Quadrotor Stabilization using Visual Servoing

Bruno Oliveira[1]

Instituto Superior Técnico - IST,
Instituto Superior Técnico, Technical University of Lisbon, Portugal

[1]bruno.de.oliveira@ist.utl.pt

***Abstract*—The objective of this work is to develop a position stabilization controller based on Visual Servoing alternatives for an existing quadrotor platform.**

We first introduce the quadrotor concept along with a mathematical model for its motion and the essential theory concerning computer vision. The quadrotor characteristics are then presented, including some tools developed in order to improve its capabilities, and the platform-specific parameters are estimated in order to create a SimuLink® model for numerical simulations.

Some of the available Visual Tracking alternatives are then described. The estimation alternatives of velocity and relative motion are also investigated. A validation of the visual tracking alternatives and the homography reconstruction is performed, and a comparative study is developed. Possible PBVS methods are tested in SimuLink® in order to chose the most suitable alternative.

Finally, a realtime implementation of the chosen PBVS scheme installed onboard the quadrotor is described, followed by a camera calibration and an assessment of estimation results.

Keywords*—*Quadrotor, Visual Servoing, Homography, PBVS, Optical Flow.

I. Introduction

In the recent years, quadrotors have been attracting a great interest in applications such as aerial surveillance and observation, and because they appear as ideal platforms to test new control and estimation algorithms. The angular stabilization of quadrotors was first considered and numerous alternative solutions have been proposed; for translation, GPS is usually the main sensor but, when accuracy is needed or when GPS is not available, vision algorithms are a good approach for the horizontal control of a quadrotor.

Visual servoing or *visual servo control* usually refer to the use of visual data to control an autonomous vehicle (not necessarily aerial), being the visual data acquired from a camera mounted directly on the robot or fixed on the surroundings of the workspace. The first case is usually known as *eye-in-hand* case while the former is known as *eye-to-hand* case. In the eye-in-hand case, the movement of the robot implies the movement of the camera, while in the eye-to-hand case the camera is fixed and the mapping from the camera frame onto the robot control frame must be performed.

There is a broad range of control schemes for visual servoing. Most of these strategies can be subdivided into two different approaches: the Image-Based Visual Servoing (IBVS) and the Position-Based Visual Servoing (PBVS).

IBVS techniques assume that a task can be fully specified directly in terms of geometric features of an image, defining an error function that depends on those features, while PBVS techniques consider a camera as an extra sensor, from which a vehicle pose can be estimated and fed into a regular control loop. A comparative study between these two approaches is presented in [1], and an overview of the most usual visual servoing techniques can be found in [2].

Although most of the visual servo control laws in literature are based on non-linear control and Lyapunov stability theorems, the control loops in this text are derived from the Linear Quadratic Regulator (LQR) control. Regular controllers are used, and an image camera is used solely as an extra sensor capable of outputting position and attitude measures: for this reason, the visual servoing techniques presented in this work are included in the PBVS strategies.

This work starts by introducing a quadrotor mathematical model in section II, followed by some computer vision theory in Section III. Afterwards, the QuavIST platform is presented in Section IV. Some visual tracking techniques are introduced and compared in Section V, and some simulations are described in Section VI. Finally, a realtime implementation is described in Section VII and the conclusions that arose during this work are presented in Section VIII.

II. Quadrotor Motion Modeling

In order to simplify the formal analysis and development of the rotorcraft model, several assumptions are made. The quadrotor is assumed as a rigid body, with four perfectly similar propellers and motors. The motion range is short enough for the Earth rotation and translation to be neglected. The torques created from aerodynamic forces are neglected. There is a imaging camera which is considered to be rigidly attached to the quadrotor, with its geometric center and center of projection centered in the quadrotor center of mass.

A. Reference frames

The movement of any general aircraft is usually described using two right-hand frames: a reference North-East-Down (NED) frame, and a second Aircraft-Body-Centered (ABC) frame. The first is set on the ground, centered in $\mathcal{O}$ and pointing towards North, East and Down. The ABC frame is set on $\mathcal{O}_c$, the center of mass of the quadrotor, aligned with its reference arms. Assuming short-range maneuvers, the Earth rotation and translation can be neglected, and the NED frame is assumed as inertial. There is an extra frame, called the Camera frame, assumed as centered in $\mathcal{O}_c$ but with a relative rotation from the ABC frame. In Figure 1, the most relevant frames are depicted.

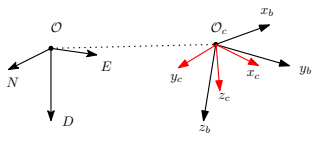


Fig. 1: The NED, ABC and camera reference frames

B. Vectors and Variables

In the present work, the superscript ^{I} will denote vectors expressed in the inertial frame, and ^{B} will denote vectors expressed in the body-fixed frame.

The gravity vector $\mathbf{g}^I \in \mathbb{R}^3$, given in the inertial frame:

$$\mathbf{g}^I = \begin{bmatrix} 0 & 0 & g_0 \end{bmatrix}^\intercal \tag{1}$$

with $g_0 = 9.81 m.s^{-2}$ being the universal gravitational constant.

The torques applied around the center of gravity of the quadrotor are denoted by $\mathbf{M}^B \in \mathbb{R}^3$,

$$\mathbf{M}^B = \begin{bmatrix} M_x & M_y & M_z \end{bmatrix}^\intercal \tag{2}$$

and the forces by $\mathbf{F}^B \in \mathbb{R}^3$:

$$\mathbf{F}^B = \begin{bmatrix} F_x & F_y & F_z \end{bmatrix}^\intercal \tag{3}$$

The quadrotor mass will be denoted by m and the inertia matrix is given in the form of a diagonal matrix, $\mathbf{I}_m = \text{diag}\,(I_x, I_y, I_z)$.

The position of the quadrotor is given by the relative displacement from $\mathcal{O}_c$ to $\mathcal{O}$ and is denoted by $\mathbf{P}^I \in \mathbb{R}^3$:

$$\mathbf{P}^I = \begin{bmatrix} N & E & D \end{bmatrix}^\intercal \tag{4}$$

The linear velocity (groundspeed) of the quadrotor is expressed in the body-fixed frame by $\mathbf{V}^B$ as follows:

$$\mathbf{V}^B = \begin{bmatrix} U & V & W \end{bmatrix}^\intercal \tag{5}$$

The angular velocity in the body-fixed frame will be denoted by

$$\mathbf{\Omega}^B = \begin{bmatrix} P & Q & R \end{bmatrix}^\intercal \tag{6}$$

The quadrotor attitude can be expressed using the Euler angles, thus giving the attitude vector $\mathbf{\Psi}$ (see Figure 2) :

$$\mathbf{\Psi} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\intercal \tag{7}$$

where ϕ, θ and ψ are the roll, pitch and yaw angles respectively.

