Universidade de São Paulo–USP
Escola de Engenharia de São Carlos
Departamento de Engenharia Mecânica
Programa de Pós-Graduação em Engenharia Mecânica

Rômulo Teixeira Rodrigues

Aterrissagem Autônoma Baseada em Visão para Mini Quadrirrotor

São Carlos
2017

Rômulo Teixeira Rodrigues

Aterrissagem Autônoma Baseada em Visão para Mini Quadrirrotor

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo como parte dos requisitos para a obtenção do título de Mestre em Ciências, Programa de Engenharia Mecânica.

Área de concentração: Dinâmica de Máquinas e Sistemas

Orientador: Prof Assoc Marcelo Becker

São Carlos
2017

Rodrigues, Rômulo Teixeira
R696a Aterrissagem autônoma baseada em visão para mini quadrirrotor / Rômulo Teixeira Rodrigues; orientador Marcelo Becker. São Carlos, 2017.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Mecânica e Área de Concentração em Dinâmica das Máquinas e Sistemas -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2017.

1. Estimação de estado. 2. Controle de movimento. 3. Visão computacional. 4. Simulação para VANT. I. Título.

FOLHA DE JULGAMENTO

Candidato: **RÔMULO TEIXEIRA RODRIGUES**

Título da dissertação: "Aterrissagem autônoma baseada em visão para mini quadrirrotor"

Data da defesa: 25.07.2017

Comissão Julgadora:	**Resultado:**
Prof. Associado **Marcelo Becker** **(Orientador)** (Escola de Engenharia de São Carlos/EESC)	APROVADO
Prof. Dr. **José Reginaldo Hughes Carvalho** (Universidade Federal do Amazonas/UFAM)	APROVADO
Prof. Dr. **Guilherme Sousa Bastos** (Universidade Federal de Itajubá/UNIFEI)	APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Mecânica:
Prof. Associado **Gherhardt Ribatski**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Luis Fernando Costa Alberto**

Acknowledgements

I would like to thank:

- Prof. Marcelo Becker for mentoring and supporting me in this journey.
- Intelligent Robots and Systems Laboratory (IRSg) in Institute for Systems and Robotics (ISR/Lisbon) for kindly granting access to Crazyflie quadrotor and Optitrack motion capture system.
- Friends that shared their time, fish and coffee with me. A special thanks to my labmate Marco Arruda for his help and patience.
- Margareth, Rômulo, Rebeca, Helena and Ana Rita whose unconditional love helps me move forward.
- Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) for providing financial support for my research.

Resumo

T. Rodrigues, Rômulo. **Aterrissagem Autônoma Baseada em Visão para Mini Quadrirrotor.** 77 p. Dissertação de mestrado – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2017.

O problema abordado nesse trabalho de mestrado, pouso autônomo de quadricópteros baseado em visão, explora dois problemas distintos que têm recebido atenção da comunidade científica na última década: estimação de estado e controle do movimento. Em particular, para o problema de aterrissagem, a estimação de estado tem como objetivo estimar a translação e rotação (pose) do veículo em relação ao local de pouso. De acordo com o local do pouso, pode-se dividir os trabalhos em duas classes, nomeadamente baseados em alvos ou regiões de forte textura. O presente trabalho foca em soluções baseadas em alvos. Dessa forma, para resolver o problema, deve-se encontrar o alvo na imagem e estimar a posição da câmera em relação ao mesmo. Para obter a escala, é necessário conhecer as dimensões do alvo. O método é complementado com fluxo ótico baseado na homografia contínua. O fluxo ótico permite estimar a velocidade do veículo em qualquer cenário de forte textura. Os dois métodos e dados inerciais são combinados através do Filtro Estendindo de Kalman. O problema de controle do movimento consiste em dirigir o veículo de forma segura para o alvo. No escopo desta dissertação, um controle de de navegação baseado em linearização da realimentação e um controlador PID garante que o erro convirja para a origem. Simulações e dados reais são realizados, de forma que o funcionamento correto da solução é confirmada. Em particular, para um veículo aéreo de pequeno porte, o método proposto em malha fechada teve um erro quadrático médio de 0.15 m em relação ao local de pouso.

Palavras-chave: estimação de estado, controle de movimento, visão computacional, simulaçao para VANT.

Abstract

T. Rodrigues, Rômulo. **Vision Based Autonomous Landing for Mini Quadrotor**. 77 p. Master Thesis – São Carlos School of Engineering, University of São Paulo, 2017.

