$
18: $t \leftarrow 0$
19: $vVec \leftarrow 1$
20: $pTarg \leftarrow true$
21: **while** $k \leq n$ **do**
22: $\mathcal{I}_i \leftarrow$ **getNewImage**
23: **if** $pTarg = true$ **then**
24: $\mathcal{I}_k^* \leftarrow \boldsymbol{I}^*[k]$
25: $keyPoints \leftarrow$ **matchingKeyPoints**$(\mathcal{I}_k^*, \mathcal{I}_i)$
26: $sortedMat \leftarrow$ **sortPoints**$(keyPoints)$
27: $pTarg \leftarrow false$
28: $t \leftarrow 0$
29: **else**
30: $sortedMat \leftarrow$ **tracking**$(\mathcal{I}_i, \mathcal{I}_{i-1}, sortedMat)$
31: **end if**
32: $\boldsymbol{H}_k \leftarrow$ **computeHomography**$(sortedMat)$
33: $\boldsymbol{e}_k \leftarrow$ **computeVisualTaskError**$(\boldsymbol{H}_k)$ ▷ (2)
34: $\lambda(\|\boldsymbol{e}_k\|) \leftarrow$ **computeAdaptiveGain**$(\boldsymbol{e}_k)$ ▷ (4)
35: $\boldsymbol{\nu}_c =$ **homographyBasedControl**$(\boldsymbol{e}_k)$ ▷ (3)
36: $h_t(t) \leftarrow$ **smoothTransitionFunction**(t) ▷ (29)
37: $\boldsymbol{\nu}_{c_{t_2}} \leftarrow$ **cameraVelocities**$(\boldsymbol{\nu}_c, h_t(t))$ ▷ (30)
38: **if** $feedForward = true$ **then**
39: **for** $j \leftarrow 1, 4$ **do**
40: $\boldsymbol{\nu}_p(j) \leftarrow \boldsymbol{V}_p(vVec, j)$
41: **end for**
42: $\boldsymbol{\nu}_e \leftarrow$ **visualVelocities**$(\boldsymbol{\nu}_{c_{t_2}})$ ▷ (6)
43: $\boldsymbol{\nu}_{e_f} \leftarrow$ **robotVelocities**$(\boldsymbol{\nu}_e, \boldsymbol{\nu}_p)$ ▷ (31)
44: **robotMovement** $\leftarrow$ **lowLevelControl**$(\boldsymbol{\nu}_{e_f})$
45: **else**
46: $\boldsymbol{\nu}_e \leftarrow$ **robotVelocities**$(\boldsymbol{\nu}_{c_{t_2}})$ ▷ (6)
47: **robotMovement** $\leftarrow$ **lowLevelControl**$(\boldsymbol{\nu}_e)$
48: **end if**
49: $\varepsilon_k \leftarrow$ **meanSquareError**$(sortedMat)$ ▷ (7)
50: ***mnSqErVec*** $\leftarrow \varepsilon_k$
51: $i \leftarrow i + 1$
52: $k \leftarrow$ **imgChVerif**$(\varepsilon_k, T_\varepsilon, \boldsymbol{mnSqErVec}, k, n, pTarg, i, vVec)$
53: $t \leftarrow t + cycleTime$
54: **end while**
return

It is worth noting that some aspects must be taken into account to have a proper performance of the proposed approach, either for Algorithm 1 or 2: the yaw angle in the initial condition must be adequate to share visual information between the current image and the first reference image. The environment must be sufficiently textured and with

constant lighting conditions to detect image points. The visual path assumed to be given, previously learned, must be adequate in terms of consecutive reference images sharing enough visual information, and a minimum number of image points can be matched. Reference images must be captured with the quadcopter in hover and the approach is able to deal with a type of obstacle as columns, as detailed in Section 3.1.

5. Experimental results

This section is dedicated to evaluating the proposed vision-based control scheme. In Section 5.1 an experiment is shown to validate the visual servo control scheme of Section 3. In Section 5.2 an experiment is presented to validate the visual path following strategy of Section 4. In Section 5.3, two experiments are presented to compare the results obtained using the control law of Section 2 and the visual path following strategy of Section 4, first without feed-forward velocity terms, and then with feed-forward velocity components. We use a *Parrot Bebop 2* drone that already has the required low-level velocity controller. Although there is no official information from *Parrot* about the internal control of *Bebop 2*, some existing works (Giernacki et al., 2020; Pinto et al., 2020) have studied its capabilities and described a typical structure of velocity control for this quadcopter, with an inner control loop for pitch and roll attitude angles, whose references are given by an outer position/velocity control loop. Yaw angle and altitude are decoupled in independent controllers. Such approach allows to control the quadcopter in position or velocity and generate its references from a high level control as a trajectory tracking controller, or a visual servo control as in our case. The effectiveness of the high level control clearly depends on the performance of the internal control; however, it is enough for the visual servo control as long as a positioning control has been tested in the quadcopter and it was able to get close to a desired position.

Vision-based control computations were implemented using C / C++ on a *Hewlett Packard* laptop computer with 12 GB of RAM, Intel®Core ™i5-7200U CPU @ 2.50GHz x 4, and *Ubuntu 18.04.4 LTS* 64-bit operating system. For communication between the quadcopter and the computer, the ViSP library was used, which has functions to control Bebop 2 (Marchand et al., 2005). The velocity commands (equation (6)) are sent to the quadcopter through the wireless connection (Wi-Fi) established between the robot and the laptop. For sending velocity commands, the *drone.setVelocity* function implemented in the *vpRobotBebop2* class of the ViSP library is used. This function sends the velocity commands to the quadcopter and does not receive any response message or feedback information, thus, no additional sensors are used to verify whether the commanded velocities are achieved. However, in addition to the results on trajectory tracking of Giernacki et al. (2020); Pinto et al. (2020), the Bebop 2 has been used efficiently by commanding velocities for formation control (Trujillo et al., 2021; Vargas et al., 2022) using the low-level internal controller. Low-level control is critical for visual servo control performance in terms of overshooting, but does not compromise its stability.

Although we implemented the proposed approach using a *Bebop 2* quadcopter, it is remarkable that the approach is applicable to other quadcopters. First, since the approach is formulated at kinematics level, no dynamic model of the quadcopter is required, and the size, mass, and other physical features of the platform are not crucial. In the image processing part, the method relies on the detection of at least 16 image points, to have more than the 8 points necessary to compute the generic homography (Malis and Chaumette, 2000). It is important the distribution of the points in the image to have a well conditioned homography computation, which is provided by point detectors such as SIFT or ORB. One aspect to look after is the speed of the quadcopter induced by the convergence time as given by the control gains. The speed of the quadcopter must be adequate to allow the point tracker based on optical flow (Bouguet et al., 2001), to work properly. Then, it is expected that different control gains are needed whether other quadcopter is used.

5.1. Pose-regulation with obstacle avoidance

Figure 6 shows the scenario for the first experiment, which consists of a forward (1.5 m) and vertical downward (0.5 m) displacement of the quadcopter while avoiding a 0.05 m wide × 0.02 m deep rectangular obstacle. The robot must be initially positioned so that the camera's field of view allows it to share sufficient visual information with the target image $\mathcal{I}^*$. In this case, the process for the computation of the control law is found in Algorithm 1 for $n = 1$.

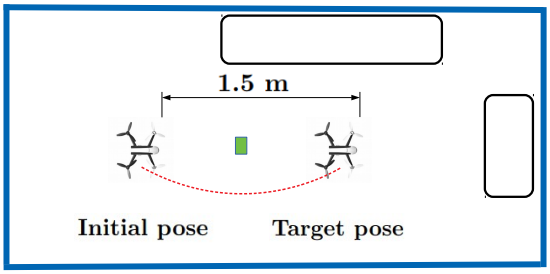


Figure 6: Top view of the scenario for the pose-regulation experiment with obstacle avoidance.

The following parameters were used for the experiment, $\sigma = 0.05$ was taken as a reference value for the calculation of the collision avoidance task error (8) and a gain $\lambda_a = 0.01$ (9). The yellow rectangle in Figure 7 was used as a security area, the large dark green cross is $\boldsymbol{c}_m^* = (428, 240)$, the center of the image (11), and the gain of the control law $\lambda_m = 0.2$ (12). For the visual task, an adaptive gain $\lambda(\|\boldsymbol{e}\|)$ was used with parameters $\lambda_0 = 2.8$, $\lambda_\infty = 0.1$ and $\lambda_0^{'} = 5.0$ (4), for the gain matrix of the intermediate visual error (19), $\lambda_{v_x} = \lambda_{v_z} = \lambda(\|\boldsymbol{e}\|)$, $\lambda_{v_y} = \lambda(\|\boldsymbol{e}\|) + 2.5$ and $\lambda_w = 0.5\lambda(\|\boldsymbol{e}\|)$. The value of the mean square error that determines the completion of the task was $T_\varepsilon = 6$ *pixels* (7). In the experiments carried out, it was observed that controlling ε, allows the quadcopter to reach the desired pose. In turn, the components of the task error $\boldsymbol{e}$ in equation (2) converge to values close to zero.

