Quadcopter Automatic Landing on a Docking Station

Tiago Gomes Carreira
tiago.carreira [at] tecnico.ulisboa.pt
Instituto Superior Técnico
October 2013

Abstract— **The goal of this thesis is the control of a quadcopter, with the aim of its spacial stabilization and its autonomous landing. The control approach seeks for full autonomy of the robot, by considering only internal sensors and processing. The method chosen for landing target recognition was the vision-based (with a common camera). Through computer vision, we estimate the quadcopter's relative pose and then proceed to control it and its autonomous landing. A method was developed to estimate the relative position and orientation of a quadcopter using planar markers and computer vision algorithms. In the end, some tests are performed on the simulator, on which an autonomous quadcopter landing is achieved.**

Keywords— **Unmanned Aerial Vehicles, Quadcopter, Target Tracking, Vision-Based Control.**

I. INTRODUCTION

A. Motivation

A commonly used Unmanned Aerial Vehicle (UAV) is the quadcopter, as it is versatile and it has a relatively simple model (when a set of simplifying assumptions is taken and is valid).

One of the problems of this kind of vehicles is autonomy. The energy needed to perform the flight is stored in batteries, which are heavy and therefore limit the autonomy of the robot. Even with fully charged batteries and in a controlled environment, the quadcopter is able to fly for only a few minutes (the quadcopter used in this thesis can fly up to 20 minutes). It is obvious the need to charge the vehicle batteries very often. Today, this charging job has to be done with direct human interaction, i.e. someone must land the quadcopter manually.

The motivation to do this procedure automatically is obvious, and making the quadcopter able to autonomously land opens a lot of possibilities. The most obvious is the possibility to let the quadcopter charge itself by docking on a charging station, without the constant need for human intervention.

B. State of the Art

In the last few years, the usage and deployment of UAVs (also known as drones) have been growing, from hobby to military applications. Some of those applications include surveying, maintenance and surveillance tasks, transportation and manipulation, and search & rescue ([1], [2]).

Within the UAVs field, there are several possible platforms but the quadcopter multi-rotor platform was the chosen one, for its versatility and simplicity. Since the quadcopter is being so used, there are many tutorials and guides on how to acquire, build, control, measure and use this platform, such as [3]–[7]. Even within the same kind of structure there are many possibilities, mainly when referring to open-source autopilots, as it is stated in [4]. Despite the variety of choices, the aim of them all is the same – having a quadcopter flying and being able to easily control the 6 DOF with only four variables.

In order to correctly control this originally unstable vehicle, a very good model of the system and also a feedback control system are required. The usual navigation device available on a UAV is an Inertial Measurement Unit (IMU) ([8]), which uses accelerometers, gyroscopes and magnetometers to provide information about the attitude of the vehicle. But since this unit works with linear and angular accelerations, the output information will suffer from accumulated error because an integrator is continuously adding (even if small) errors, which leads to drift between the system estimate and the real pose of the vehicle. For these reasons, the common sensors available on the quadcopter are not enough for the purpose of this work.

Currently, there is a very accurate system that is able to interact with an UAV and get its full pose with so much precision that even some acrobatic moves are possible to perform on a quadcopter, such as multi-flips moves, as developed by [9]. The Vicon motion capture system ([10]) is so good it is often used as a ground truth system (mentioned in [3], [4], [6], [9], [11]–[15]). This motion tracking system is based on multiple cameras, with very high frame rates (from 120 to 1000 fps) capable of detecting tiny movements of the quadcopter. In a well calibrated environment, it is possible to perform the acrobatics viewed in [16]. All this work provides top results, but it comes with a disadvantage – the need of external devices to track and control the vehicles.

The ISR's Search and Rescue project [2] is one of the applications where the Vicon-like systems cannot be used. In such kind of environments there is the need of another system, capable to provide a good estimation for the (relative) position and orientation of the quadcopter. There are some options, related with Augmented Reality (AR) which may give a good solution, mostly because this kind of systems require a much more simple structure and calibration: a regular calibrated (and one only) camera, a printed marker board and an AR software. At the moment there are a few very good solutions for AR processing, namely the ArUco [17] (used further) or the ARToolKit [18]. These systems

are intended to apply computer vision techniques in order to compute the position and orientation of a markers board with respect to (WRT) the camera. This kind of solution is very interesting because it performs in real time and thus it may be used in an onboard processing environment.

Although the kind of sensor systems stated here provide absolute pose of the quadcopter, in this thesis, the relative pose estimation problem will be addressed. As an autonomous landing on a station is intended, only a relative pose to the station is enough to design a controller. Furthermore, a relative localization system is more useful for its simplicity and it is often possible to implement it.

C. Objectives and Contributions

The objective of this thesis consists on making the quadcopter to perform an autonomous landing on a docking station. It is also a requirement that the quadcopter performs this task without relying on some highly calibrated environment, thus using only on board devices to achieve this goal.

Specifically, it is used a quadcopter equipped with a downwards regular camera and an onboard processing unit (Pandaboard). In order to help the identification of the docking station, a markers board is used. The image captured by the camera is processed and the markers must be detected and identified; computer vision algorithms are applied for estimating the pose of the quadcopter. This pose estimation is then used for controlling the quadcopter's attitude and performing the required maneuvers to achieve the desired goal: lead the quadcopter to land on the docking station autonomously.

In this thesis, the following steps were achieved:

- Visual-based pose estimation algorithms implementation and reliability analysis;
- Autonomous landing algorithm;
- Fully autonomous concept implementation, without external devices.

D. Outline

In Section II the basic concepts are reviewed, by introducing the quadcopter and its physics model, and by reviewing the basics on image processing, computer vision and control analysis.

The Section III is where the development part is stated. It explains the vision-based algorithms developed, the controllers used and the conceptual solutions for the initial problems.

The hardware and software implementations are stated in Section IV. The configurations chosen, the markers used and the system schematics is also included in this section. It has also the description of some tests made over the algorithms used, in order to classify the quality of the chosen methods.

The last section focus on testing the developed algorithms and showing their results.

II. BASIC CONCEPTS

A. Quadcopter Model

The structure of the quadcopter is a symmetrical, cross configuration, with the propellers fixed and parallel. Two propellers are rotating clockwise (front and rear propellers) and the other two propellers are rotating counterclockwise (left and right propellers). All the propellers are built so their air flow points downwards for the described rotation direction so they get an upward force. It will be always considered that all rotors are linked through a rigid link structure.