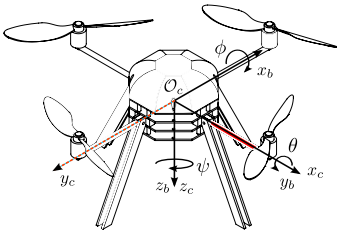


Fig. 2: The ABC and camera frames represented on the rotorcraft

C. Coordinate transformations

Since multiple frames are used in this text, there is the need to translate a vector from one to another. This transformation, defined as a rotation matrix $\mathbf{R}$ from the NED frame to the ABC frame, can be parametrized with the Euler angles. The $c\alpha$, $s\alpha$ and $t\alpha$ symbols represent the cosine, sine and tangent functions of α respectively [3],

$$\mathbf{R} = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\phi s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\phi s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix} \tag{8}$$

D. Non-Linear Model

Obtaining a mathematical model of a quadrotor kinematics and dynamics is essential to perform a set of important computational tasks, such as open-loop stability analysis, control tuning, estimator design, algorithm validation and testing, among others. This modeling of the quadrotor may be presented in the state-space form, with the sate and output vectors $\mathbf{X} \in \mathbb{R}^n$ and $\mathbf{Y} \in \mathbb{R}^p$ in (9), similar to a scheme outlined in Figure 3. The input vector $\mathbf{U} \in \mathbb{R}^m$ vector comprises the each of the propellers rotation speed.

$$\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{U}) \tag{9a}$$

$$\mathbf{Y} = g(\mathbf{X}, \mathbf{U}) \tag{9b}$$

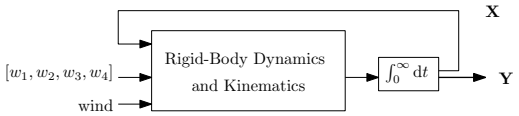


Fig. 3: Block Diagram for the quadrotor dynamical model

1) Kinematics: The kinematic equations relate the angular or translation position with the velocities with the *translation* in (10) and the *rotation* in (11) using the Euler angles.

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \mathbf{R}^\intercal \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & t\theta s\phi & t\theta c\phi \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \tag{11}$$

2) Dynamics: The dynamics can also be subdivided into the translation equation in 13, and the rotation equation in 14.

The $\mathbf{F}_a^B$ vector represents the aerodynamic forces,

$$\mathbf{F}_a^B = -\,\mathbf{k}_a \mathbf{V}_a^B |\mathbf{V}_a^B| \tag{12}$$

where $\mathbf{k}_a$ is an aerodynamic constant and $\mathbf{V}_a^B = \mathbf{V}^B - \mathbf{R}\mathbf{V}_w^I$ is the airspeed defined from the groundspeed $\mathbf{V}^B$ and the wind speed $\mathbf{V}_w^I$. The $|\mathbf{V}_a|$ operator is the module of the airspeed.

$$m\mathbf{a}^B = \mathbf{F}^B + m\mathbf{R}\mathbf{g}^I - \mathbf{\Omega} \times m\mathbf{V}^B + \mathbf{F}_a^B \tag{13}$$

$$\begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} \frac{M_x}{I_x} \\ \frac{M_y}{I_y} \\ \frac{M_z}{I_z} \end{bmatrix} - \begin{bmatrix} \frac{I_z - I_y}{I_x} QR \\ \frac{I_x - I_z}{I_y} PR \\ \frac{I_y - I_x}{I_z} PQ \end{bmatrix} \tag{14}$$

E. State-Space Model

Using the dynamical model presented previously, it is possible to create a 12-state vector in the form described in Equation (9) as depicted in Figure 3 with vectors $\mathbf{V}^B$, $\mathbf{\Omega}^B$, $\mathbf{P}^I$ and $\mathbf{\Psi}$. The state vector $\mathbf{X}$ becomes

$$\mathbf{X} = \begin{bmatrix} (\mathbf{V}^B)^\intercal & (\mathbf{\Omega}^B)^\intercal & (\mathbf{P}^I)^\intercal & (\mathbf{\Psi})^\intercal \end{bmatrix}^\intercal \tag{15}$$

The input vector $\mathbf{U}$ of that model includes the four angular velocities of the propellers,

$$\mathbf{U} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix}^\intercal \tag{16}$$

F. Linear Model

The non-linear state-space model presented in Equation (9) can be approximated to a linear, time-invariant model, or LTI, in the form

$$\dot{\mathbf{X}} = \mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{U} \tag{17a}$$

$$\mathbf{Y} = \mathbf{C}\mathbf{X} + \mathbf{D}\mathbf{U} \tag{17b}$$

where $\mathbf{A}$ is usually called the dynamics or state transition matrix, $\mathbf{B}$ the input matrix, $\mathbf{C}$ the output matrix and $\mathbf{D}$ the direct output matrix.

The proposed linearization consists in approximating the behavior of the non-linear system around an operating point $\mathbf{X_0}$. For the tasks proposed for this text, that operation point is the hovering condition, corresponding to the state $\mathbf{X_0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -z_0 & 0 & 0 & 0 \end{bmatrix}^\intercal$ and the input $\mathbf{U}_0 = \begin{bmatrix} w_1^0 & w_2^0 & w_3^0 & w_4^0 \end{bmatrix}^\intercal$.

For the proposed hovering condition, the input vector $\mathbf{U}_0$ containing the rotors angular velocities is obtained solving $\mathbf{F}^I = m\mathbf{g}$, balancing the total thrust created by the propellers with the prototype weight.

Assuming that the thrust created by a propeller with index i can be approximated with a quadratic relation using a thrust constant $K_{T,i}$ by $F_i = K_{T,i}{w_i}^2$,

$$\sum_{i=1}^{4} K_{T,i}(w_i^0)^2 = mg_0 \tag{18}$$

which leads to Equation (19) assuming similar aerodynamic characteristics of the four propellers and therefore similar thrust constants K_T and rotors angular speed w^0:

$$w^0 = \sqrt{\frac{mg_0}{4K_T}} \tag{19}$$

III. Computer Vision

A. Frames and Vectors

In computer vision, there is the concept of *pose* which is defined as the combination of a camera position and orientation, usually recurring to a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ and a relative translation vector $\mathbf{t} \in \mathbb{R}^3$. The homogeneous coordinates of a $\mathbf{X} \in \mathbb{R}^3$ position vector are also used in this field and are defined as $\mathcal{X} = \begin{bmatrix} \mathbf{X}^\intercal & 1 \end{bmatrix}^\intercal \in \mathbb{R}^4$.