The controller performance depends on the selection and tuning of some parameters. In particular, λ_∞ regulates the velocity of the quadcopter at the beginning of the visual task when the error $\boldsymbol{e}$ is usually large. λ_0 allows the robot to generate sufficient velocities to complete the visual task. $\lambda_0^{'}$ handles the transition velocity from λ_∞ to λ_0 when $\|\boldsymbol{e}\|$ approaches zero. The distance between the quadcopter and the obstacle to activate the collision avoidance task relies in σ. $\boldsymbol{c}_m^*$ determines the placement in the image of the virtual point $\boldsymbol{c}_m(t)$, which is used to keep the current points cloud. $t_f - t_0$ stands for the duration of the activation and deactivation time of tasks in $h(t)$. Note that for the collision avoidance task, it should not be too long, so the task is activated early, and the robot can respond and perform the avoidance. In the visibility task, it can be a little longer because it does not compromise the integrity of the quadcopter. Also, $t_f - t_0$ determines the transition time in $h_t(t)$. A short transition time makes the smoothing of velocity discontinuities ineffective, while a long transition time causes that the velocity vector, obtained from the controller, to not be fully considered until a long time. As a by-product, the duration of task execution becomes excessive.

In Figure 7, a sequence of images can be seen with the quadcopter performing pose regulation and obstacle avoidance tasks. The images correspond to the internal and external views of the robot in the columns to the left and right, respectively. The red crosses represent the feature points in the current image and the green crosses are the corresponding points in the target image. The yellow rectangle is the safety area for the visibility task. Since the focus of this work is visual servo control and task combination using only 2D information as measurements, the obstacle detection process was implemented by a color (green) segmentation technique. This method imposes a bounding box on the segmented obstacle, and it selects the points $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ on each edge of the bounding box around the middle height of the image. Obstacle detection could be done more generally using artificial intelligence techniques as in Liu et al. (2019). Thus, if a more sophisticated detection algorithm is used and nonuniform obstacles are encountered, the important aspect is to have a bounding box enclosing the obstacle to define the maximum width of the obstacle in the image similarly as done in our results.

Figure 8 depicts the cycle time for the pose regulation experiment with obstacle avoidance. It can be observed that in 96 percent of the iterations, the cycle time value was 40 ms and the average working frequency of the controller was 24.7 Hz.

Figure 9 shows the evolution in time of the mean square error ε from equation (7) and the error of the visual task denoted by $\boldsymbol{e}$ equation (2). It can be seen how the mean square error converges to a value of $\varepsilon < T_\varepsilon$, with $T_\varepsilon = 6$ *pixels* (red dotted horizontal line) at $t = 12$ *s* (black dotted vertical line) and the values of the task error $\boldsymbol{e}$ converge to values close to zero. After that time, ε remains close to the reference, and the visual errors remain close to zero until the end of the experiment at $t = 20$ *s*.

Figure 10 displays the velocity commands sent to the quadcopter (6), where the velocities in the camera reference frame were calculated with equation (13). The cyan rectangle represents the time interval during which the quadcopter performed obstacle avoidance. It can be observed that in the time in-

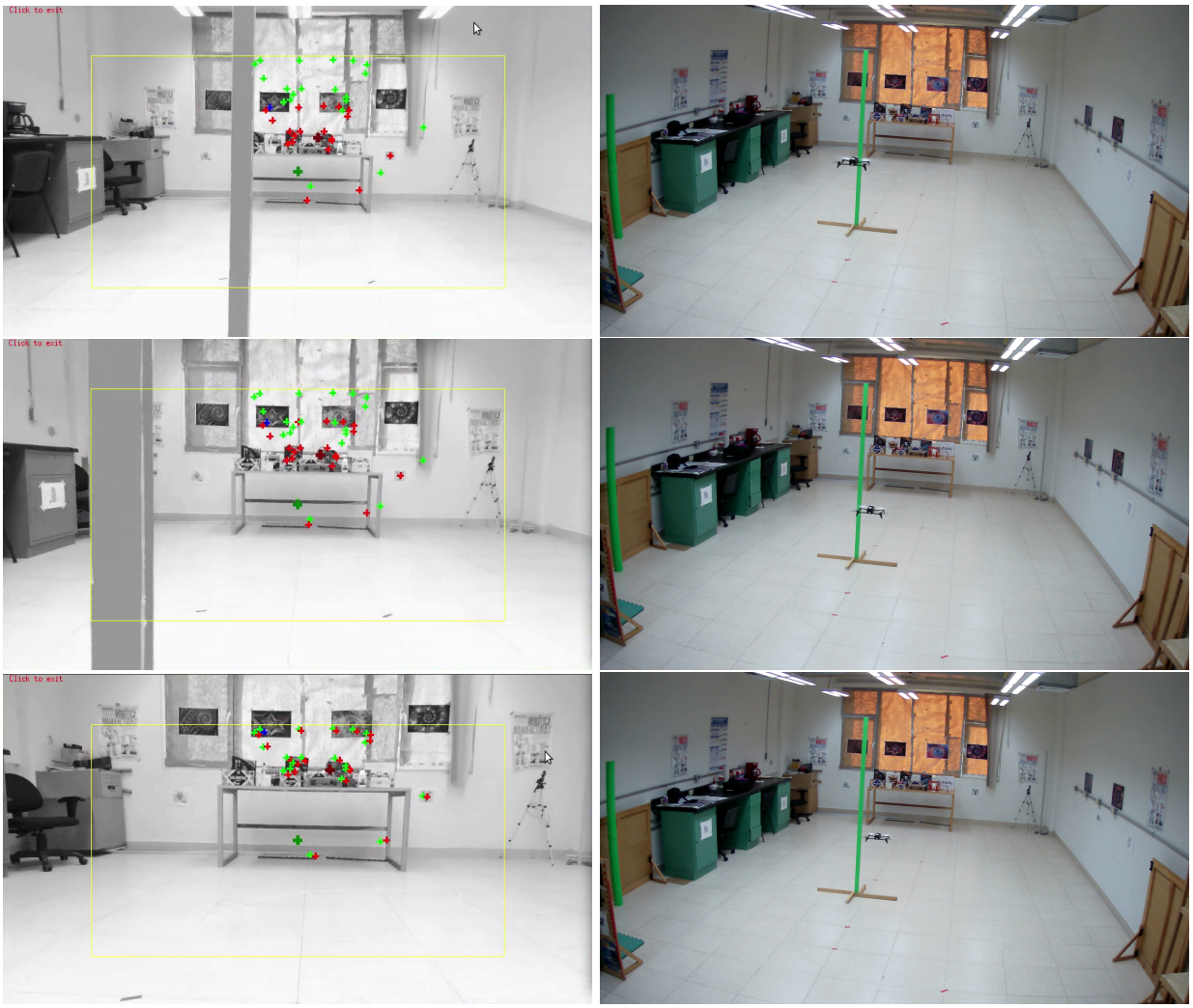


Figure 7: Internal (left side column) and external (right side column) view of the quadcopter performing the pose-regulation experiment with obstacle avoidance. The first row corresponds to the moment when the robot starts the visual task, the second row corresponds to the moment in which it avoids the obstacle, and the last row to the end of the visual task.

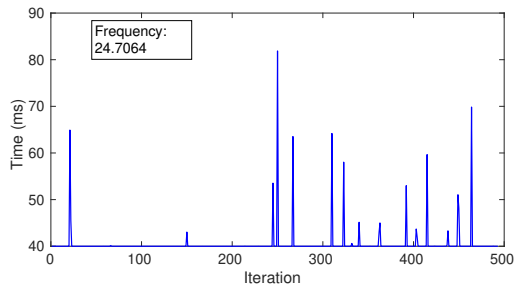
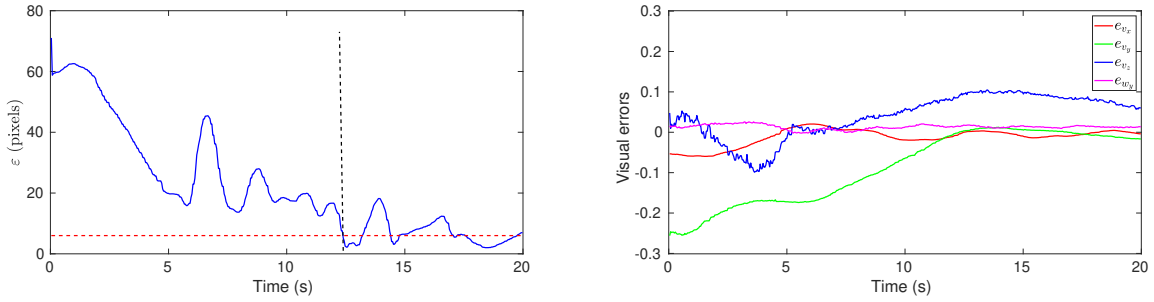


Figure 8: Cycle time for the pose-regulation experiment with obstacle avoidance.

terval from 0 to 5 seconds, the forward velocity v_x decreases smoothly, and the lateral velocity v_y increases in the same way. At instant of time $t = 5\ s$, the obstacle leaves the camera field of view, generating a change in the behavior of the quadcopter velocities, increasing its forward velocity and decreasing its lateral velocity. The behavior of the velocity v_z is because the robot starts the task to a height of 1.0 m, and the target image is at a reference height of 0.50 m, so the quadcopter must move downwards until it reaches a height of 0.5 m. The second row of Figure 10 illustrates the activation function for the collision avoidance task, and the third row the activation function for the visibility



(a) Mean square error.