To describe a 6 Degrees of Freedom (DOF) rigid body it is usual to define two reference frames [19]: Ground inertial reference G-Frame and Body-fixed reference B-Frame (Fig. 1). The G-Frame $(\boldsymbol{O_G}, X_G, Y_G, Z_G)$ is a fixed right-hand referential with X_G pointing towards North, Y_G pointing towards West and Z_G pointing upwards. The B-Frame $(\boldsymbol{O_B}, X_B, Y_B, Z_B)$ is a moving right-hand referential with X_B pointing towards quadcopter's front, Y_B pointing towards quadcopter's right and Z_B pointing downwards. The origin of this Frame is on the quadcopter's center of mass.

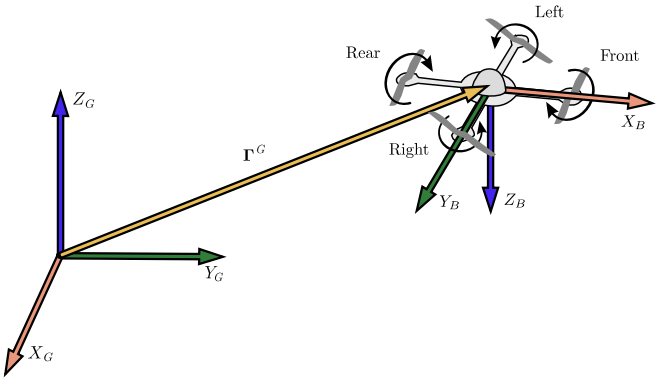


Fig. 1: Frames and references considered (G-Frame and B-Frame)

The pose of the quadcopter, usually defined in the G-Frame, is a 6 DOF vector composed by a linear position and an angular position: $\boldsymbol{\xi} = (\boldsymbol{\Gamma^G}, \boldsymbol{\Theta^G}) = (X, Y, Z, \phi, \theta, \psi)$, with $\boldsymbol{\Gamma^G} = (X, Y, Z)[m]$ being the usual Cartesian coordinates, and $\boldsymbol{\Theta^G} = (\phi, \theta, \psi)[rad]$ being the roll, pitch and yaw which are rotations around X^G, Y^G and Z^G, respectively.

The G-Frame and the B-Frame are related with a rigid body transformation, i.e. a translation $\boldsymbol{\Gamma^G}$ and a rotation given in Z-Y-X (yaw-$pitch$-$roll$) Euler angles, defined by Eq. (1).

$$\begin{aligned} \boldsymbol{R_\Theta} &= \boldsymbol{R(\psi)R(\theta)R(\phi)} = \\ &= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \\ \boldsymbol{R_\Theta} &= \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \end{aligned} \quad (1)$$

B. Quadcopter Kinematics

As the quadcopter is a sub-actuated system (only four actuators to control 6 DOF) there is a need to redefine the

actuation variables. To do this we will consider the four basic movements:

- **Throttle** ($U_1[N]$) – Force related with a linear acceleration along Z_B-axis.
- **Roll** ($U_2[N.m]$) – Force related with an angular acceleration along X_B-axis.
- **Pitch** ($U_3[N.m]$) – Force related with an angular acceleration along Y_B-axis.
- **Yaw** ($U_4[N.m]$) – Force related with an angular acceleration along Z_B-axis.

The differential kinematics equation of a generic 6 DOF rigid body is given by Eq. (2), where $\boldsymbol{V^B} = (u, v, w)[m.s^{-1}]$ is the linear velocity and $\boldsymbol{\omega^B} = (p, q, r)[rad.s^{-1}]$ is the angular velocity, both in the B-Frame.

$$\dot{\boldsymbol{\Gamma}}^G = \boldsymbol{R_\Theta} \cdot \boldsymbol{V^B} \tag{2a}$$

$$\dot{\boldsymbol{\Theta}}^G = \boldsymbol{T_\Theta} \cdot \boldsymbol{\omega^B} \tag{2b}$$

C. Quadcopter Dynamics

The dynamics of a 6 DOF rigid-body takes into consideration the mass of the body $m[kg]$ as well as its inertia matrix $\boldsymbol{I}[Nms^2]$. Therefore, the dynamics of the quadcopter can be described in Eq. (3), using Newton's Law and Newton-Euler's Law.

$$m\ \boldsymbol{eye(3)} \cdot \dot{\boldsymbol{V}}^B + \boldsymbol{\omega}^B \times (m\boldsymbol{V}^B) = \boldsymbol{F}^B \tag{3a}$$

$$\boldsymbol{I} \cdot \dot{\boldsymbol{\omega}}^B + \boldsymbol{\omega}^B \times (\boldsymbol{I}\boldsymbol{\omega}^B) = \boldsymbol{\tau}^B \tag{3b}$$

where $\boldsymbol{eye(3)}$ is an identity 3×3 matrix, $\boldsymbol{F}^B = (F_x, F_y, F_z)[N]$ is a vector with the force applied in $(x, y, z)^B$ direction on the B-Frame and $\boldsymbol{\tau}^B = (\tau_x, \tau_y, \tau_z)[N.m]$ is a vector with the torques applied in in each axis $(x, y, z)^B$. In this case, adding the gravity forces and the actuation forces, we have

$$\boldsymbol{F}^B = \boldsymbol{R_\Theta}^T \cdot [0\ 0\ -mg]^T + [0\ 0\ U_1]^T \tag{4a}$$

$$\boldsymbol{\tau}^B = [U_2\ U_3\ U_3]^T \tag{4b}$$

The set of Eqs. (2) to (4) define the full mathematical model for the quadcopter.

D. Vision in the Loop

The basics of computer vision is based on the pinhole camera model. This model performs a transformation from 3D points ($\tilde{\boldsymbol{X}}[m]$) into 2D image points ($\tilde{\boldsymbol{u}}[pixels]$), both in homogeneous coordinates.

$$\lambda\tilde{\boldsymbol{u}} = \boldsymbol{\Pi}\tilde{\boldsymbol{X}} \Leftrightarrow \lambda\tilde{\boldsymbol{u}} = \boldsymbol{K} \cdot \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \end{bmatrix} \cdot \tilde{\boldsymbol{X}} \tag{5}$$

where λ is a scale factor, the $\boldsymbol{\Pi}_{3\times4}$ is the projection matrix, build from the intrinsic parameters matrix $\boldsymbol{K}_{3\times3}$ and the extrinsic parameters (rotation $\boldsymbol{R}_{3\times3}$ and translation $\boldsymbol{T}_{3\times1}$). The intrinsic parameters are obtained through a calibration process, which is described in [20].

For the matters of this work, we need to calculate $\boldsymbol{R}$ and $\boldsymbol{T}$, i.e. finding the pose of the camera, from the previously known $\boldsymbol{K}$ and a set of correspondences $\tilde{\boldsymbol{u}}$–$\tilde{\boldsymbol{X}}$. This is known as Perspective-n-Point (PnP) problem ([21]).