The problem addressed in this master dissertation, vision based autonomous landing of mini quadrotor, explores two distinct topics that have received considerable attention from the research community in the last decade: state estimation and motion control. For the landing problem, the state estimation task aims at estimating the pose of the vehicle with respect to the landing site. Concerning the landing site, two different strategies can be found in the literature, namely based on landmarks and high-textured flat regions. This work focuses on landmark based methods using a single onboard camera and inertial measurements. The solution relies on finding the landmark center on an image and estimating the pose in respect to the corresponding point in the world. In order to obtain the scale, the landmark dimensions must be known. The method is complemented using continuous homography optical flow, which allows estimating the velocity of the vehicle in any texture flat region. Both estimation and inertial data are fused within a filtering framework – Extended Kalman Filter. The motion control task consists in bringing the vehicle safely towards the landing target. In the scope of this work, a set-point controller based on feedback linearization and a proportional-integral-derivative (PID) controller ensures the error gently converges to the origin. Simulations and real world experiments are conducted to ensure the reliability of the method. In particular, for a small vehicle, the proposed visual closed loop autonomous landing system achieved $0.15\,m$ accuracy.

Keywords: state estimation, motion control, computer vision, UAV simulation.

Contents

List of Figures

List of Tables

Abbreviations

UAS	Unmanned aerial system
RPAS	Remotely piloted aircraft systems
GPS	Global Positioning System
IMU	Inertial measurement unit
EKF	Extended Kalman filter
PID	Proportional-integrative-derivative
VTP	Virtual target point
MoCap	Motion capture
ROS	Robot Operating System
GUI	Graphical User Interface
V-REP	Virtual Robot Experimentation Platform
3D	3-Dimensional
2D	2-Dimensional
DoF	Degrees of freedom
CG	Center of gravity

Chapter 1

Introduction

This chapter introduces the topic of the present dissertation: vision based autonomous landing of a mini-quadrotor. The discussion starts off in Section 1.1 highlighting the importance of unmanned aerial vehicles (UAV) in our society. Then, Section 1.2 introduces the problem to be tackled. Section 1.3 brings forward the proposed solution and contributions. Finally, Section 1.4 describes briefly the structure of this document.

Within the scope of this dissertation the words UAS - unmanned aerial system - and RPAS - remotely piloted aircraft system - have the same meaning as UAV.

1.1 Motivation

In the last years, unmanned aerial vehicles have drawn large interest not only in the airborne system community, but also among military, civilian, capital ventures and, last but not least, research institutes. Unmanned aircraft is the fastest growing market segment in the robotic industry (CHENG; KUMAR, 2008). Putting words into numbers, UAS fully integration with other airspace system will lead to an economic impact of $82.1 billion in the U.S. between 2015 and 2025, creating more than 100,000 jobs (JENKINS; VASIGHI, 2013). The remainder of this section addresses some UAV applications already employed in behalf of our society.

Forest Fire Detection

At the time that environment preservation grows as a global concern, the world keeps an eye on Brazil forest conservation policies. In this plot, Amazonas state plays an important role considering the Amazon rain forest covers about 98% of its land (BUTLER, 2014). Amidst other threats, fire alone has affected 620 km^2 of Brazilian forest per year (BUTLER, 2014). Detecting and localizing fire in its early stage is essential to diminish the overall damage that ensues forest fire. Merino, Ramiro & Ollero (2014) employ a fleet of UAS for automatic detection, verification, and localization of potential fire spots.