(b) Visual task error.

Figure 9: Errors of the visual task for the pose-regulation experiment with obstacle avoidance.

task. It can be seen that the evasion task is activated at $t = 0$ s, increases its value smoothly until it reaches its maximum value of 1 at $t = 3$ s, remains active until $t = 5$ s, and begins to deactivate until $t = 8$ s. The visibility task was not activated, so the activation function h_m remains at zero.

Figure 11 depicts the errors of the collision avoidance task and the visibility task. It can be observed that the error of the avoidance task grows in the time interval from 0 to 5 seconds, and at $t = 5$ s the obstacle leaves the field of view of the camera, therefore the current value can no longer be measured, remaining this at the reference value $\sigma = 0.05$. The error of the visibility task is observed to behave increasingly because the target mean is the center of the image (large dark green cross) and the current mean is the centroid of the current key points cloud (large dark red cross), the task is activated as soon as one of the current points (red crosses) leaves the safety area (yellow rectangle); therefore, as long as there is no red point outside the safety area, the task is not activated and there is no correction of the robot pose for this task (see Figure 5). The objective of the visibility task is not to bring the error completely to zero, but through this task to avoid the key points going out of the field of view of the camera. It can be seen in Figure 7 that during the execution of all the visual task the current points remain within the safety area, therefore the task is not activated.

5.2. Visual path following with obstacle avoidance

This subsection presents the extension of hierarchical visual servo control of a quadcopter to follow a visual path while it avoids obstacles if needed. Figure 12 shows the scenario for this second experiment, which consists of following a visual path made up of four target images, and the robot must evade two obstacles. The point detection, the matching process, and control law are the same as those used in the first experiment, computed according to Algorithm 1 for $n = 4$. Unlike the experiment in Section 5.1, this controller runs at an average frequency of 55 Hz. The difference is due to the removal of a 40 ms delay in the algorithm, which was intended to synchronize the controller with the sampling rate of the Bebop 2 camera, which in video mode works at 30 fps. Eliminating the delay increased the average frequency and improved the performance of the quadcopter. From the starting point to the first image, the robot must make a frontal displacement of 1.5 m at a height of 1 m. On the way, it must avoid a cylindrical obstacle of 0.11 m in diameter. Once the pose associated with the first target image $\mathcal{I}_1^*$ is reached, the target image is changed to image $\mathcal{I}_2^*$, in this case, a forward displacement of 0.5 m at a height of 1.0 m is performed. Once the pose associated with the target image $\mathcal{I}_2^*$ is reached using the condition $\varepsilon < T_\varepsilon$ with $T_\varepsilon = 12$ $pixels$, the change from the target image to the target image $\mathcal{I}_3^*$ is made. In that section of the path, the quadcopter must travel 1.5 m forward at a height of 1.0 m while avoiding a rectangular obstacle of 0.05 m wide x 0.02 m deep. Once the pose associated with the target image $\mathcal{I}_3^*$ is reached, the target image is changed to the last image of the visual path $\mathcal{I}_4^*$; this time, the robot must move 0.5 m laterally towards the left side, 1.0 m towards the front, descend to a height of 0.5 m while oriented 30 degrees measured from the orientation of the previous target pose. The transition function $h(t)$ has a duration of 1.6 s and $h_0 = 0.2$.

In Figure 13, a sequence of images of the quadcopter performing the visual path following with obstacle avoidance can be observed. These cor-

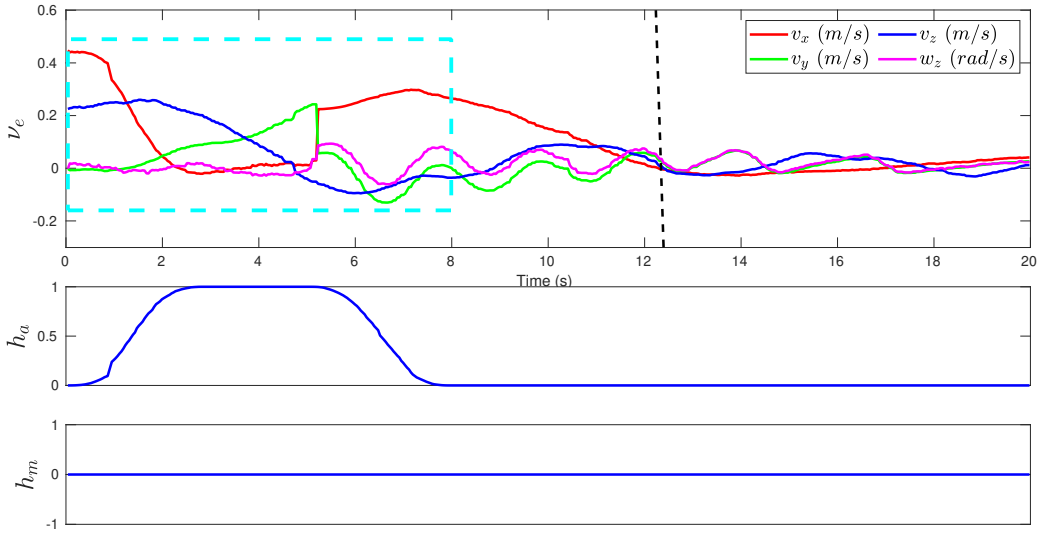
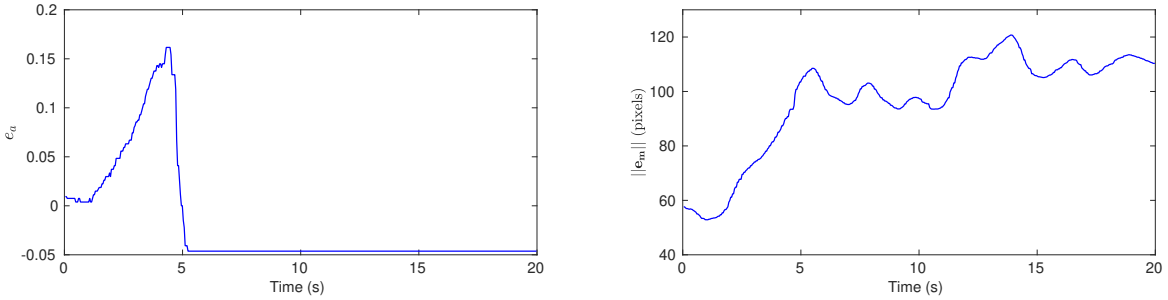


Figure 10: Velocities for pose-regulation experiment with obstacle avoidance. The first row corresponds to velocities, where the cyan rectangle represents the time interval in which obstacle avoidance was performed. The second row represents the activation function of the collision avoidance task, and the third is the activation function of the visibility task.



(a) Collision avoidance task error.

(b) Visibility task error.

Figure 11: Errors for the pose-regulation experiment with obstacle avoidance.

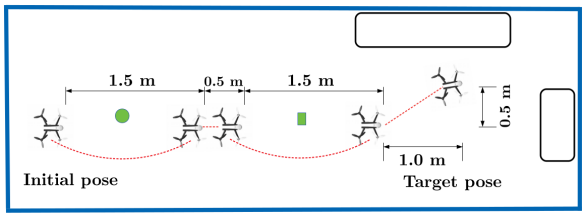


Figure 12: Top view of the scenario for the visual path following experiment with obstacle avoidance.

respond to the internal and external views of the quadcopter. In the picture on the left side in the last row, right at the end of the visual task, it can be seen that the current points (red crosses) are very close to the desired points (green crosses).