E. Control

The Proportional-Integral-Derivative (PID) controller is used within this thesis. It is known for its simplicity and effectiveness. It is described mathematically in Eq. (6) and it is schematized in Fig. 2.

$$u(t) = K_p\ e(t) + K_i \int_0^t e(\tau)d\tau + K_d\ \frac{d}{dt}e(t) \tag{6}$$

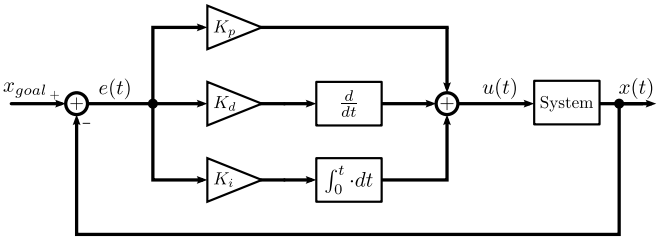


Fig. 2: Block diagram for PID controller

The way the PID controller works is not covered here, for simplicity.

III. VISION-BASED AUTONOMOUS CONTROL

The vision block will operate based on the detection of a specific AR plane marker (or set of markers), that will be identified as the target. As the markers board can be easily made, by simply printing a sheet paper, this procedure is very easy to do. The detection of the camera's pose is based on the previously known geometry and sizes of the markers, by trying to find the link between the observations and the physical constraints, defined by the markers geometry. The frames within this thesis are described in Fig. 3.

A. ArUco

To perform a vision-based detection of the marker, a library called ArUco was used ([17]).

ArUco works with a special type markers, like the example in Fig. 4. It can be seen as a 7×7 boolean matrix, with the outer cells filled with black color (which makes a perfect square, easy to find with image processing). The remaining 5×5 matrix is a 10-bits coded ID (up to 1024 different IDs), where each line has only 2 bits of information out of the 5 bits, with the other 3 being used for error detection. These extra 3 bits add asymmetry to the markers, which allow a unique orientation detection for most of the markers.

For detecting the marker, the following steps are taken:

1) Converting color image to gray image
2) Apply adaptive thresholding
3) Detect contours
4) Detect rectangles
   a) Detect corners
   b) Detect linked corners
   c) Consider only figures with four connected corners
5) For all detected (possible) markers

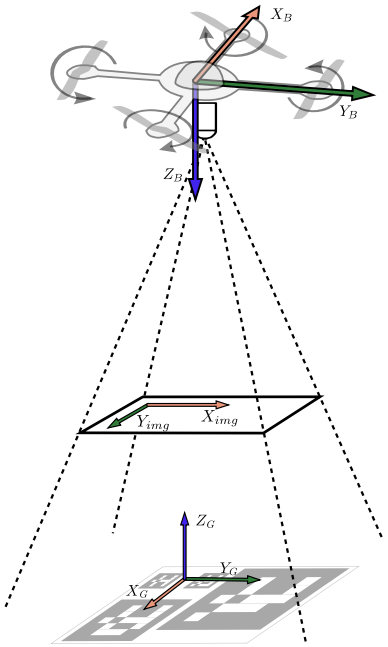


Fig. 3: Gathering all frames
B-Frame (quadcopter's body), img-Frame (camera's image) and G-Frame (ground)

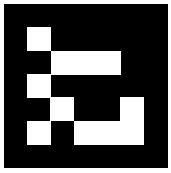


Fig. 4: ArUco marker example (id number 201)

a) Calculate homography (from corners)
b) Threshold the area using Otsu, which assumes a bimodal distribution and finds the threshold that maximizes the extra-class variance while keeping a low intra-class variance.
c) Detect and identify a valid marker, and if not detected tries the four rotations

6) Consider only valid markers
7) Detect extrinsic parameters

The extrinsic parameters are calculated with the help of an Open Source Computer Vision (OpenCV) function: `solvePnP()` ([21]). For each marker, the four corners in the image and their respective 3D coordinates are provided to the algorithm, as well as the calibrated $\boldsymbol{K}$, in order for the algorithm to estimate the $\boldsymbol{R}$ and $\boldsymbol{T}$.

B. Vision-Based Pose Estimation Algorithm (VBPEA)

The original ArUco is only able to deal with similar markers. In order to have multiple marker with different sizes, this library got new improvements:

- increase of the precision on calculating the camera pose, due to the increased number of markers used.
- the pose is calculated based on one reference point, instead of multiple markers points;
- the pose provided by the `solvePnP()` function is given WRT the reference frame;

This block performs a good detection of the markers and the height of the quadcopter. However it is not good at estimating the horizontal components (X, Y).

C. Vision-Based Horizontal Estimation Algorithm (VBHEA)

Because of mathematical issues regarding the (X, Y) estimation when the camera is above a horizontal marker, this new algorithm was developed. It considers the rotation matrix for a perfect horizontal image, thus simplifying the Eq. (5) into the Eq. (7).

$$\lambda\tilde{\boldsymbol{u}} = \boldsymbol{K}\cdot\begin{bmatrix}\boldsymbol{R} & -\boldsymbol{RT}\end{bmatrix}\cdot\tilde{\boldsymbol{X}} \Leftrightarrow \boldsymbol{T} = \boldsymbol{X} - \boldsymbol{R}^{-1}\boldsymbol{K}^{-1}\lambda\tilde{\boldsymbol{u}} \Leftrightarrow$$
$$\Leftrightarrow \begin{bmatrix}Tx\\Ty\\Tz\end{bmatrix} = \begin{bmatrix}X\\Y\\Z\end{bmatrix} - \begin{bmatrix}c_\psi & -s_\psi & 0\\ s_\psi & c_\psi & 0\\ 0 & 0 & 1\end{bmatrix}^{-1}\begin{bmatrix}fx & \gamma & Ox\\ 0 & fy & Oy\\ 0 & 0 & 1\end{bmatrix}^{-1}\lambda\begin{bmatrix}u\\v\\1\end{bmatrix} \quad (7)$$

where the constraint $T_z = Z^*$ applies, i.e. the current T_z must be equal to the Z^* given by the VBPEA, thus implying $\lambda = Z - Z^*$.

D. Control Algorithms

By gathering the data provided by VBPEA and VBHEA we have the full pose of the quadcopter. The VBPEA is responsible for detecting the markers on the image and compute the Z coordinate of the camera. Within the process of identifying the marker, the algorithm computes also the yaw angle of the camera. These two variables (Z, Yaw) will be inputs for the VBHEA, as well as the image.

The VBHEA takes the processed image and applies Eq. (7), thus providing the (X, Y) coordinates of the camera. Because the VBPEA provides good estimates for (Z, Yaw) and the VBHEA provides better estimates for (X, Y), the group of these two algorithms behaves as one block.