The rotation matrix is usually parametrized using the Euler angles or the quaternions formulations and, since the translational vector $\mathbf{t}$ is three-dimensional, the minimal pose representation of some frame $\mathcal{F}_C$ with respect to a frame $\mathcal{F}_*$ is a six parameter vector, ${}^*\mathbf{r}_C \in \mathbb{R}^6$, defined in Equation (20).

$${}^*\mathbf{r}_C = \begin{bmatrix} {}^*\mathbf{t}_C^\intercal & \left(\mathbf{\Psi}^C\right)^\intercal \end{bmatrix}^\intercal \in \mathbb{R}^6 \tag{20}$$

Three main frames are generally used: the current camera frame, $\mathcal{F}_C$, centered in $\mathcal{O}_c$, the desired camera frame, $\mathcal{F}_*$, centered in $\mathcal{O}_*$ and the inertial frame, $\mathcal{F}_I$. The inertial frame is defined as rigidly attached to the target and assumed motionless relatively to the previously defined NED frame. The camera frame is attached to the camera center of projection, with the z axis along the optical axis and the x and y axes pointing to the right of the image and its bottom, respectively.

Any point $\mathbf{X}^C \in \mathbb{R}^3$, defined in the current camera frame $\mathcal{F}_C$, can be defined in the desired camera frame $\mathcal{F}_* \in \mathbb{R}^3$ by $\mathbf{X}^*$ applying the affine rigid-body transformation in (21):

$$\mathbf{X}^* = {}^*\mathbf{R}_C\mathbf{X}^C + {}^*\,\mathbf{t}_C \tag{21}$$

where ${}^*\mathbf{R}_C \in \mathbb{R}^{3\times3}$ is the rotation matrix from the current to the desired camera frame and ${}^*\mathbf{t}_C \in \mathbb{R}^3$ is the translational vector from $\mathcal{O}_c$ to $\mathcal{O}_*$. Note that the general rotation matrix $\mathbf{R}$ was defined in (8).

B. Homography

To obtain the $\mathbf{X}^C$ coordinates in the $\mathcal{F}_c$ frame, using the affine transformation in Equation (21) results in

$$\mathbf{X}^C = {}^C\mathbf{R}_*\mathbf{X}^* + \frac{1}{d^*}{}^C\mathbf{t}_*(n^*)^\intercal\mathbf{X}^* = {}^C\mathbf{H}_*\mathbf{X}^* \tag{22}$$

where ${}^C\mathbf{H}_* = {}^C\mathbf{R}_* + \frac{1}{d^*}{}^C\mathbf{t}_*(\mathbf{n}^*)^\intercal$ is called the Euclidean planar homography from the desired camera frame to the current camera frame.

The term $\frac{1}{d^*}{}^C\mathbf{t}_*$ induces a scalar ambiguity in this formulation, meaning that the homography matrix can only be estimated up to a scalar factor. In the uncalibrated case, a projective homography, $\mathbf{G} = \mathbf{K}\mathbf{H}\mathbf{K}^{-1} \in \mathbb{R}^{3\times3}$ is defined.

The decomposition of the planar homography matrix results in eight possible solutions: four of them are eliminated by imposing a positive depth constraint. From these, only two are physically admissible assuming that the camera was in the same side of the image plane both in the reference pose and the current pose. By knowing the normal vector to the image plane, $\mathbf{n}$, it is possible to obtain the correct solution and the distance d to correct the translational vector, which is obtained in the raw form $\mathbf{t}/d$. For further detail, see [4].

In practice, the planar homography decomposition is performed by means of Singular Value Decomposition (SVD), using a MatLab® routine developed by Ezio Malis, which was adapted to C.

IV. The QuavIST platform

The QuavIST has a control solution already implemented, based on the Paparazzi® project. In order to improve the capabilities of the platform, a set of software tools was developed in the scope of this work, which are described in the following.

A. Software Tools

The `grab` tool runs on the onboard PC-104, with the purpose of fetching the telemetry data from the Paparazzi® via the USB/CAN channel, decoding relevant data (namely the GPS data and local time) and simultaneously take georeferenced pictures with the Webcam with a programmable period. Another important task of this tool is the direct channeling of the telemetry data to a local binary file, to overcome the loss of packets that constantly happen in outdoor flights with the XBee wireless link.

The `log2bin` tool runs on a Linux terminal, and decodes Paparazzi® binary logs into a text log file along with a configuration file, both in the format used by the Paparazzi® groundstation software, therefore maintaining the compatibility with all the tools that can be used with Paparazzi®.

The `kb_control` tool is used to test the propellers, the USB uplink to send control strings and the general external control interface. It consists in a `ncurses` Linux terminal interface, offering the user the possibility to control the QuavIST with a keyboard, increasing or decreasing the thrust, roll, pitch and yaw desired values. It sends the control string to the Paparazzi® via the USB external control link, with a fixed period (guaranteeing that a string is sent at least each second in order to avoid the disengagement from the Paparazzi® controller). It also grabs the telemetry messages via the USB/CAN channel and decodes the attitude, battery, thrust and system state messages and prints them in the user interface.

The `of_controller` tool was derived from `kb_control`, with the inclusion of a visual servoing control feedback loop. It performs the following tasks:

- decodes telemetry with the attitude data, barometer and SONAR measurements and system state,
- grabs pictures from the imaging camera with a given fixed frequency and performs optical flow,
- estimates relative motion from homography decomposition,
- estimates velocity by high-passing the successive motion estimations,
- calculates control inputs based on an lQR control loop,
- send control inputs via the external control USB link to the Paparazzi®,
- logs all the relevant telemetry data and output control strings in separate text files.

B. QuavIST Physical Parameters

The specific parameters of the QuavIST determined in the scope of this work are presented in the following.

The quadrotor weights $2.93\,\mathrm{kg}$ when carrying a battery. E

The Inertia Matrix is quite difficult to measure and, for the present work, it was obtained by scaling the matrix from [3], resulting in $I_x = I_y = 3.8 \times 10^{-3}\mathrm{kg\,m^2}$ and $I_z = 7.1 \times 10^{-3}\mathrm{kg\,m^2}$.

The $\mathbf{k}_a$ matrix relating the airspeed to the aerodynamic force is obtained from the work in [5],

$$\mathbf{k}_a = \mathrm{diag}\left(6.1 \times 10^{-3}, 6.1 \times 10^{-3}, 25 \times 10^{-3}\right) \mathrm{kg\,m^{-1}} \quad (23)$$

The K_T constant, relating the angular speed of the propellers to the propulsion force, is obtained from tables specific to the propellers of the QuavIST. The value obtained from Prof. Azinheira was $K_T = 2.7 \times 10^{-5}\mathrm{kg\,m\,rad^{-2}}$, with force and torque coefficients of $C_T = 0.0734$ and $C_P = 0.0044$ taken from propeller specific tables and corrected with experimental data from a hovering flight.