Figure 14 displays the mean square error (ε) and the task error for the visual task ($\boldsymbol{e}$). The black vertical lines dotted in the mean square error image represent the time instants in which the changes of the target image were made ($\varepsilon < T_\varepsilon$ with $T_\varepsilon = 12\ pixels$ for the intermediate images $\mathcal{I}_1^*$, $\mathcal{I}_2^*$, and $\mathcal{I}_3^*$), it can be seen that just at these moments ($t = 12\ s$, $t = 15\ s$, and $t = 26\ s$) the error has abrupt changes in its value. Finally, the quadcopter ends the task when the error reaches a value $\varepsilon < T_\varepsilon$, with $T_\varepsilon = 7\ pixels$. In the graph of error $\boldsymbol{e}$, it can be seen

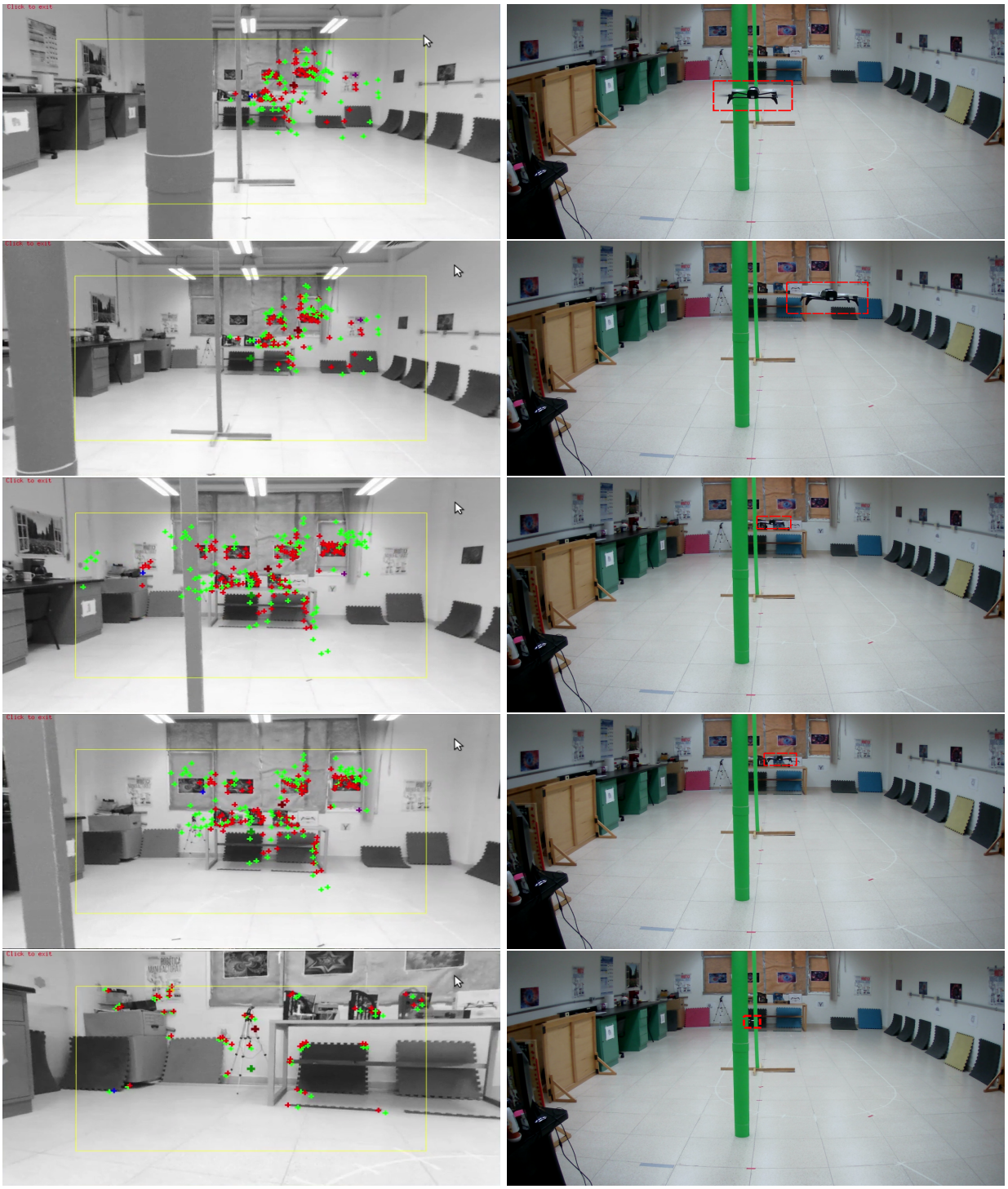
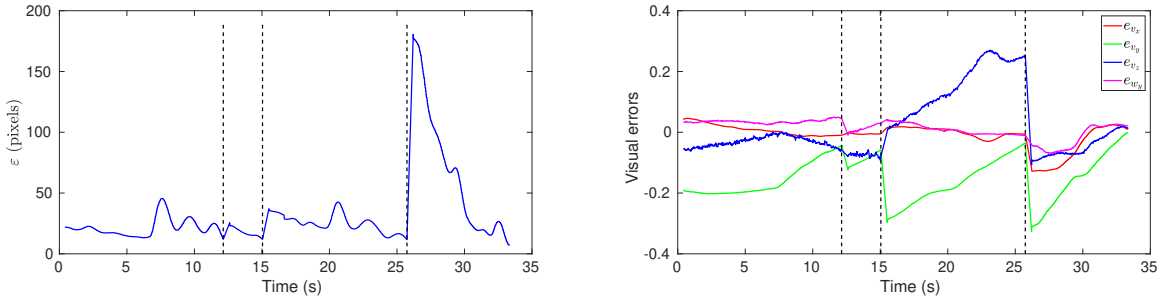


Figure 13: Internal (left side column) and external (right side column) view of the quadcopter performing the visual path following experiment with obstacle avoidance. The first row corresponds to the moment when the robot starts the first visual subtask, the second row to the moment in which it avoids the first obstacle, the third row to the start of the third visual subtask (position between the two obstacles), the fourth corresponds to the moment in which it performs the evasion of the second obstacle, and the fifth row corresponds to the moment in which the quadcopter finishes the fourth and last visual subtask. The red rectangle shows the position of the robot.



(a) Mean square error.

(b) Visual task error.

Figure 14: Errors of the visual task for the visual path following experiment with obstacle avoidance. The black dotted vertical lines represent the time instants at which the target image changes were made.

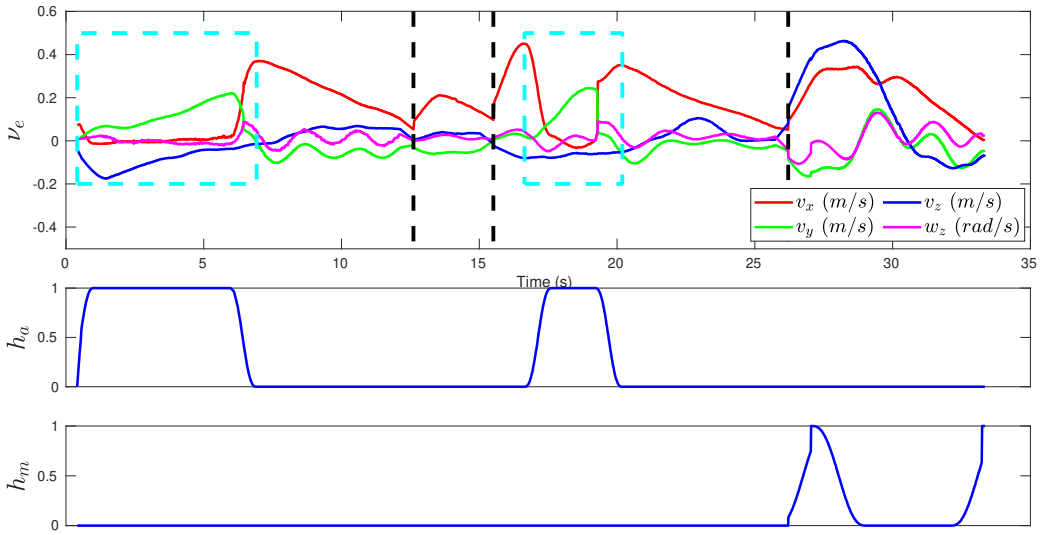
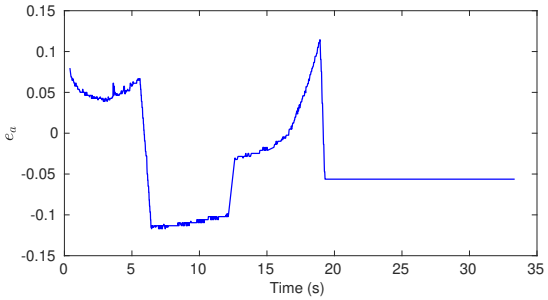


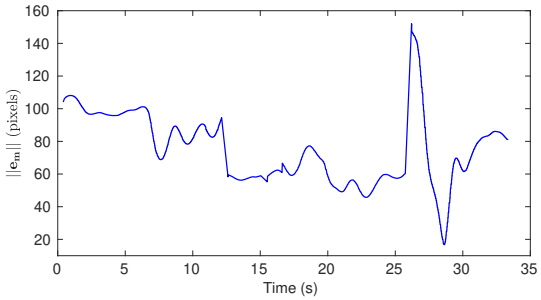
Figure 15: Velocities for the visual path following experiment with obstacle avoidance. The first row corresponds to velocities, where the cyan rectangles represent the time intervals in which obstacle avoidances were performed, and the black dotted vertical lines represent the time instants at which the target image changes were made. The second row represents the activation function of the collision avoidance task, and the third is the activation function of the visibility task.

that, as for error ε, there are abrupt changes in the instants of time in which the target image changes were made. At the end of the task, it is observed that each of the error components $\boldsymbol{e}$ converges to values close to zero.

Figure 15 depicts the velocity commands; the black dotted vertical lines represent the time instants in which the changes of the target image were made; it can be seen that after making the changes of the target image, the velocity transitions are made smoothly, due to the action of the transition function $h_t(t)$. The cyan-colored rectangles represent the time intervals in which the robot executed the obstacle avoidance task, it can be observed that at the beginning and end of the execution of the avoidance task, the changes in velocities are made smoothly by the action of the activation function $h_a(t)$. Finally, it can be observed that the velocities converge to values close to zero. The second row shows the activation function of the collision avoidance task. It can be seen that activation and deactivation are performed smoothly for the two time intervals in which the quadcopter avoided the two obstacles. The third row shows the activation



(a) Collision avoidance task error.