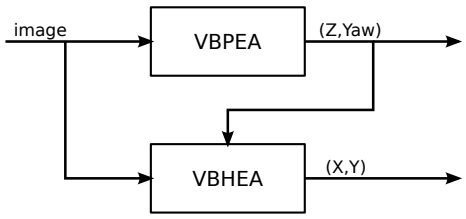


Fig. 5: Integration of VBPEA with VBHEA in order to obtain the (X, Y, Z, Yaw) on hovering

These gathered blocks are providing a full pose $(X, Y, Z, \phi = 0, \theta = 0, \psi)$, assuming the quadcopter is perfectly horizontal. This allows better stability and therefore allows to a better and smoother control.

E. Hovering Algorithm

This algorithm uses the $(X, Y, Z, Yaw, Roll, Pitch)$ from the previous blocks and applies a PID controller to the (X, Y) position of the quadcopter and stabilize the robot above the target, by keeping a constant height from it. Due to the fact that the error may be not differentiable, the derivative gain is applied directly to the feedback value, instead of the error.

In order to apply a PID controller, a destination point $\boldsymbol{X}_d = (X, Y, Z)_d$ is defined, the one above the target on which the quadcopter will be hovering. With the purpose of aligning the quadcopter reference frame with the world frame, a yaw rotation is applied to the the destination point (originally in the G-Frame), thus considering the destination point WRT the B-Frame.

$$\boldsymbol{X}_d^B = \boldsymbol{R}(\psi)\boldsymbol{X}_d^G \tag{8}$$

Considering the current position of the quadcopter $\boldsymbol{X} = (X, Y, Z)$, a PID controller is applied on the position error $\boldsymbol{E} = \boldsymbol{X}_d - \boldsymbol{X}$. As the control variables are Thrust, Roll, Pitch, Yaw (trpy), the controller defined in Equations (9) provides the control for the commanded roll ($\hat{\phi}$) and commanded pitch ($\hat{\theta}$) movements.

$$\hat{\theta} = K_{p_x} E_x - K_{d_x} \frac{dX}{dt} + K_{i_x} \int_0^t E_x \; dt \tag{9a}$$

$$\hat{\phi} = K_{p_y} E_x - K_{d_y} \frac{dY}{dt} + K_{i_y} \int_0^t E_y \; dt \tag{9b}$$

A PID controller is also applied on yaw error $E_{yaw} = \psi_d - \psi$, in order to maintaining the quadcopter aligned with the marker. This is useful for implementing on a real quadcopter to land on a real docking station, as it probably matters the quadcopter orientation when landed. The commanded yaw ($\hat{\psi}$) is calculated in Eq. (10).

$$\hat{\psi} = K_{p_{yaw}} E_{yaw} - K_{d_{yaw}} \frac{d\psi}{dt} + K_{i_{yaw}} \int_0^t E_{yaw} \; dt \tag{10}$$

For maintaining the quadcopter's height Z, a PID controller is also applied to the height error E_z from the previously defined error vector $\boldsymbol{E}$, thus giving a command thrust $\hat{Z}$.

$$\hat{Z} = K_{p_z} E_z - K_{d_z} \frac{dZ}{dt} + K_{i_z} \int_0^t E_z \; dt \tag{11}$$

The Fig. 6 shows the block diagram of the implementation of this algorithm.

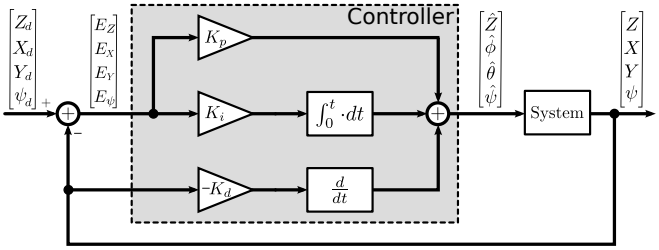


Fig. 6: PID controller applied to the detected pose

F. Landing Algorithm

This algorithm consists on using the PID controller on the Z component of the position, by incrementally changing the destination point for hovering, by decreasing its Z coordinate, after some conditions are verified. This incremental change on the desired Z_d helps maintaining stability, by continuously checking those conditions.

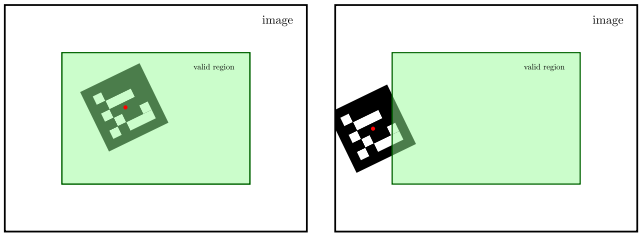
In order for the algorithm starts to decrease the Z_d, the following conditions must be checked:

- variability of the $roll$ and $pitch$ command for the quadcopter;
- position of the reference point within the center of the image;
- image change velocity;
- height of the quadcopter.

To evaluate the commands variability, a low pass filter is applied to the $roll$ and $pitch$. The discrete low filter is applied as in Eq. (12). If the variability is below a threshold, the quadcopter is considered to be stable.

$$y_i = \alpha x_i + (1 - \alpha) y_{i-1} \tag{12}$$

To evaluate if the reference point is centered within the image, we project the 3D reference point to the image, using Eq. (5), and check if the projection is inside a rectangular region in the center of the image. Fig. 7 represents this validation. The only validated point is the reference point, to avoid further markers getting an invalid tag when they are out (or in the limit) of the image.



(a) Reference point inside the valid region, thus valid

(b) Reference point outside the valid region, thus invalid

Fig. 7: Validating if reference point is in the image center region
The red dot represents the reference point, in this example coinciding with the marker's center

The evaluation of the image change velocity consists on calculating the velocity of the reference point, in $[pixels.s^{-1}]$ (in Eq. (13)), which will provide information about how much the reference point is moving, within the image. This information will help on preventing the image to get too much blur effect. When the quadcopter is higher, the velocities within the image will be slower and it will be tagged as valid for landing. This effect is desired because if the quadcopter is high enough, even if it is not completely stable from the image point of view, the blur effect is smaller than when the quadcopter is lower. When the velocity is below a certain threshold, the image is considered to be stable.

$$v_i = \frac{\|\boldsymbol{u_i} - \boldsymbol{u_{i-1}}\|}{\Delta t} = \frac{\left\| \begin{bmatrix} u \\ v \end{bmatrix}_i - \begin{bmatrix} u \\ v \end{bmatrix}_{i-1} \right\|}{\Delta t} \tag{13}$$

The Z_d will be decreased only if its height is close from the Z_d, by less than a threshold Z_{thresh}.