C. Linearized State-Space

A numerical linearization about the hovering condition results in the matrices in Equation (24).

$$\mathbf{A} = \begin{bmatrix} \begin{bmatrix} & \begin{bmatrix} 0 & -g_0 & 0 \\ g_0 & 0 & 0 \end{bmatrix} \\ zeros(7,9) & \\ & zeros(5,3) \end{bmatrix} \\ \begin{bmatrix} eye(5) & zeros(5,7) \end{bmatrix} \end{bmatrix} \quad (24a)$$

$$\mathbf{B} = \begin{bmatrix} zeros(2,4) \\ \begin{matrix} -0.010 & -0.010 & -0.010 & -0.010 \\ 0 & -2.294 & 0 & 2.294 \\ 2.294 & 0 & -2.294 & 0 \\ -0.013 & 0.013 & -0.013 & 0.013 \end{matrix} \\ zeros(6,4) \end{bmatrix} \quad (24b)$$

Analyzing the open-loop stability with MatLab®, it is concluded that all the poles are located at the origin; the system is marginally stable, and therefore it needs automatic stabilization. This stabilization is implemented in the Paparazzi®, accepting as input reference roll and pitch angles, yaw rate and thrust force. To calculate this references, a high-level control loop is required, which is designed using the Optimal Linear Control theory.

V. Visual Tracking

With visual tracking methods it is primarily desired to follow some target in a sequence of images. This is accomplished by sequentially extracting correspondences of the same real-world 3D points projected onto two consecutive images, taken from two different poses but exhibiting some intersection. For real applications, it is required for these methods to be not only time-efficient, but also accurate. Visual tracking methods are usually studied as optimization problems, with different functions to be optimized for each strategy.

There is a vast number of possible approaches concerning visual tracking. Three main strategies are analyzed in the current work: the Speeded-Up Robust Features (SURF) detector and Fast Approximate Nearest Neighbor Search (FLANN) matcher, the Harris corner detector with Lucas-Kanade Optical-Flow with Pyramids (LKOFP) and, finally, dense tracking with the Efficient Second-Order Minimization (ESM).

A. Optical Flow

The purpose of the techniques concerning the optical flow is calculating the relative motion of a camera relatively to a scene, based on visual tracking of relevant image features. These techniques are usually subdivided onto two main classes: *dense* optical flow algorithms and *sparse* optical flow algorithms [6].

The dense techniques associate a velocity to each pixel in an image, or equivalently a relative displacement of the same pixel from a reference image to a current image. One example of a dense technique is the Horn-Schunck method [7]. Sparse techniques rely on some algorithm previously applied to specify which pixels offer some interesting characteristics and should be tracked (one of these selection methods would be, for example, the Harris detector. An example of a sparse technique would be the Lucas-Kanade method [8].

In general, dense optical flow techniques tend to be slower, which explains the general trend to use sparse algorithms in engineering applications.

The Lucas-Kanade sparse algorithm was chosen among the available options due to its satisfying performance and popularity ([9] and [10]), along with the fact that there are several robust implementations, one of them included in the OpenCV library. This library implements both the original algorithm and the version with pyramids which has shown great improvements to the original version.

B. Velocity estimation with Optical Flow

In order to close the guidance control loop, some estimative of the velocity vector $\mathbf{V}^B$ must be obtained. The available data from the Paparazzi®/UAVision® filters could be used, but an effort was made to lower the dependency on external data and to develop a full and self-sufficient control strategy. This brings some advantages, such as lower computational load, higher robustness to external faults and the provision of complementary measurements to the available flight data. Furthermore, there is no guarantees on the GPS availability (especially for indoor testes where it simply does not exist), and the velocity estimation using visual data might produce better results.

Two alternative approaches were studied and are now explained.

High-Pass Approach: A naive approach for velocity estimation based on Optical-Flow would be the computation of the time derivative of the position estimates, which would be a sufficient procedure in a deterministic environment. However, when in the presence of random noise, the derivative operations largely magnify the effect of noise. For that reason, a low-pass filter could be introduced, therefore damping the high-frequency phenomena introduced by the noise. The integration of the derivative term in series with a low-pass filter results in a high-pass filter, with a continuous time transfer function given by the Laplace transform

$$G_{hp}(s) = \frac{s}{s\tau_{hp} + 1} \tag{25}$$

with the time constant τ_{hp} as a tuning parameter with direct influence in the filter cutoff frequency. It must be taken into account that τ_{hp} must verify the Nyquist sampling theorem to avoid anti-aliasing, yielding $\tau_{hp} \leq T_G/2$. In fact, the effect of that parameter was studied experimentally in simulation, with the quadrotor performing random maneuvers while computing the velocity error. It was observed that the optimal value for τ_{hp} is precisely $\tau_{hp} = T_G/2$.

Kalman-Filter Approach: A common approach when estimating the velocity of the camera frame is the simple computation of the optical flow vector between two consecutive image frames, transforming that vector to euclidean coordinates and dividing by the time between the two image samples.

However, when dealing with non-negligible attitude, the angles imposed by the angular velocities introduce significant distortions to the image pixels, and must be accounted for. A common approach is the integration of Inertial Measurement Unit (IMU) data (see [11] and [12]).

Assuming the mean optical flow field $\overline{OF}$ vector (in pixel/s) can be decomposed in a rotational component, $\overline{R}_{OF}$, and translational component $\overline{T}_{OF}$:

$$\overline{OF} = \overline{R}_{OF} + \overline{T}_{OF} \tag{26}$$

and that these components in each tracked point i are related to the mean optical flow field by the intrinsic camera parameters, present in its calibration matrix, and each point coordinates (u_i, v_i) by:

$$R_{OFi} = \begin{bmatrix} \frac{u_i v_i}{f} & -\left(f + \frac{{u_i}^2}{f}\right) & v_i \\ \left(f + \frac{{v_i}^2}{f}\right) & -\frac{u_i v_i}{f} & -u_i \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \tag{27a}$$

$$T_{OFi} = \frac{1}{D} \begin{bmatrix} -f & 0 & u_i \\ 0 & -f & v_i \end{bmatrix} \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{27b}$$

Assuming a hovering condition and therefore $W = 0$, the estimation of the velocity vector in the body frame $\mathbf{V}^B$ can be obtained using the estimated mean optical flow field $\overline{OF}$ and the direct angular velocities measurements from the IMU, $\widetilde{\mathbf{\Omega}}^B$, with the Kalman filter approach with state vector $\mathbf{x}^v = \begin{bmatrix} U & V & P & Q & R \end{bmatrix}^\intercal$ and output vector $\mathbf{y}^v = \begin{bmatrix} \overline{OF}_x & \overline{OF}_y & \widetilde{P} & \widetilde{Q} & \widetilde{R} \end{bmatrix}^\intercal$.

C. Speeded-Up Robust Features

The SURF algorithm was introduced in [13] as an improved feature detector and descriptor from the already existent methods. It is composed of two distinct phases: interest point detection and feature description. The computation of the interest points is based on approximations to the Hessian matrix including a scale factor σ. Both the scale invariance and rotation invariance are guaranteed by the use of approximations to the circular Gaussian filters: the scale invariance is accomplished with efficient tracking in multiple scalings of the Gaussian filters for the same image, computing a pyramid of scaled images, while the rotation invariance relies on the fact that the Gaussian filters are circular. The SURF feature description rely on a pixel's neighborhood in order to include information about a pixel's spatial intensity patterns: it uses the same interest point description proposed by the Scale-Invariant Feature Transform (SIFT) method [14], which computes a histogram of local gradients for each interest point.