(b) Visibility task error.

Figure 16: Errors for the visual path following experiment with obstacle avoidance.

function of the visibility task. In this one, smooth activation and deactivation can be observed. It was active in the interval from $t = 26\ s$ to $t = 29\ s$ and was activated again at $t = 32\ s$, remaining active because the robot reached the pose associated with the target image $\mathcal{I}_4^*$.

Figure 16a illustrates the error of the evasion task. It is observed that during the interval from $t = 0\ s$ to $t = 5\ s$, the error initially has a decreasing behavior, then increases to finally fall to the reference value for the first obstacle of $\sigma = 0.13$ because the obstacle leaves the field of view of the camera. Then, in the interval from $t = 11\ s$ to $t = 20\ s$, the error has an increasing behavior to finally take a constant value of $\sigma = 0.06$, which corresponds to the reference value for the second obstacle. The change in the value of σ is due to the fact that the distance between the robot and the obstacle at the time of task activation and during task execution is intended to be approximately equal for both cases. Figure 16b shows the error of the visibility task, the reasons for the increasing error behavior are the same as described in the first experiment. The peak that stands out at $t = 26\ s$ is due to the last change of the target image. To achieve the pose associated with this target image, the quadcopter must move in the four degrees of freedom, which generates a high visibility task error at the beginning of the last visual sub-task (11), activating for a while the visibility control.

5.3. Visual path following with feed-forward

This subsection presents the application of the proposed visual servo control for a longer cyclic path, where the robot must return to the same initial position. In this case, no obstacles are present along the path, therefore, it is possible to use the control scheme that includes a feed-forward component.

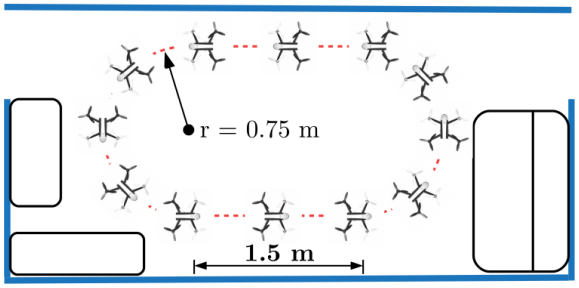
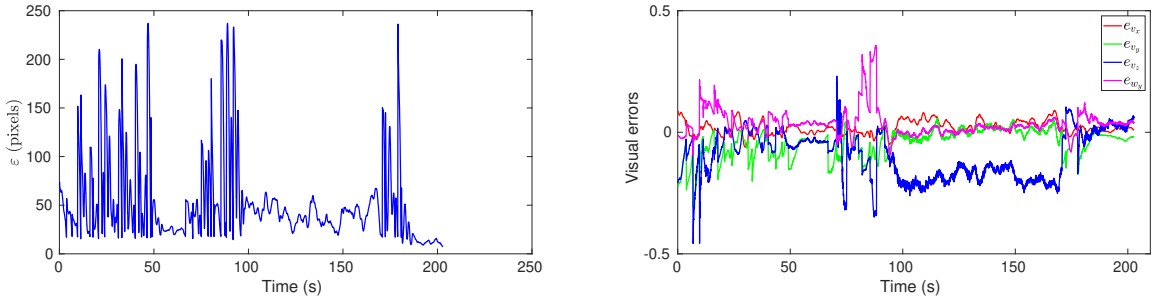


Figure 17: Top view of the scenario for the experiment of visual path following with feed-forward.

Figure 17 displays the scenario for the visual path following the experiment, first considering the *control law without feed-forward* velocity components. The path has a total travel distance of 7.7 m and is composed of 30 images, of which 8 correspond to frontal linear displacements (50 cm distance between them) and 22 correspond to rotation (15°) plus translation movements (approximately 20 cm), all acquired at 1 m height. The adaptive gain parameters $\lambda(\|\boldsymbol{e}\|)$ used were $\lambda_0 = 2.1$, $\lambda_\infty = 1.0$ and $\lambda_0^{'} = 5$. The values of the gain matrix are: $\lambda_{v_x} = \lambda_{v_z} = \lambda(\|\boldsymbol{e}\|)$, $\lambda_{v_y} = \lambda(\|\boldsymbol{e}\|) + 0.1$ and $\lambda_{w_x} = \lambda_{w_y} = \lambda_{w_z} = 0.5\lambda(\|\boldsymbol{e}\|)$. The threshold T_ε, which determines the condition to complete each of the visual sub-tasks, according to the mean square error ε, from equation (7), was defined as $T_\varepsilon = 18\ pixels$, for the target images $\{\mathcal{I}_1^*, ..., \mathcal{I}_{29}^*\}$, and $T_\varepsilon = 7\ pixels$, for the last target image $\mathcal{I}_{30}^*$. The first value of $T_\varepsilon = 18\ pixels$, allows the robot to navigate nimbly and accurately between intermediate target images. The second $T_\varepsilon = 7\ pixels$, allows the robot more accurately reaching the final



(a) Mean square error.

(b) Visual task error.

Figure 18: Errors of the visual task for the experiment of visual path following without feed-forward.

pose, according to the target image $\mathcal{I}^*_{30}$. The process in this experiment is described in Algorithm 2; in this case, the detection of features was performed with a corner detector based on (Shi et al., 1994). The controller was run at an average frequency of 24 Hz. As explained in Section 4.2, in this kind of experimental setup the visual servo control law of equation (30) is applicable and not the hierarchical control law.

Figure 18a depicts the mean square error ε of equation (7). It can be observed that the error converges to a value of $\varepsilon < 18$ for the target images $\mathcal{I}^*_1, ..., \mathcal{I}^*_{29}$ and $\varepsilon < 7$ for the last target image $\mathcal{I}^*_{30}$. Figure 18b presents the error of task $\boldsymbol{e}$ of the equations (2). It can be seen how errors go from low to high values smoothly due to the effects of the transition function $h_t(t)$, but those oscillatory effects will be reflected in the velocities, as will be presented below.

In Figure 19, the velocity vector $\boldsymbol{\nu}_e$ of the equation (6) is shown, which for its calculation is used the velocity vector $\boldsymbol{\nu}_{c_{t_2}}$ of equation (30). It can be seen that the rotational velocity w_z has three time intervals during which it has sudden changes in value $10 \leq t \leq 50$, $70 \leq t \leq 95$ and $175 \leq t \leq 190$. These variations are due to the changes of target image, and they are smoothed on the rise through the transition function $h_t(t)$, and they are reduced by the controller, which generates these sudden changes. In the first two intervals, the quadcopter moves on the curve's path, and in the last interval, it reaches the final target. The variations in the v_x and v_y components are due to the need to correct the robot's position on the (xy) plane while performing the navigation task. Changes in the v_z component are due to the quadcopter generates variations in its flight altitude. At the end of the task ($t = 200$ s), all components of $\boldsymbol{\nu}_e$ converge to a value close to zero.

Next, we present the results of an experiment using the *control law with feed-forward* velocity components. The scenario for this experiment is the same as the experiment of visual path following without feed-forward velocity terms (see Figure 17). The visual memory in this case is slightly different, composed of 35 images, with some more images in the linear segment of the path, all images acquired at 1 m height.

The adaptive gain parameters $\lambda(\|\boldsymbol{e}\|)$ used were $\lambda_0 = 1.5$, $\lambda_\infty = 0.8$ and $\lambda_0^{'} = 5$. The values of the gain matrix were: $\lambda_{v_x} = \lambda_{v_z} = \lambda(\|\boldsymbol{e}\|)$, $\lambda_{v_y} = \lambda(\|\boldsymbol{e}\|) + 2$ y $\lambda_{w_x} = \lambda_{w_y} = \lambda_{w_z} = 0.5\lambda(\|\boldsymbol{e}\|)$. The threshold T_ε for the intermediate images and the last image are the same as in the previous experiment. For the computation of the feed-forward velocity vector $\boldsymbol{\nu}_p$ from equation (32), a scaling factor of $\gamma = 0.75$ was used. For the computation of the velocity vector $\boldsymbol{\nu}_{e_f}$ (equation (31)), a weighting of $p = 0.5$ was used. Therefore, the contribution of the feed-forward velocity vector is 50% of the total velocity vector sent to the quadcopter. The process of computing the control law is the same as described in the experiment without feed-forward velocities according to the Algorithm 2. The controller was run at an average frequency of 23 Hz.

Figure 20 shows a sequence of images of the quadcopter performing the visual path following with feed-forward velocities. These correspond to the internal and external views of the robot.