It was experimentally observed that as soon as any part of the robot touches the ground, this will affect the IMU which

will make the autopilot to respond. This results in unexpected behaviour and instability. The solution for safely landing is consists on turning off the rotors when the quadcopter is close enough of the ground. On experimental trials, it was tested that the quadcopter can be turned off safely even if it is 15cm from the ground. Having this measure as the maximum, the landing algorithm is ready to turn off the rotors when it is 5cm from the ground. This height guarantees that it will not have that unexpected behaviour and still lands safely.

If all the previous conditions are verified, the quadcopter is considered able to go down; then the Z_d is decreased by a step Z_{step}. This process will continue until the quadcopter is lower than a limit distance from the ground. When it reaches the limit, the quadcopter's rotors are turned off.

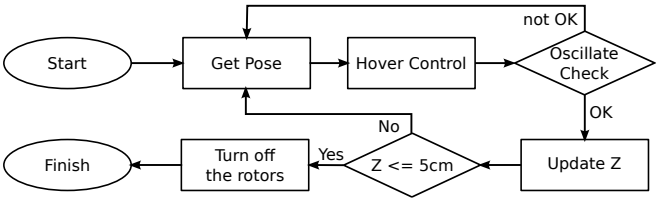


Fig. 8: Landing algorithms flowchart

G. Integration and Implementation

The simulation is performed based on a physics simulator called Gazebo[22]. Gazebo is a multi-robot simulator for indoor or outdoor environments and it is capable of simulating a population of robots, sensors and objects, in a three-dimensional world. It generates both realistic sensor feedback (with noise included) and physically plausible interactions between objects (it includes an accurate simulation of rigid-body physics). Since this simulator operates on Robot Operating System (ROS) it makes it easier to implement all the rest.

The use of this simulator in the thesis implies the modelling of a quadcopter robot, an IMU sensor on the robot, a camera assembled on the bottom of the quadcopter, as well as the interacting world which must have some markers included. One of the advantages of using the simulator is the similarity to the real world application (conceptually the same), thus we expect to use the very same architecture during real tests and simulations tests.

The full architecture is composed by this set of nodes:

- **usb_cam** raw data from the USB camera
- **image_view** image visualization only
- **Gazebo** physics simulator with a graphical interface
- **quad_sim** spawn a virtual quadcopter with a downwards camera on Gazebo and model its dynamics.
- **Vision-Based Pose Estimation Algorithm (VBPEA)** provides the height and yaw angle of the quadcopter
- **Vision-Based Horizontal Estimation Algorithm (VBHEA)** provides (X, Y) location of the quadcopter
- **pose_fusion** fusing the information about the estimated pose.
- **quad_hover_control** a PID controller on the position of the quadcopter.
- **quad_landing_control** makes the decisions and defines what should be the destination point for the quad_hover_control.

All these nodes work together as in Fig. 9.

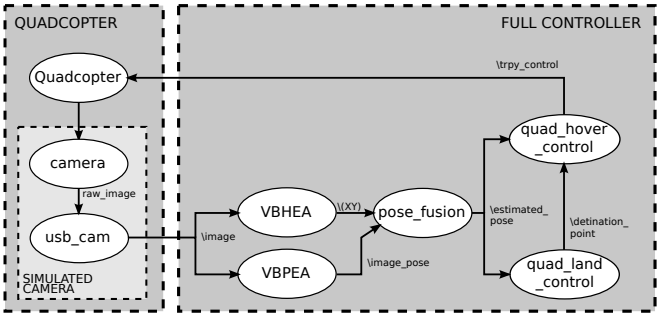


Fig. 9: Full architecture gathering all algorithms

IV. SIMULATION SETUP

The experimental setup was developed in two separate frameworks. The first one was a development exclusively on simulation. The hardware was modeled in software and then in a simulator. The second framework consists on the real quadcopter, equipped with a Pandaboard and a camera, and a set of markers. Both frameworks were developed as similar as possible, to maximize the compatibility of the developed methods between both frameworks.

The quadcopter used in this thesis was the UAVision Quadcopter UX-4001 mini, developed by UAVision[23] and equipped with the Paparazzi autopilot ([24]). It is also equipped with some essential sensors (gyros, accelerometers, magnetometer) and some other sensors not used in this thesis (GPS, Zigbee, WiFi). The Paparazzi autopilot is responsible for stabilizing the quadcopter and performs a low level control of all the four rotors, so it is possible to control the desired thrust, yaw rotation, roll and pitch. This allows a better control interface for the end user.

Fig. 10: UAVision Quadcopter UX-4001 mini

A. Experiments

In order to classify the algorithms developed, this series of tests were conducted.

1) Vision-Based Pose Estimation Algorithm: In order to determine how the number of markers detected affects the output, a test bench was assembled (Fig. 11). It consists on a real camera aligned with the marker(s), by making the camera XY-frame parallel to the markers XY-frame, and by keeping a steady (X, Y) coordinates on the camera, while varying the distance from the marker(s), i.e., measuring the Z coordinate while keeping $X = Y = 0$.

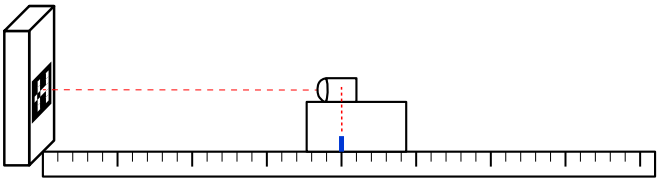


Fig. 11: Test bench for the VBPEA experiment

2) Comparing VBPEA with a Laser Range Finder (LRF): In this experiment, a real quadcopter with an attached LRF was used. In order to compare the VBPEA estimations with the data provided by the LRF, a downwards camera was also on board of the quadcopter. The LRF was measuring the height of the quadcopter, the same variable provided by the VBPEA we want to test.

3) Vision-Based Horizontal Estimation Algorithm: As this is one of the crucial algorithms, its quality must be verified. This set of tests consist on fixing the real camera at a fixed Z coordinate and varying the X or Y coordinate (one of the horizontal coordinates), while maintaining the other coordinate equal to zero (Fig. 12).

Three different trials were conducted, with different markers boards, by giving the camera a horizontal displacement during each one.

- Trial 1 – Using two fixed, 4 cm sided markers, aligned vertically (in 2D);
- Trial 2 – Using two fixed, 4 cm sided markers, aligned horizontally (in 2D);
- Trial 3 – Using the full markers board (4 markers, different sizes), aligned horizontally (in 2D).