One of the key performance advantages of the SURF algorithm is the use of an *integral image*. The integral image,

denoted $\mathcal{I}_i$, with each pixel $\mathcal{I}_i(u,v)$ consisting of the sum of the pixels within the rectangular window from the origin of the image up to that pixel's coordinates, (u,v), is computed as follows:

$$\mathcal{I}_i(u,v) = \sum_{i=0}^{i\leq u}\sum_{j=0}^{i\leq v}\mathcal{I}(i,j) \tag{28}$$

The computation of this integral image lightens the computational load when calculating any sum of pixel intensities over an upright and rectangular area, decreasing the number of elementary operations to three sums and four memory requests.

D. Effective Second Order Minimization

The tracking approach proposed in [15] with real application results in [16] does not rely on geometric primitives for its optimization function: it relies on pixel intensity or photometric measurements, and is therefore classified as a dense tracking algorithm. The deformation undergone by some region within the image is estimated with the minimization of a cost function, usually the *sum of squared differences* between a reference and current image in Equation (29),

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{q}\left[\mathcal{I}\left(\mathbf{w}(\tilde{\mathbf{G}}\mathbf{G}(\mathbf{x}))(\mathbf{p}_*^i)\right) - \mathcal{I}^*(\mathbf{p}_*^\mathbf{i})\right]^2 \tag{29}$$

where $\mathcal{I}^*$ is the reference image and $\mathcal{I}$ the current image, warped by $\hat{\mathbf{G}} = \tilde{\mathbf{G}}\mathbf{G}(\mathbf{x})$, where $\tilde{\mathbf{G}}$ is an estimative for the projective homography, $\mathbf{G}(\mathbf{x})$ is the incremental projective homography and $\mathbf{x} \in \mathbb{R}^8$ is a parametrization of $\mathbf{G}$ up to a scalar factor. The total number of pixels in $\mathcal{I}$ is given by q.

The strategy of the ESM is to establish second-order approximations to (29) based on Taylor-series expansions, and creating approximations to the Jacobians resulting from the second-order expansion, and perform an iterative minimization process until the error reaches a defined threshold.

E. Comparative Study of the Visual Tracking alternatives

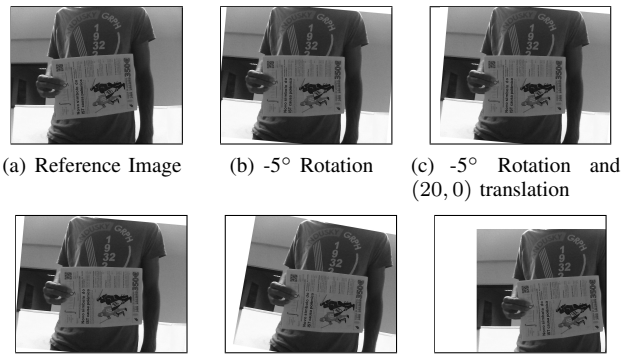
In order to validate the Visual Tracking and Homography estimation and reconstruction algorithms, a simple simulation was developed. This simulation consists of transforming a reference image with the GIMP and trying to recover the parameters of that transformation afterwards. The set of resulting images analyzed with the visual tracking alternatives is in Figure 4.

In order to decide which would be the best visual tracking solution among the studied options, Table (I).

From the analysis of Table (I), the pros and cons in the next subsections are infered.

SURF/FLANN:

- Results in the best accuracy in pose estimation
- High robustness when concerning major transformations
- Very slow, even when setting the minimum Hessian threshold as extremely low
- Not many parameters are given to the regular user for customization or tuning
- Depends on the installation and onboard availability of the OpenCV library



(a) Reference Image (b) -5° Rotation (c) -5° Rotation and $(20,0)$ translation

(d) -5° Rotation and $(20,20)$ translation (e) $(140,36)$ translation and -10° rotation (f) Translation of $(312,98)$

Fig. 4: Reference images for reconstruction validation

	SURF/FLANN	LKOFP	ESM
Source (4)		**Error**	
(b)	0.0145°	0.0045°	0.0181°
	0 pixels	0.7843 pixels	0.0395 pixels
(c)	0.0385°	0.0442°	0.0491°
	0.0460 pixels	0.7663 pixels	0.0491 pixels
(d)	0.0572°	0.0177°	0.0181°
	0.2100 pixels	0.7641 pixels	0.0252 pixels
(e)	0.0004°	0.0524°	0.2033°
	0.9818 pixels	1.0381 pixels	111.8459 pixels
(f)	0.0045 pixels	0.1741 pixels	321.8870 pixels
	0.5825°	0.0000°	14.6559°
Duration	2.2570 s	0.5858 s	0.4343 s

TABLE I: Overall Performance benchmark

LKOFP:

- Reasonable precision for estimating pose
- Good robustness when applied for major transformations
- Very fast even for high resolution images
- High number of user-defined parameters, excellent for tuning
- Depends on the installation and onboard availability of the OpenCV library

ESM:

- Reasonable precision for estimating pose
- Very fast, depending on the size of the region of interest (ROI)
- Reasonable number of user-defined parameters
- Poor robustness when applied for major transformations
- Does not depend on any external C library

Considering the facts previously presented, the LKOFP seems to be the more appealing alternative to the studied problem. Indeed, due to the nature of its pyramid implementation, the LKOFP will not lose tracking in the case of a major sudden movement between two frames (at the cost of instantaneous increased computation time, whose effect must be studied in order to guarantee the control frequency). The LKOFP has also proven to be suitable for good velocity estimations [17]

[11], which is extremely useful to close the control loop. That alternative also seems to exhibit a fairly high computation frequency even for large images, with satisfiable error.

At the time of writing of this text, the SURF and FLANN approach is not yet suitable for realtime applications, as it takes a long time to perform the overall tracking process.

The ESM approach is quite decent for realtime applications; however, due to the nature of the dense tracking algorithms, the ESM depends on high-frequency to maintain accurate estimations, for it easily loses the tracked ROI in case of quick motions. The free implementation of the ESM, available online (version 1.2 at the time of writing), also seems to be extremely vulnerable to sudden luminosity changes, which might easily happen in an external environment. Nonetheless, the ESM could also suit as a second alternative for realtime visual servoing.

VI. Simulation

Both the simulation and implementation low-level loops require a guidance loop. In the scope of this work, that loop was developed using the LQR control and Computer Vision estimation. In the following, a PBVS control loop is developed and explained.