In Figure 21a, the mean square error ε of equation (7) is shown. It can be observed that the error in the time intervals $0 \leq t < 8$ and $25 \leq t < 35$, reaches lower values. During these time intervals, the quadcopter navigates over the two straight seg-

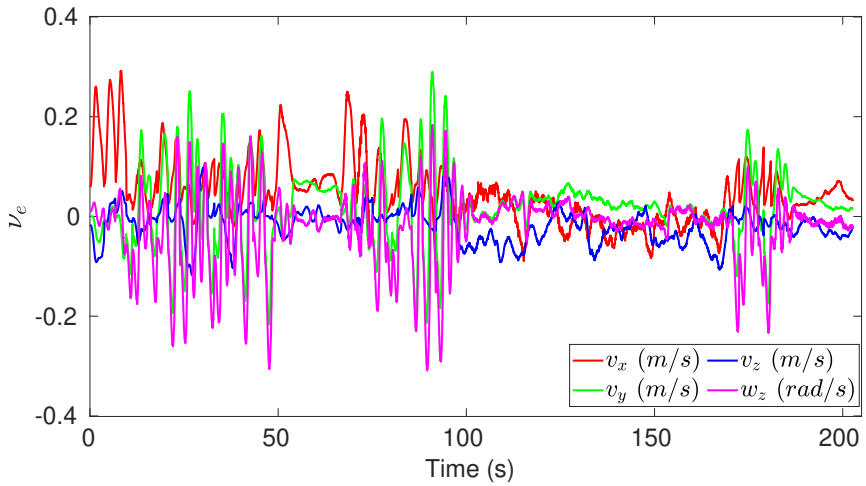


Figure 19: Velocities for experiment of visual path following without feed-forward.

ments of the oval, and therefore, it is easier for the robot to correct the error. In the time intervals $8 \leq t < 25$ and $35 \leq t$, the sudden error changes due to the changes of the target image. During these time intervals, the quadcopter navigates over the two curve segments of the oval. It can be seen that the error does not reduce to the threshold $T_\varepsilon = 18$. In general, the changes of target images were given by the condition of increment of the error ε along the last five iterations, since the path-following is not as accurate as using only the feedback components; however, the quadcopter is able to follow the path and reach the final target faster than in the experiment without feed-forward terms.

In Figure 21b, the error of task $\boldsymbol{e}$ of equations (2) is presented. It is observed how the visual errors at the end converges to values close to zero. In addition, sudden changes can be observed in the e_{v_y}, e_{v_z}, and e_{w_y}, which are due to changes of the target image. The behavior of e_{v_y} is due to the low-level control of the robot that generates small changes in the y axis of the camera when trying to maintain the reference height of 1 m (see Figure 1).

In Figure 22, it is shown the velocity vector $\boldsymbol{\nu}_{e_f}$ of equation (31), which is calculated using the velocity vector $\boldsymbol{\nu}_{c_{t_2}}$ of equation (30) and the feed-forward velocity vector from equation (32), obtained from visual path processing described in Section 4.2. Two types of lines are presented in the figure. Continuous lines represent the components of the vector $\boldsymbol{\nu}_{e_f}$ that was applied to the quadcopter, and dashed lines represent the feed-forward velocities obtained from visual path processing. Notice that the velocity component v_x corresponding to the forward motion of the robot remains on average 0.17 m/s. Therefore, it can be deduced that the quadcopter moves in continuous forward motion, unlike the experiment without feed-forward velocity components. It can be seen that the shape of the dashed line plots is consistent with the shape of the commanded velocities (in a continuous line) that the quadcopter executed. If the quadcopter starts near the position associated with the first target image, it would be expected that these feed-forward velocities alone in open-loop would make the robot move close to the path.

Finally, it is worth noting that for the case of this experiment, the robot traversed a total of 7.71 m in 51.32 s, only 25% of the time taken to perform the same navigation task with the visual servo control scheme without feed-forward velocities terms. Thus, the main advantage of using feed-forward components is that the quadcopter has not reduced its forward velocity for each target image; then the path is traveled faster than using the control law without feed-forward components. An issue is that the feed-forward component is only applicable when there are no obstacles in the path to be followed by the quadcopter, since the feed-forward velocities are computed from the reference visual path without deviations.

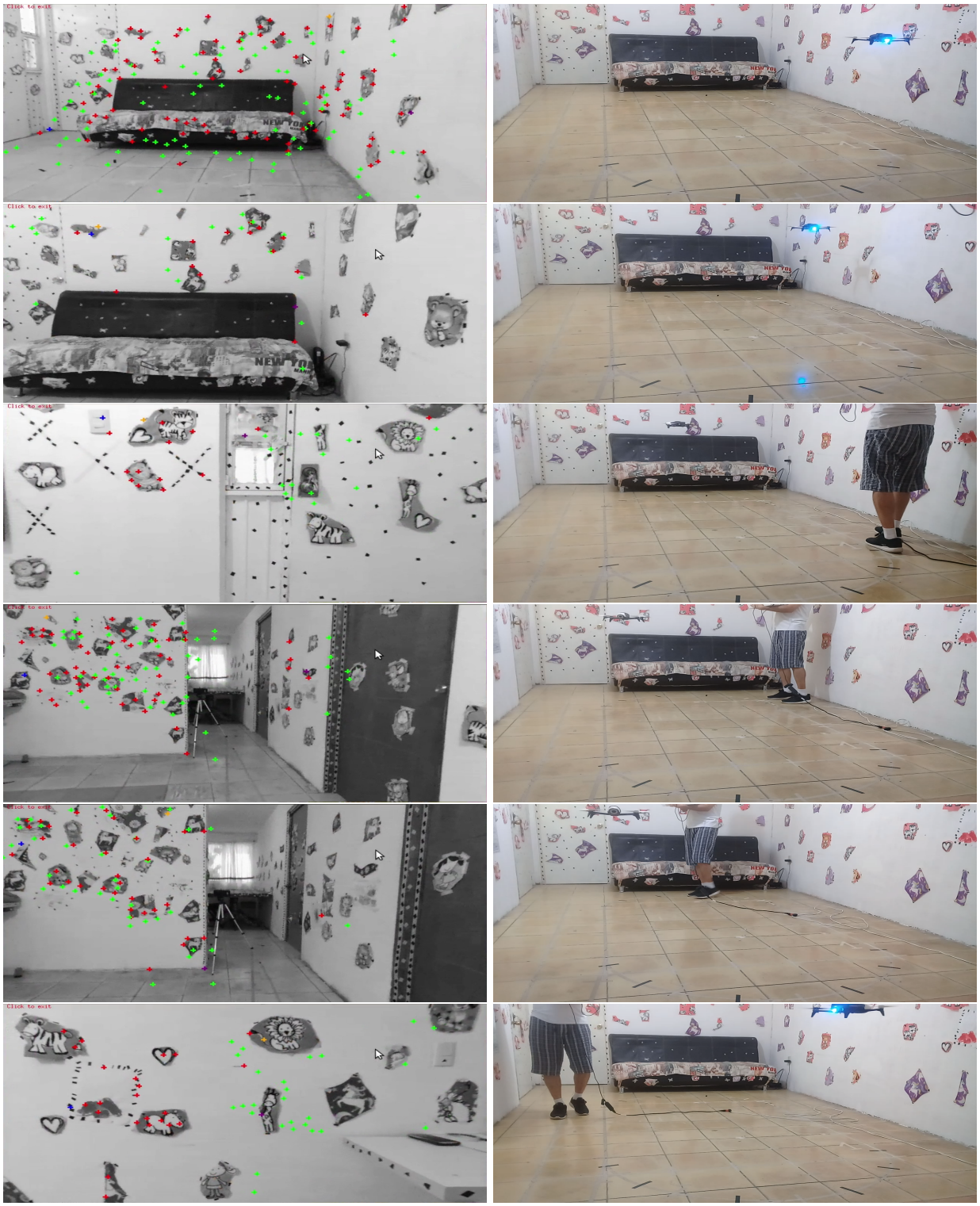
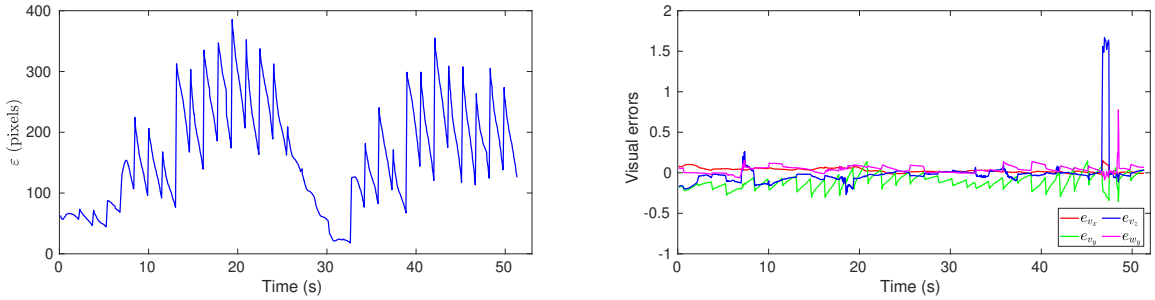


Figure 20: Internal (left side column) and external (right side column) view of the quadcopter performing the visual path following with feed-forward experiment. The first and second rows show the navigation of the first straight line of the circuit, the third row shows the pose of the robot in the middle of the first curve, the fourth and fifth rows show the navigation of the second straight line, and the last row shows the pose of the quadcopter in the middle of the second curve.