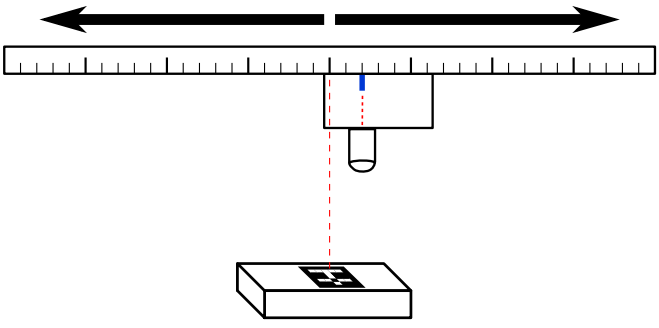


Fig. 12: Test bench for the VBPEA experiment

4) Visual-Based Horizontal Position Estimation: To demonstrate the problem of finding the (X, Y) coordinates, a test was made in the simulator. This test consists on simulating the quadcopter with a downwards camera and a markers board, running both VBPEA and VBHEA, and comparing both outputs, as well as the ground truth information, possible in the simulation environment.

5) Inbuilt quadcopter controllers: This test consists on introducing a step input and observing the controller's output. For classifying roll and pitch, only the roll controller was observed and similarity for pitch is assumed.

6) Case 1 - Steady Hovering: In order to test how the quadcopter behaves when hovering is intended, the Gazebo was used. For the simulation, the following steps were followed by the presented order:

- Start Gazebo;
- include world model in the Gazebo;
- spawn the quadcopter, with a downwards camera;
- start VBPEA;
- start VBHEA;
- start hovering control.

7) Case 2 - Landing on a Steady Target: This experiment is the final experimentation, with all developed algorithms running. This one is similar to the previous experiment from Section IV-A.6, with one more process running. For the simulation, the following steps were followed by the presented order:

- Start Gazebo;
- include world model in the Gazebo;
- spawn the quadcopter, with a downwards camera;
- start VBPEA;
- start VBHEA;
- start hovering control;
- start landing control

As it was developed, this experiment is based on sequential hovering processes, as the objective is to hover the target, but with different altitudes along time. After stabilizing the first time, the quadcopter starts to smoothly descend its altitude.

V. EXPERIMENTAL RESULTS

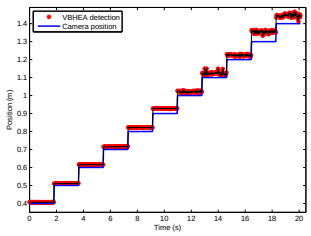
A. Results from the Experiments

1) Vision-Based Pose Estimation Algorithm: The analysis of the results in Figure 13 and 14 demonstrate that the number of detected markers relates with the error on the VBPEA output; the more detected markers, the less is the error. The Figure 15 shows the comparison between the detected error in each trial, where it is clear that the estimation is better when more markers are used.

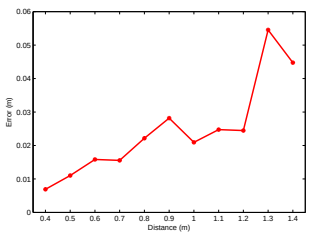
This results confirm that the VBPEA performs very well. Even with just one marker, the Z position error is less than 5% of the the camera Z coordinate (in Figure 15), which is enough for the intended application.

2) Comparing VBPEA with a Laser Range Finder: In order to compare the VBPEA with another sensor performing a similar task, some tests using a LRF on board the quadcopter were conducted, and the results are expressed in Figure 16. Because the quadcopter was commanded manually, it was difficult to keep the marker within the image and sometimes, as it occurred at 6-7 s, the VBPEA was not able to provide any data.

Although the real data is noisier than the previous tests in Section V-A.1 (as expected), the obtained VBPEA output is still acceptable for the intended purpose, with errors smaller

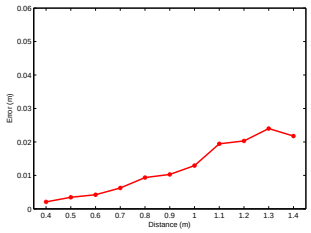


(a) Comparison between the distance of the camera and the distance perceived by the VBPEA

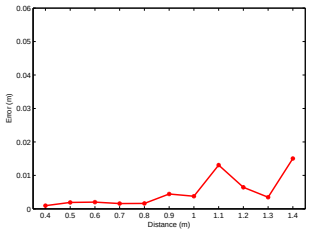


(b) Absolute value of the error for the VBPEA detection

Fig. 13: Trial for testing the VBPEA detection, using one marker

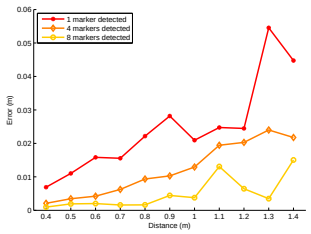


(a) Absolute value of the error for the VBPEA detection

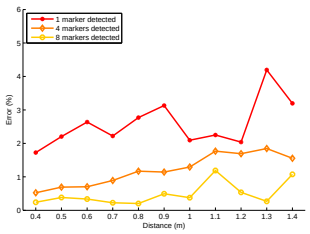


(b) Absolute value of the error for the VBPEA detection

Fig. 14: Trials for testing the VBPEA detection, using four and eight markers

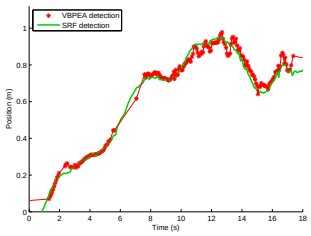


(a) Comparison between absolute error detection

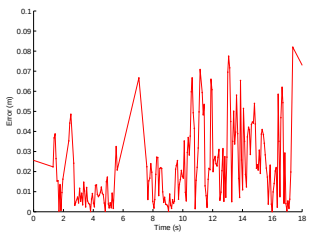


(b) Relative error detection, to the distance of the camera

Fig. 15: Comparison between error detection, using one, four or eight markers



(a) Comparison between the distance perceived by the LRF and the VBPEA



(b) Absolute value of the difference between the LRF and the VBPEA detections

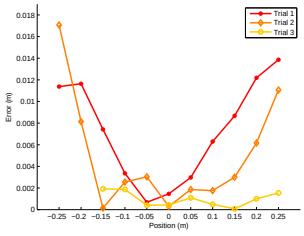
Fig. 16: Comparison between LRF and VBPEA

than 10 cm. This error corresponds to less than 10% of the detected height.

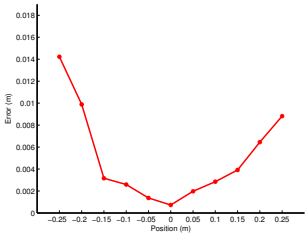
This results allow to conclude that the VBPEA is able to be used within a real application.

3) Vision-Based Horizontal Estimation Algorithm: In this experiment, real data from a real camera were used. Three different trials were conducted, with different markers boards.

The Figure 17 show some interesting results by comparing the error among the three trials. All trials show some common behaviour, that the error gets bigger as the camera gets further from the center, and when more markers are visible the error gets smaller. These behaviours were already expected, as in the experiment in Section V-A.1.