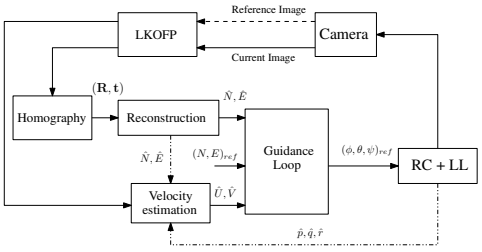


Fig. 5: Visual Servoing with position and velocity feedback

The homography reconstruction outputs both 3D position and attitude in the form of Euler angles. However, the altitude and attitude estimations are known to be less accurate and therefore discarded discarded in the loop in Figure 5, although it is possible to make robust estimations of depth from optical flow [18]. Some extra robusteness comes from the integration of velocity feedback, estimated whether based on direct filtering of the reconstructed position, or Kalman filter estimation integrating rate-gyros and direct OF measurements. In Figure 5, the block [RC+LL] represents the full rotorcraft dynamics including the low-level control loop.

The LQR gains in the *Guidance Loop* block are computed using Matlab® for the high-level feedback,

$$\mathbf{K}_G = \begin{bmatrix} 0.000 & 0.085 & 0 & 0.035 \\ -0.085 & 0 & -0.035 & 0 \end{bmatrix} \tag{30}$$

with the folowing diagonal weighting matrices:

$$\mathbf{Q}_G = \text{diag}\left(\begin{bmatrix} 0.1 & 0.1 & 1 & 1 \end{bmatrix}\right) \tag{31a}$$

$$\mathbf{R}_G = \text{diag}\left(\begin{bmatrix} 750 & 750 \end{bmatrix}\right) \tag{31b}$$

A. Sample Simulation

In order to test he controller performance, a simple maneuver is performed by the quadrotor in order to illustrate the operation of the controller previously described.

The simulated stabilization loop includes an internal altitude regulator, and takes as input a reference altitude and the reference Euler angles.

Figure 6 displays the plot of the system states when an initial position error of $\begin{bmatrix} N_{err} & E_{err} & D_{err} \end{bmatrix}^\intercal = \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}^\intercal$ m is imposed at a constant altitude of $D = -5$m and the quadrotor tries to return to the reference hovering position at $\begin{bmatrix} N & E & D \end{bmatrix}^\intercal = \begin{bmatrix} 0 & 0 & -5 \end{bmatrix}^\intercal$ m. The simulation was performed using the high-pass filter velocity estimator, tracking 50 points in an image of 640×640 pixel.

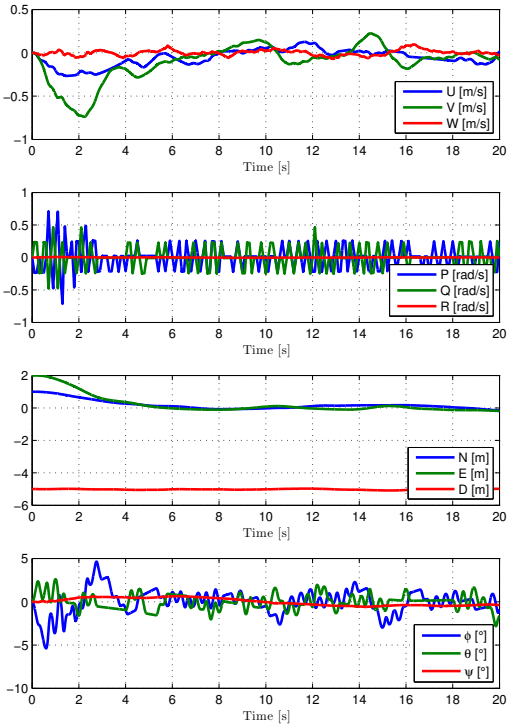


Fig. 6: LKOFP Guidance Controller Response

It can be observed that the quadrotor returns to the reference position in about 5 seconds and is capable of maintaining a hovering condition around this position with satisfiable performance. After stabilization, some major fluctuations are however observed in the Euler angles and the Angular velocities: this is due to the high variance of the gyroscopes measurements and their influence in the Attitude-Heading Reference System (AHRS) filter. Some velocity fluctuations are also observed, which can be explained by the low image resolution affecting the velocity estimation filter.

B. Velocity Estimators Performance Simulation

In order to evaluate the two alternative OF-based velocity estimation approaches, a set of Simulink® simulations was made with the quadrotor trying to hover. The chosen performance indicator was the root-mean-square value of the horizontal displacement from the target.

High-Pass Filter Approach: After discretization for a high-level control frequency T_G and for $\tau_{hp} = T_G/2$, the continuous highpass transfer function in Equation (25) becomes

$$H(z) = \frac{z-1}{zT_G} = \frac{V(z)}{P(z)} \tag{32}$$

where $V(z)$ can express both the $\mathcal{Z}$-transform of the velocity in the x or y axes, and $P(z)$ the $\mathcal{Z}$-transform of the position in the same axis.

The pole at the origin of the imaginary plane (and therefore inside the unitary circle) guarantees the stability of the filter. A numerical application of the transfer function in (32) is the equivalent difference equation in (33) for $T_G = 1/10$s, where $v(n)$ can express once more both the velocity in the x and y axes, and $p(n)$ the position in the same axis.

$$v(n) = 10p(n) - 10p(n-1) \tag{33}$$

Kalman Filter Approach: A Kalman Filter implementation is usually calibrated recurring to the diagonal values of the $\mathbf{R}^v$ and $\mathbf{Q}^v$ covariance matrices. The values of $\mathbf{R}^v$ are usually based on statistical noise properties, and the diagonal values of $\mathbf{Q}^v$ might be seen as the level of trust of the state modeling.

Using statistical noise properties obtained in laboratory, and after some bold calibration (which must be refined in a real implementation), the resulting covariance matrices are given in Equation (34) (note that the noise-spectral density relative to the angular rates are in $\text{rad}^2\,\text{s}^{-2}\,\text{Hz}^{-1}$ and $\text{diag}(\mathbf{v})$ is a diagonal matrix with values given by the vector $\mathbf{v}$).

$$\mathbf{R}^v = \text{diag}\left(\begin{bmatrix}0.017 & 0.017 & 1.48\times10^{-6} & 1.66\times10^{-6} & 1.37\times10^{-6}\end{bmatrix}\right) \tag{34a}$$

$$\mathbf{Q}^v = \text{diag}\left(\begin{bmatrix}1\times10^{-3} & 1\times10^{-3} & 1\times10^{-2} & 1\times10^{-2} & 1\times10^{-2}\end{bmatrix}\right) \tag{34b}$$

Comparative Study: From Figure 7, one can establish a comparative study between the high-pass filter and the Kalman filter approach. It can be concluded that both alternatives exhibit similar results, although the Kalman filter approach clearly shows lower accuracy for lower control frequencies. With such a similar performance, one might be tempted to implement the Kalman filter solution, due to its higher integration of sensor data and the fact that the covariance matrices can be used to tune the filter response. However, the Kalman filter represents a much higher computational load, which might not bring relevant advantages. However, the high-pass solution consists in two memory accesses and a subtraction, which makes it a good candidate for a real implementation. Nonetheless, both alternatives were implemented in C and tested in realtime.