(a) Mean square error.

(b) Visual task error.

Figure 21: Errors of the visual task for the experiment of visual path following with feed-forward.

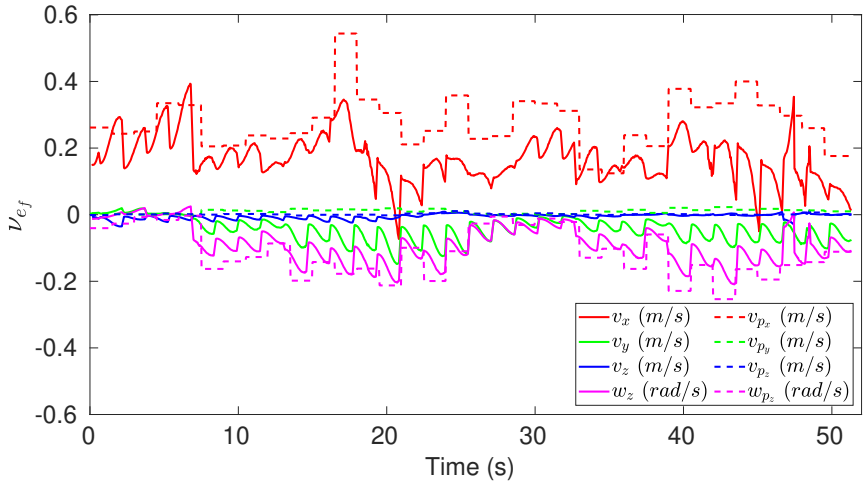


Figure 22: Velocities for experiment of visual path following with feed-forward. The continuous lines represent the components of the vector $\boldsymbol{\nu}_{e_f}$ that was applied to the quadcopter, and the dashed ones represent the feed-forward velocities obtained from the visual path processing.

6. Conclusions

In this paper, we proposed a hierarchical control scheme for quadcopters where the main goal is that the quadcopter follows a visual path, which consists of a sequence of images previously acquired. The hierarchical control is composed of three components, each one dedicated to solve a task in a hierarchy defined as follows: first, a collision avoidance task, second a visibility task (maintenance of point features in the camera's field of view), and finally a visual servoing task. The stability analysis of the closed-loop control law is provided considering the three tasks, which allowed us to verify the asymptotic convergence of the errors to the origin.

The main feature of the proposed control scheme is that it was formulated in the image space, i.e. it is fully 2D; the solution for each task is only defined in terms of information obtained from a monocular RGB camera. The control scheme is applicable to quadcopters that have a low-level velocity controller able to execute commanded translational velocities in x, y, z and the rotational velocity in yaw. In our case, the effectiveness of the proposed control scheme was validated through experiments using a commercial quadcopter. We have presented experiments for pose-regulation with a single target image and also visual path following with a sequence of target images, in both cases performing obstacle avoidance if needed. Besides, we proposed to include a feed-forward component in the control law, which showed that the quadcopter is able to complete a path in less time than just using a feedback component.

Different from Courbon et al. (2010); Do et al.

(2019), where position-based visual servo controllers are proposed, our image-based controller does not rely on the rotation matrix and the scaled translation vector between pairs of images, which requires a particular estimation process. In Courbon et al. (2010); Do et al. (2019), the quadcopter moves only in the xy plane, i.e., it always navigates at the same altitude. In contrast, the proposed control scheme considers the four degrees of freedom of the quadcopter to perform translational 3D movements with changes in altitude. Moreover, obstacle avoidance is commonly performed by estimating the quadcopter and obstacle locations in 3D space by means of optical flow, IMU and ultrasonic sensors as suggested in Do et al. (2019). The obstacle avoidance task in the proposed control scheme is solved in image space, it is fully 2D and reactive.

As future work, it is planned to replace the simple segmentation of obstacles as done here with a more general and robust object detection, for instance, based on deep learning algorithms. In addition, to avoid reliance on computer vision algorithms for the detection, matching, and tracking of point features, and enhance the applicability of the approach for low textured scenes, it is planned to use deep learning algorithms to estimate a homography matrix between pairs of images and the rest of the proposed control scheme could be the same as described in this paper.

7. Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

8. Acknowledgment

The first author acknowledges the financial support of the National Council of Science and Technology (CONACyT) under scholarship No. 943100.

Appendix A. Sub-matrices of $\boldsymbol{M}$

Using the properties of equation (26) in the matrix $\boldsymbol{M}$ of equation (25), we get the following:

$$
\begin{aligned}
\boldsymbol{M}_a =& h_a(t)\lambda_a \boldsymbol{J}_a \hat{\boldsymbol{J}}_a^+,\\
\boldsymbol{M}_b =& (1-h_a(t))h_m(t)\lambda_m \boldsymbol{J}_a \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_m^+,\\
\boldsymbol{M}_c =& (1-h_a(t))h_v(t)\lambda_v \boldsymbol{J}_a \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_v^+,\\
\boldsymbol{M}_d =& h_a(t)\lambda_a \boldsymbol{J}_m \hat{\boldsymbol{J}}_a^+ - h_a(t)\lambda_a \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+\\
&+ (1-h_m(t))h_a(t)\lambda_a \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+,\\
\boldsymbol{M}_e =& h_m(t)\lambda_m + (1-h_a(t))h_m(t)\lambda_m \boldsymbol{J}_m \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_m^+\\
&- (1-h_a(t))h_m(t)\lambda_m \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_m^+,\\
\boldsymbol{M}_f =& (1-h_m(t))h_v(t)\lambda_v \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_v^+\\
&+ (1-h_a(t))h_v(t)\lambda_v \boldsymbol{J}_m \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_v^+\\
&- (1-h_a(t))h_v(t)\lambda_v \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_v^+,\\
\boldsymbol{M}_g =& \Big(h_a(t)\lambda_a \boldsymbol{J}_v \hat{\boldsymbol{J}}_a^+ - h_a(t)\lambda_a \hat{\boldsymbol{J}}_v \hat{\boldsymbol{J}}_a^+\Big)\\
&+ \Big((1-h_m(t))h_a(t)\lambda_a \boldsymbol{J}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+\\
&\quad -(1-h_m(t))h_a(t)\lambda_a \hat{\boldsymbol{J}}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+\Big)\\
&+ \Big(h_a(t)\lambda_a \hat{\boldsymbol{J}}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+\\
&\quad -h_a(t)\lambda_a \boldsymbol{J}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+\Big)\\
&+ (1-h_v(t))h_a(t)\lambda_a \hat{\boldsymbol{J}}_v \hat{\boldsymbol{J}}_a^+,\\
\boldsymbol{M}_h =& \Big((1-h_a(t))h_m(t)\lambda_m \boldsymbol{J}_v \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_m^+\\
&\quad -(1-h_a(t))h_m(t)\lambda_m \hat{\boldsymbol{J}}_v \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_m^+\Big)\\
&+ \Big(h_m(t)\lambda_m \boldsymbol{J}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+\\
&\quad -h_m(t)\lambda_m \hat{\boldsymbol{J}}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+\Big)\\
&+ \Big((1-h_a(t))h_m(t)\lambda_m \hat{\boldsymbol{J}}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_m^+\\
&\quad -(1-h_a(t))h_m(t)\lambda_m \boldsymbol{J}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_m^+\Big)\\
&+ (1-h_v(t))h_m(t)\lambda_m \hat{\boldsymbol{J}}_v \hat{\boldsymbol{J}}_m^+,\\
\boldsymbol{M}_i =& \Big((1-h_a(t))h_v(t)\lambda_v \boldsymbol{J}_v \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_v^+\\
&\quad -(1-h_a(t))h_v(t)\lambda_v \hat{\boldsymbol{J}}_v \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_v^+\Big)\\
&+ \Big((1-h_m(t))h_v(t)\lambda_v \boldsymbol{J}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_v^+\\
&\quad -(1-h_m(t))h_v(t)\lambda_v \hat{\boldsymbol{J}}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_v^+\Big)\\
&+ \Big((1-h_a(t))h_v(t)\lambda_v \hat{\boldsymbol{J}}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_v^+\\
&\quad -(1-h_a(t))h_v(t)\lambda_v \boldsymbol{J}_v (\hat{\boldsymbol{J}}_m \boldsymbol{N}_a)^+ \hat{\boldsymbol{J}}_m \hat{\boldsymbol{J}}_a^+ \hat{\boldsymbol{J}}_a \hat{\boldsymbol{J}}_v^+\Big)\\
&+ h_v(t)\lambda_v.
\end{aligned}
$$

References

Ayub, M.F., Ghawash, F., Shabbir, M.A., Kamran, M., Butt, F.A., 2018. Next generation security and surveillance system using autonomous vehicles, in: Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS), pp. 1–5.

Benhimane, S., Malis, E., 2007. Homography-based 2d visual tracking and servoing. The International Journal of Robotics Research 26, 661–676.

Bouguet, J.Y., et al., 2001. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Intel Corporation 5, 4.

Chaumette, F., Hutchinson, S., 2006. Visual servo control. i. basic approaches. IEEE Robotics & Automation Magazine 13, 82–90.