(a) Absolute errors

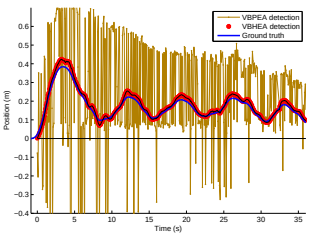


(b) Average of errors

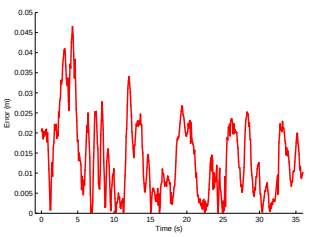
Fig. 17: Comparison between the different trials

In conclusion, this algorithm's output is reliable and, within these trials, it has errors above 7%. More results about the VBHEA are addressed in the next section.

4) Visual-Based Horizontal Position Estimation: The results observed in Figure 18 show that the horizontal coordinates from the VBPEA are very noisy and therefore they are useless for their intended purpose. On the other hand, the output from the VBHEA is more accurate. The absolute error for the VBHEA shows that this method provides data with error lower than 5 cm. This error does not depend only on the distance of the marker but also on the angle of the quadcopter, with the latter being the major responsible for the error. Such fact may be observed during the changes of direction of the quadcopter, i.e. on the peaks of the plot, where the error is bigger.



(a) Comparing X coordinate

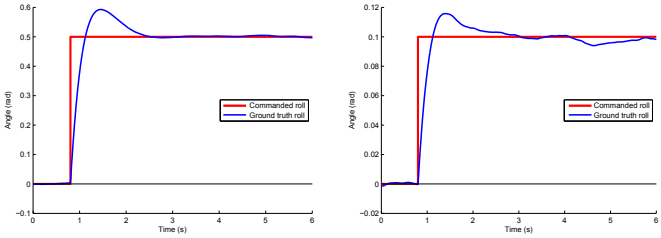


(b) Absolute error in VBHEA's X coordinate

Fig. 18: Comparison between the outputs from the VBPEA, VBHEA and ground truth from the simulator

5) Inbuilt quadcopter controllers: The controller stated in Section III-E uses the approximation that the autopilot controller is perfect. However, when testing the simulated quadcopter on Gazebo, the results show that it is not true at all.

By analysing Figure 19 is is possible to define some features about the controller.

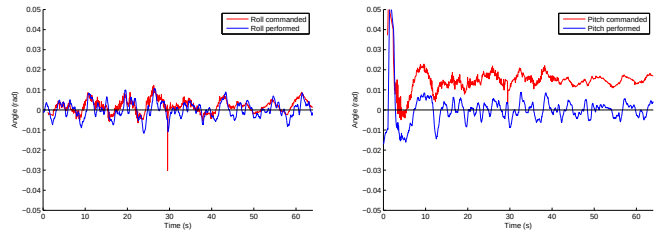


(a) Response to a step = 0.5 rad **(b)** Response to a step = 0.1 rad

Fig. 19: Step response of the simulated quadcopter autopilot, for the roll movement

- The rising time is ˜250 ms. This value is not very low, considering the kind of system it is applied, but it is much faster than the settling time.
- The settling time is ˜1800 ms. This value is very high and reduces the performance of the system response. On the other hand, it is useful to have a lower performance on simulation because if the developed controllers work in a noisy simulation environment, they will probably also work on real systems.
- The overshoot is ˜20%. This overshoot value is normal and it helps on setting the linear velocity of the quadcopter faster.
- The steady state value is very close to the desired value, having some noise introduced in order to have a simulation more similar to the real system.

6) Case 1 - Steady Hovering: This study case showed positive results. The quadcopter was able to hover the target without loose the markers from the image. The Figure 20 shows some interesting results.



(a) Comparing roll **(b)** Comparing pitch

Fig. 20: Comparing the commanded roll and pitch with the real output

It is notorious a constant drift within the pitch that is being continuously corrected by hovering controller. This simulated drift aims on creating more realistic data, as this kind of systematic error exists in real sensors. Since the quadcopter was hovering, this is a good result; the developed controller is able to successfully compensate the existent drift.

The differences between the input and the output variables were expected, from the results in Section V-A.5. Although the output does not replicate de intended values, it still follows the general commands, thus being able to control the quadcopter.

7) Case 2 - Landing on a Steady Target: This study case tested the landing process of a simulated quadcopter and it showed some good results. The visualization of this simulation is available at [25]. The quadcopter landed on the target and the reference point was visible within the image. As the camera has a low angle of view (˜45°), in order to being possible to see the reference marker, the camera must be within an area with less than 5 cm radius. As the reference point was visible within the image, this means the quadcopter landed on the right place, with a 5 cm error.

The quadcopter landed smoothly and it took less than one minute to do it. In Fig. 21 it is possible to see the vertical trajectory followed by the quadcopter.

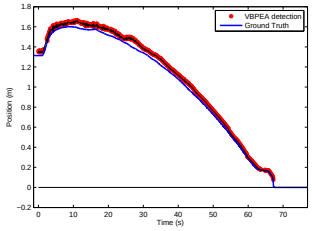


Fig. 21: Height of the quadcopter during the landing process

A movie about this experiment can be viewed in [25].

VI. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this thesis, a method for autonomous landing a quadcopter was developed. It was demonstrated that it is possible to implement these methods in a non calibrated and unknown environment, as long as it is possible to use a markers board within the scenario. It was also proved that it is possible to perform an autonomous landing with onboard sensors only.

An algorithm for full autonomous hovering and landing was developed. For the quadcopter's pose estimation, a Vision-Based Pose Estimation Algorithm, based on markers observation, was implemented and improved, by considering the fixed orientation of the on board camera. This method demonstrated dependable results, good for the controlling algorithms.

In order to prove the concepts in this thesis, a virtual simulation environment was also developed, using ROS tools. This implementation not only allowed to test these methods but it will also be useful for further investigations within the fields of UAVs robotics. The simulator also includes a model for a quadcopter. In the simulator, a successful landing was taken, thus producing the desired achievement.

In conclusion, the following items were developed:

- Visual-based pose estimation algorithm development and implementation;
- Identification of the required software for onboard controlling, for future development;
- Development of a simulator with a controllable quadcopter;
- Autonomous hovering algorithm;
- Autonomous landing algorithm.

Although the autonomous landing was not performed on a real quadcopter, the developed algorithms and implementations are able to solve the initial addressed problem, which is the development of a quadcopter autonomous landing algorithm (in this case, a simulated quadcopter).

B. Future Work

There are several possible future developments within this work.

The most obvious future development is the implementation on a real quadcopter. The communications bridge between the Pandaboard and the Paparazzi must be considered.

The VBPEA and the VBHEA could become one single algorithm in order to improve the performance of the pose estimation.