C. Controller Performance Simulation

In order to study the influence of some factors in the controller performance, and to compare the performance of the two velocity estimation alternatives, some simulations were performed. This analysis is based in Simulink® simulations.

The following simulations were all performed using the following default parameters (except, of course, for the studied parameter itself): a square image with 320×320 pixels, 10 tracked points, $T_G = 1/10\,\text{s}$, $T_{LL} = 1/50\,\text{s}$, $\tau_{hp} = T_G/2\,\text{s}$. Fourty simulations were performed for every different value, each simulation using a randomly distributed point cloud, with the mean located in the hovering target position, and unitary standard deviation (in meter RMS).

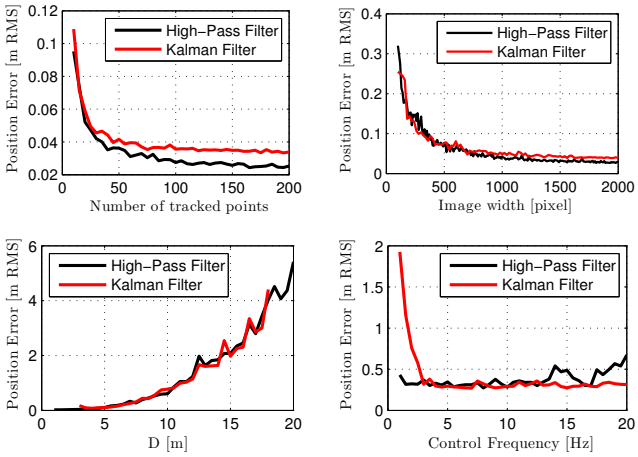


Fig. 7: Influence of various factors in the control loop performance

From Figure 7, it can be concluded that tracking more than 50 image points might not bring relevant improvements to the performance, but will increase the computational load and therefore the allowed maximum control frequency: a control frequency of 1 Hz may be enough for the high-pass solution, but a frequency of around 3 Hz might be necessary for the Kalman-filter alternative. An image of 500×500 pixels might be enough for decent tracking performance. Finally, the distance of the quadrotor to the target in the focal axis has a significant impact on the performance of the controller: the further away the quadrotor is from the target, the more a slight fluctuation in a pixel affects the results (as this is multiplied by the distance in the focal axis, which can be approximated by D if the target is the ground).

VII. Realtime Implementation

A. Realtime Controller Architecture

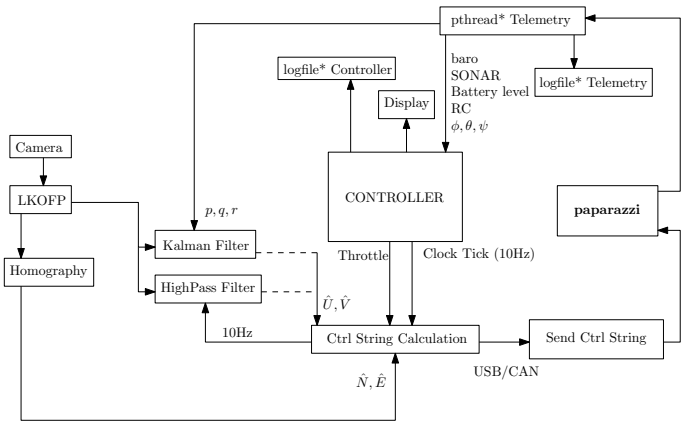


Fig. 8: Flowchart of the realtime controller

The chosen architecture is based on the Linux signal handlers [19], emulating a realtime clock using interrupt calls, which offers great reliability. The program calls an independent thread

which is constantly logging and outputting important data from the telemetry bus, and enters a main loop which is constantly acquiring camera frames and estimating the optical flow, outputting data to the velocity estimators. When the interrupt function is called (at 10 Hz), the motion reconstruction and velocity estimation with the high-pass filter are calculated and the control reference, corresponding to ϕ, θ and throttle, is calculated: these tasks are performed *inside* the interrupt routine due to their low computational load and can guarantee very low fluctuations in the control reference output; on the other hand, acquiring camera frames, estimating optical flow and estimating the Kalman filter output are done *outside* the interrupt routine, as they are heavier computational tasks and my influence the control reference output frequency. This reference output is made via the USB/CAN converter directly to the Paparazzi® microcontroller. Both the control outputs, velocity and motion estimations are logged within the main loop to a different text file to avoid cache conflicts.

B. Vertical regulator

The low-level loop from UAVision® includes a throttle reference input, which must be calculated independently in a control loop in Figure 9. It was also designed using LQR control with the feedback gains obtained from [20],

$$\mathbf{K}_h = \begin{bmatrix} 423 & 0 \\ 0 & 230 \end{bmatrix} \tag{35}$$

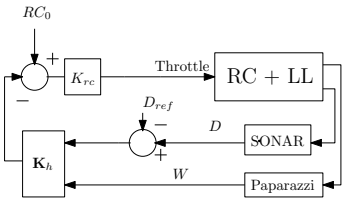


Fig. 9: Vertical Position Auxiliary Controller

C. Assessment of the estimation results

Preliminary Testing: As a first test to the estimation algorithms in a realtime situation, the model was put facing a wall at approximately 1m in the focal axis, while logging the reconstruction and velocity estimation results (from the High-Pass velocity approach) in order to develop a simple statistical analysis. The controller was tuned for tracking 100 points in a 640×480 pixel image at 10Hz. At the time of writing of this section, it was concluded that the velocity estimation using the Kalman filter approach would not be possible due to the lack of a telemetry message integrating both the angular rates and Euler angles, with enough sampling rate and with known scale factors. For that reason, only the high-pass alternative was studied.

The results of this analysis are in Table (II).

Realtime Controller Testing: During the experiment with the rotorcraft and the camera facing a wall at an approximate distance of 1 m in the focal axis, it was concluded that at this distance, a minimum pose variation in the rotorcraft would have a huge effect in the image point tracking. Furthermore, due to the unavoidable presence of the ground adjacent to the camera, if the Harris corner detector algorithm picks an image point corresponding to the ground, the planar assumption made for the homography is not respected. For these reasons, a new experiment had to be performed.

OF Frequency [Hz]	Vel. σ [m/s$_{\text{RMS}}$]	Trans. σ [m$_{\text{RMS}}$]
23.1165	0.004	0.003

TABLE II: Statistical Analysis of the Controller Estimators

The alternative was to hang the rotorcraft in a 3 m crane available in the laboratory, as shown in Figure 10, and let it swing in order to study the effect of small pose variations in the tracking values. The results are shown in pixel, and are not scaled with the distance in the focal axis.