Chiang, M.L., Tsai, S.H., Huang, C.M., Tao, K.T., 2021. Adaptive visual servoing for obstacle avoidance of micro unmanned aerial vehicle with optical flow and switched system model. Processes 9, 2126.

Courbon, J., Mezouar, Y., Guénard, N., Martinet, P., 2010. Vision-based navigation of unmanned aerial vehicles. Control Engineering Practice 18, 789–799.

Delfin, J., Becerra, H.M., Arechavaleta, G., 2014. Visual path following using a sequence of target images and smooth robot velocities for humanoid navigation, in: 2014 IEEE-RAS International Conference on Humanoid Robots, IEEE. pp. 354–359.

Delfin, J., Becerra, H.M., Arechavaleta, G., 2016. Visual servo walking control for humanoids with finite-time convergence and smooth robot velocities. International Journal of Control 89, 1342–1358.

Do, T., Carrillo, L.C., Roumeliotis, S.I., 2015. Autonomous flights through image-defined paths, International Symposium on Robotics Research (ISRR).

Do, T., Carrillo-Arce, L.C., Roumeliotis, S.I., 2019. High-speed autonomous quadrotor navigation through visual and inertial paths. The International Journal of Robotics Research 38, 486–504.

Fehr, M., Schneider, T., Dymczyk, M., Sturm, J., Siegwart, R., 2018. Visual-inertial teach and repeat for aerial inspection. arXiv preprint arXiv:1803.09650 .

Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24, 381–395.

Gans, N.R., Hu, G., Nagarajan, K., Dixon, W.E., 2011. Keeping multiple moving targets in the field of view of a mobile camera. IEEE Transactions on Robotics 27, 822–828.

Garcia, A., Mattison, E., Ghose, K., 2015. High-speed vision-based autonomous indoor navigation of a quadcopter, in: Int. Conf. on Unmanned Aircraft Systems, pp. 338–347.

Giernacki, W., Kozierski, P., Michalski, J., Retinger, M., Madonski, R., Campoy, P., 2020. Bebop 2 quadrotor as a platform for research and education in robotics and control engineering, in: Int. Conf. on Unmanned Aircraft Systems (ICUAS), IEEE. pp. 1733–1741.

Iacono, M., Sgorbissa, A., 2018. Path following and obstacle avoidance for an autonomous uav using a depth camera. Robotics and Autonomous Systems 106, 38–46.

Jiang, Y., Gao, W., Na, J., Zhang, D., Hämäläinen, T.T., Stojanovic, V., Lewis, F.L., 2022. Value iteration and adaptive optimal output regulation with assured convergence rate. Control Engineering Practice 121, 105042.

Kermorgant, O., Chaumette, F., 2013. Dealing with constraints in sensor-based robot control. IEEE Transactions on Robotics 30, 244–257.

Khalil, H.K., 2002. Nonlinear Systems. Prentice Hall, Upper Saddle River, New Jersey.

Kozák, V., Pivoňka, T., Avgoustinakis, P., Majer, L., Kulich, M., Přeučil, L., Camara, L.G., 2021. Robust visual teach and repeat navigation for unmanned aerial vehicles, in: 2021 European Conference on Mobile Robots (ECMR), pp. 1–7.

Lee, J., Mansard, N., Park, J., 2012. Intermediate desired value approach for task transition of robots in kinematic control. IEEE Transactions on Robotics 28, 1260–1277.

Lim, J., Pyo, S., Kim, N., Lee, J., Lee, J., 2020. Obstacle magnification for 2-d collision and occlusion avoidance of autonomous multirotor aerial vehicles. IEEE/ASME Transactions on Mechatronics 25, 2428–2436.

Lin, H.Y., Peng, X.Z., 2021. Autonomous quadrotor navigation with vision based obstacle avoidance and path planning. IEEE Access 9, 102450–102459.

Lin, J., Zhu, H., Alonso-Mora, J., 2020. Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments, in: IEEE Int. Conf. on Robotics and Automation, pp. 2682–2688.

Liu, J.J., Hou, Q., Cheng, M.M., Feng, J., Jiang, J., 2019. A simple pooling-based design for real-time salient object detection, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 3917–3926.

Liu, N., Shao, X., Li, J., Zhang, W., 2020. Attitude restricted back-stepping anti-disturbance control for vision based quadrotors with visibility constraint. ISA transactions 100, 109–125.

Loianno, G., Brunner, C., McGrath, G., Kumar, V., 2017. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. IEEE Robotics and Automation Letters 2, 404–411.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. International journal of computer vision 60, 91–110.

Malis, E., Chaumette, F., 2000. 2 1/2 d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. International Journal of Computer Vision 37, 79–97.

Marchand, E., Spindler, F., Chaumette, F., 2005. Visp for visual servoing: a generic software platform with a wide class of robot control skills. IEEE Robotics and Automation Magazine 12, 40–52.

Martinez-Carranza, J., Rojas-Perez, L.O., 2022. Warehouse inspection with an autonomous micro air vehicle. Unmanned Systems , 1–14.

Mellinger, D., Kumar, V., 2011. Minimum snap trajectory generation and control for quadrotors, in: IEEE Int. Conf. on Robotics and Automation, IEEE. pp. 2520–2525.

Mercado, D., Castillo, P., Lozano, R., 2018. Sliding mode collision-free navigation for quadrotors using monocular vision. Robotica 36, 1493–1509.

Nguyen, T., Mann, G., Gosine, R., Vardy, A., 2016. Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle. J Intell Robot Syst 84, 217–240.

Nitsche, M., Pessacg, F., Civera, J., 2020. Visual-inertial teach and repeat. Robotics and Autonomous Systems 131, 103577.

Obregón-Flores, J., Arechavaleta, G., Becerra, H.M., Morales-Díaz, A., 2021. Predefined-time robust hierarchical inverse dynamics on torque-controlled redundant manipulators. IEEE Transactions on Robotics 37, 962–978.

Park, J., Kim, Y., 2015. Collision avoidance for quadrotor using stereo vision depth maps. IEEE Transactions on Aerospace and Electronic Systems 51, 3226–3241.

Pinto, A.O., Marciano, H.N., Bacheti, V.P., Moreira, M.S.M., Brandão, A.S., Sarcinelli-Filho, M., 2020. High-level modeling and control of the bebop 2 micro aerial vehicle, in: Int. Conf. on Unmanned Aircraft Systems (ICUAS), IEEE. pp. 939–947.

Shi, J., et al., 1994. Good features to track, in: 1994 Proceedings of IEEE conference on computer vision and pattern recognition, IEEE. pp. 593–600.

Škrinjar, J.P., Škorput, P., Furdić, M., 2018. Application of unmanned aerial vehicles in logistic processes, in: International Conference on New Technologies, Development and Applications, Springer. pp. 359–366.

Song, X., Sun, P., Song, S., Stojanovic, V., 2022. Event-driven nn adaptive fixed-time control for nonlinear systems with guaranteed performance. Journal of the Franklin Institute 359, 4138–4159.

Spindler, F., Gaumerais, G., 2019. Tutorial: Visual-servoing with parrot bebop 2 drone. https://visp-doc.inria.fr/doxygen/visp-daily/tutorial-bebop2-vs.html. 22-05-2020.

Spitzer, A., Michael, N., 2021. Feedback linearization for quadrotors with a learned acceleration error model, in: IEEE Int. Conf. on Robotics and Automation, pp. 6042–6048.

Tang, Z., Cunha, R., Cabecinhas, D., Hamel, T., Silvestre, C., 2021. Quadrotor going through a window and landing: An image-based visual servo control approach. Control Engineering Practice 112, 104827.

Trujillo, M.A., Becerra, H.M., Gomez-Gutierrez, D., Ruiz-Leon, J., Ramirez-Treviño, 2021. Hierarchical task-based formation control and collision avoidance of UAVs in finite time. European Journal of Control 60, 48–64.

Vargas, S., Becerra, H.M., Hayet, J.B., 2022. Mpc-based distributed formation control of multiple quadcopters with obstacle avoidance and connectivity maintenance. Control Engineering Practice 121, 105054.

Warren, M., Greeff, M., Patel, B., Collier, J., Schoellig, A.P., Barfoot, T.D., 2019. There's no place like home: Visual teach and repeat for emergency return of multirotor uavs during gps failure. IEEE Robotics and Automation Letters 4, 161–168.

Yang, X., Chen, J., Dang, Y., Luo, H., Tang, Y., Liao, C., Chen, P., Cheng, K.T., 2021. Fast depth prediction and obstacle avoidance on a monocular drone using probabilistic convolutional neural network. IEEE Transactions on Intelligent Transportation Systems 22, 156–167.

Zheng, D., Wang, H., Wang, J., Zhang, X., Chen, W., 2019. Toward visibility guaranteed visual servoing control of quadrotor uavs. IEEE/ASME Transactions on Mechatronics 24, 1087–1095.

Zheng, P., Tan, X., Kocer, B.B., Yang, E., Kovac, M., 2020. Tiltdrone: A fully-actuated tilting quadrotor platform. IEEE Robotics and Automation Letters 5, 6845–6852.