It is possible to improve the simulator by adding a moving station with a markers board, to help further development on following and landing on moving targets.

For the pose estimation algorithms, the information from the IMU could be added, possibly resulting in a more accurate pose estimation and improving noise reduction. For the (X, Y) estimation from the VBHEA, a model including the small angle would also improve the estimations.

Using the image, a method to track the changes within the image could be developed, thus estimating the camera motion. This could be useful for the hovering application. This method is also known as optical flow.

REFERENCES

[1] R. Mahony and V. Kumar, "Aerial robotics and the quadrotor [from the guest editors]," Robotics Automation Magazine, IEEE, vol. 19, no. 3, pp. 19–19, September 2012.

[2] Institute for Systems and Robotics, Instituto Superior Técnico. (2013) The Rescue Project. Consulted on October 2013. [Online]. Available: http://rescue.isr.ist.utl.pt/

[3] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," Robotics Automation Magazine, IEEE, vol. 19, no. 3, pp. 20–32, September 2012.

[4] H. Lim, J. Park, D. Lee, and H. J. Kim, "Build your own quadrotor: Open-source projects on unmanned aerial vehicles," Robotics Automation Magazine, IEEE, vol. 19, no. 3, pp. 33–45, September 2012.

[5] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," Robotics Automation Magazine, IEEE, vol. 19, no. 3, pp. 46–56, September 2012.

[6] A. Franchi, C. Secchi, M. Ryll, H. H. Bulthoff, and P. R. Giordano, "Shared control: Balancing autonomy and human assistance with a group of quadrotor uavs," Robotics Automation Magazine, IEEE, vol. 19, no. 3, pp. 57–68, September 2012.

[7] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation: Safe and efficient load manipulation with aerial robots," Robotics Automation Magazine, IEEE, vol. 19, no. 3, pp. 69–79, September 2012.

[8] A. King, "Inertial Navigation -- Forty Years of Evolution," GEC review, vol. 13, no. 3, pp. 140–149, 1998.

[9] R. D'Andrea, A. Schollig, M. Sherback, and S. Lupashin, "A simple learning strategy for high-speed quadrocopter multi-flips," in Robotics and Automation (ICRA), 2010 IEEE International Conference on, 2010, pp. 1642–1648.

[10] ©Vicon Motion Systems Ltd. UK. (2013) Consulted on October 2013. [Online]. Available: http://www.vicon.com/

[11] D. Mellinger, M. Shomin, and V. Kumar, "Control of quadrotors for robust perching and landing," in Proc. Int. Powered Lift Conf, 2010, pp. 119–126.

[12] I. Sa and P. Corke, "Estimation and Control for an Open-Source Quadcopter," in Proceedings of the Australasian Conference on Robotics and Automation 2011, 2011.

[13] D. Mellinger, N. Michael, and V. Kumar, "Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors," The International Journal of Robotics Research, vol. 31, no. 5, pp. 664–674, 2012.

[14] J. Friis, E. Nielsen, R. F. Andersen, J. Boending, A. Jochumsen, and A. Friis, "Autonomous Landing on a Moving Platform," Control Engineering, 8th Semester Project, Aalborg University, Denmark, 2009.

[15] P. R. Giordano, H. Deusch, J. Lächele, and H. Bülthoff, "Visual-vestibular feedback for enhanced situational awareness in teleoperation of UAVs," in Proc. of the AHS 66th Annual Forum and Technology Display, 2010.

[16] R. D'Andrea, "The Astounding Athletic Power of Quadcopters," http://www.ted.com/talks/raffaello_d_andrea_the_astounding_athletic_power_of_quadcopters.html, TED Talk, June 2013, consulted on October 2013.

[17] Aplicaciones de la Visión Artificial from Universidad de Córdoba. (2013, April) ArUco: a minimal library for Augmented Reality applications based on OpenCV. Consulted on October 2013. [Online]. Available: http://www.uco.es/investiga/grupos/ava/node/26

[18] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on. IEEE, 1999, pp. 85–94.

[19] T. Bresciani, "Modelling , Identification and Control of a Quadrotor Helicopter," Ph.D. dissertation, Lund University, 2008.

[20] ROS.ORG. (2013) camera_calibration ROS Package. Consulted on October 2013. [Online]. Available: http://www.ros.org/wiki/camera_calibration

[21] F. Moreno-Noguer, V. Lepetit, and P. Fua, "Accurate Non-Iterative $O(n)$ Solution to the PnP Problem," in Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. IEEE, 2007, pp. 1–8.

[22] Gazebo. (2011) Consulted on October 2013. [Online]. Available: http://gazebosim.org/

[23] UAVision. (2013) Consulted on October 2013. [Online]. Available: http://uavision.com/

[24] P. Project. Consulted on October 2013. [Online]. Available: http://paparazzi.enac.fr/wiki/Autopilots

[25] Tiago Gomes Carreira, "Master Thesis Extra Files," Master's thesis, Instituto Superior Técnico, October 2013. [Online]. Available: http://web.ist.utl.pt/tiago.carreira/thesis

[26] ROS - Robotic Operative System. (2013) Consulted on October 2013. [Online]. Available: http://www.ros.org

[27] H. I. T. L. H. L. at the University of Washington, H. L. at the University of Canterbury, New Zealand, and A. I. Seattle. (2013) ARToolKit. Consulted on October 2013. [Online]. Available: http://www.hitl.washington.edu/artoolkit/

[28] Pandaboard. Consulted on October 2013. [Online]. Available: http://pandaboard.org/

[29] Wikipedia. (2010, March) List of Unmanned Aerial Vehicles. Consulted on October 2013. [Online]. Available: http://en.wikipedia.org/wiki/List_of_unmanned_aerial_vehicles

[30] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, An Invitation to 3-D Vision: From Images to Geometric Models. SpringerVerlag, 2004.

[31] S. Bouabdallah, "Design and Control of Quadrotors with Application to Autonomous Flying," Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, 2007.

[32] H. R. D. d. Silva, "Controlo de Formações em Veículos Aéreos não Tripulados," Master's thesis, Instituto Superior Técnico, June 2012.

[33] D. Koks, Explorations in Mathematical Physics: The Concepts Behind an Elegant Language. Springer, 2006.

[34] J. B. Kuipers, Quaternions and Rotation Sequences. Princeton university press, Princeton, 1999.

[35] S. Suzuki and K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following," Computer Vision, Graphics, and Image Processing, vol. 30, no. 1, pp. 32–46, 1985, consulted on October 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/cvgip/cvgip30.html#SuzukiA85

[36] R. Szeliski, Computer Vision: Algorithms and Applications. Springer, 2011, consulted on October 2013. [Online]. Available: http://szeliski.org/Book/