Fig. 10: Preliminary controller testing environment

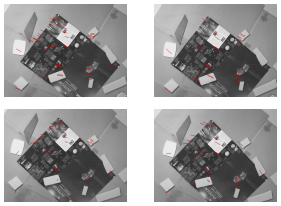


Fig. 11: Onboard screenshots of some processed images with tracked points

In Figure 11, there are some images taken from the QuavIST camera, with red lines illustrating the pixel displacements from the reference position, estimated optical flow. In Figure 12, the evolution of the relevant state variables for the guidance controller is plotted, and Table (III) shows the computed variance values for the state variables.

Parameter	x_c [pixel]	y_c [pixel]	$\dot{x}_c$ [pixel/s]	$\dot{y}_c$ [pixel/s]
Value	4.3286	2.1671	19.2601	9.6363
Parameter	ϕ [°]	θ [°]	ψ [°]	
Value	0.1425	0.3044	0.1567	
100 tracked points	640×480 pixel image		10Hz	$D = -2.41$m

TABLE III: Variance values during preliminary controller testing (RMS values)

It can be observed that the small pose fluctuations impose significant oscillations in the image, which leads to the conclusion that, although a guidance controller could be implemented with the selected algorithm, it would require tuning effort.

Flying the Quadrotor: Due to the lack of time to test the controller exhaustively, it was not possible to completely check its robustness, not only concerning visual tracking, but also motion reconstruction and velocity estimation.

Although the controller was already running onboard, it was decided that it would be a big risk flying the QuavIST in an unknown situation. The main problem directly concerns the visual tracking: it was not possible to test some environments that would exhibit good and enough features to track for the LKOFP. Furthermore, the rotorcraft would have to fly at a

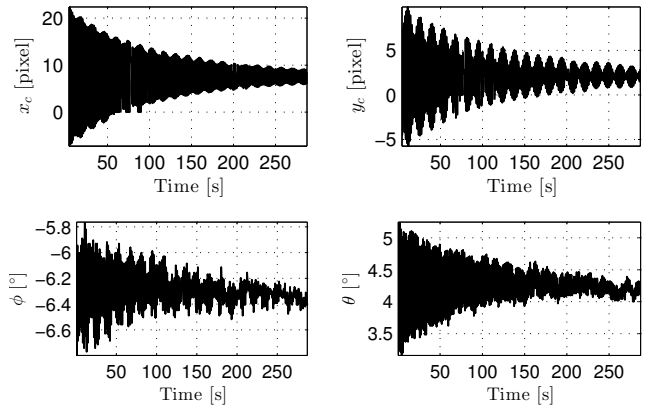


Fig. 12: Evolution of relevant state variables during preliminary controller testing

relatively high altitude (probably around 5m) to assure that a sudden small attitude fluctuation would not move the tracked points outside the camera angle, which would restrain the reaction of the manual pilot in case of some error and, at this altitude, the SONAR would have to be replaced by the barometric pressure sensor, introducing additional noise to the altitude measurements.

VIII. Conclusions and Future Work

A. Conclusions

Since the QuavIST was a new platform, unknown to the students, a great effort was required for understanding the platform and being able to use it. Several tools have to be developed, which turned out to be of great importance.

Tt was observed that the linear optimal control theory (LQR) is a good approach when designing a PBVS control loop, instead of the usual techniques based on Lyapunov stability theorems.

The Optical-Flow strategy, namely the Lucas-Kanade algorithm with pyramids, proved to be the most suitable visual tracking alternative to the aim of this work. It offers not only decent homography estimation/reconstruction results at a great frequency, but also the possibility to estimate velocity. The SURF and ESM are not sufficiently mature ate the moment of writing of this work: the first is still very slow but offers great estimation results, while the former offers a great sampling time but lacks robustness.

Both the number of tracked image points and image dimensions have a great influence in the control error, and can overcome each other (it is possible to have good results with low resolution, if the number of tracked points is larger, and vice-versa).

B. Future Work

In order to conclude the objective of this work, exhaustive tests should be made to the realtime controller, and the flight test could be performed in a known environment (a closed playground like the AEIST is suggested) Concerning the visual servoing techniques, some other control laws could be tested, along with different tracking strategies (being the ESM the most promising one at the moment). Another promising alternative would be the implementation of an IBVS visual servoing strategy, by defining error functions directly based on images instead of performing reconstruction, and therefore eliminating the error associated to it.

Finally, an effort should be made in order to obtain a more accurate camera calibration matrix, and more tests to the homography reconstruction should be performed, with euclidean data.

References

[1] F. Chaumette and S. Hutchinson, "Visual servo control part 1: Basic approaches," *IEEE Robotics and Automation Magazine*, 2006.

[2] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, 1996.

[3] B. S. M. Henriques, "Estimation and control of a quadrotor attitude," Master's thesis, Instituto Superior Técnico, 2011.

[4] T. F. F. Gonçalves, *Aircraft Control by Visual Servoing*. PhD thesis, Instituto Superior Tecnico and Institut National de Recherche en Informatique et Automatique, June 2011. Provisory document.

[5] G. Lopes, "Desenvolvimento e implementação de um controlador num quadrirotor," Master's thesis, Instituto Superior Tecnico, December 2009.

[6] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media, Inc., 2008.

[7] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *ARTIFICAL INTELLIGENCE*, vol. 17, pp. 185–203, 1981.

[8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, pp. 121–130, 1981.

[9] J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. A. Burkitt, "Performance of optical flow techniques," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, pp. 236–242, jun 1992.

[10] G. Emont, "Gstreamer and opencv for image stabilisation," 2012. [Online; accessed 15-October-2012].

[11] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "Mav navigation through indoor corridors using optical flow," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3361–3368, may 2010.

[12] E. Rondon, L. R. Garcia-Carrillo, and I. Fantoni, "Vision-based altitude, position and speed regulation of a quadrotor rotorcraft," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 628–633, oct. 2010.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," 2003.

[15] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization." IEEE/RSJ International Confference on Intelligent Robot Systems, 2004.

[16] E. Malis and S. Benhimane, "A unified approach to visual tracking and servoing," *Robotics and Autonomous Systems*, vol. 52, pp. 39–52, 2005.

[17] L. Muratet, S. Doncieux, Y. Briere, J.-a. M. A, and P. E. Blouin, "A contribution to vision-based autonomous helicopter flight in urban environments," *Robotics and Autonomous Systems*, vol. 50, pp. 195–209, 2005.

[18] B. Shahraray and M. K. Brown, "Robust depth estimation from optical flow," in *Computer Vision., Second International Conference on*, pp. 641–650, dec. 1988.

[19] Linux man-pages project, "signal(7) - linux manual page," 2013. [Online; accessed 8-March-2013].

[20] J. Cafe, "Controlo de voo e estimacao de atitude de um quadrotor," Master's thesis, Instituto Superior Tecnico, 2013. Unpublished.