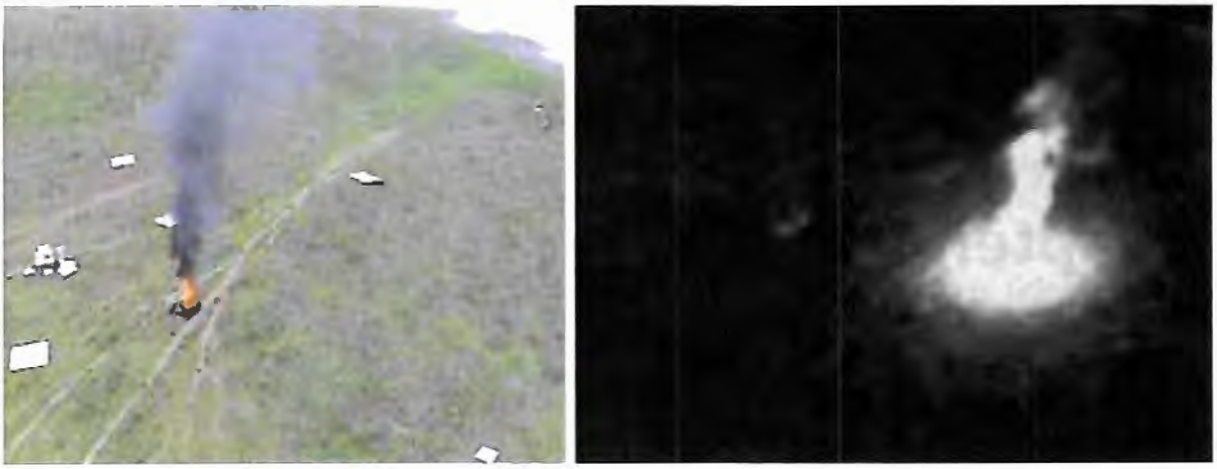


Figure 1: Fire detection. Visual (left) and infrared (right) image frames. Source: (MERINO; RAMIRO; OLLERO, 2014)

Vehicles are equipped with visual and infrared cameras, as shown in Fig. 1. The different wavelength images provide meaningful information for fire detection. As long as smoke does not affect the sensor, flames are easily identified with visual image. Meanwhile, the infrared device allows fire base recognition. The information provided by distinct vehicles in the fleet enhances the accuracy of the fire alarm system, preventing false alarms. Global positioning system (GPS) and inertial measurement unit (IMU) data are merged with digital terrain maps for geo-localization of the fire.

Precision Agriculture

Precision agriculture or precision farming aims at providing farmers an individual profile of the needs and state of each plant in the field. The benefits, which are many, include better product quality, adequate pest and fertilize management, and natural resources conservation (GRISSO et al., 2002).

Precision farming requires high resolution images. Artificial satellites partially fulfill the image demand for some precision farming applications such as crop grow and stress assessment (ZHANG; KOVACS, 2012). However, low revisiting frequency, good weather requirement, and high costs limit the usage of satellites. Within this context, RPAS emerges as a low cost alternative. Low weight unmanned aircraft are easily deployed and are able to fly at low altitudes avoiding clouds. Some companies already offer UAS based solutions, e.g. PrecisonHawk and Ursula Agriculture. The system demonstrated in (URSULA, 2016) maps efficiently blackgrass, a weed that competes with crops like wheat (Fig. 2). The seller claims that mapping blackgrass helps selecting the most effective control measure. In contrast to traditional farming, it does not necessarily compromise the yield.

For an extended UAV application list the reader is referred to (VALAVANIS; VACHTSEVANOS, 2015).

Figure 2: Blackgrass mapping. Blackgrass (left) and field mapped using an UAV (right). Source: (URSULA, 2016)

Regulations on UAV operation, such as (FAA, 2015) (DECEA, 2015), strictly require a human operator in the control loop, i.e. pilot is tele-operating or supervising the aircraft via a communication link . However, autonomous and semi-autonomous behaviours including take off, hovering, landing, obstacle avoidance and target tracking may greatly improve the mission performance and relieve the burden over the operator.

1.2 Problem Statement

The main goal of the present work is investigating the autonomous landing problem for small quadrotors operation in a non-structured environment. This is a rich problem in the sense that it allows exploring **state estimation** and **motion control**, both active topics in the research community.

State estimation comprises estimating pose (translation and rotation) and velocity (linear and angular) of the vehicle with respect to a given reference frame[1]. Sensors provide raw input data. However, sensory information is usually associated with uncertainty, delay and physical constraints. Therefore, most solutions employ multiple sensors in a complementary fashion. Data fusion manages and merges different measurements improving the overall state estimation.

Motion control concerns on how actuators shall interact with the environment such that the vehicle safely translates to the landing site. A typical solution consists in splitting the problem in two tasks: kinematic and dynamic. From a control system perspective, kinematic task is accomplished using an outer loop which computes desired rotation, velocity and acceleration that minimizes a position or velocity error. Meanwhile, dynamic task employs an inner loop that controls forces and torques such that the vehicle remains stable and follows the outer loop commands. This work assumes that the dynamic task

[1]Some authors include sensor biases and calibration parameters as state of the vehicle, e.g. (CADENA et al , 2016)

is already solved and focuses on the kinematic problem.

In particular, for airborne vehicles, Bachrach et al. (2010) stress the following challenges:

- **Restricted payload:** Low payload limits onboard computation power and forbids employing multiple high-performance sensors.
- **Odometry:** Mini-UAVs are often equipped with low cost IMU. Although adequate for attitude control, the performance of these MEMS (micro-electro-mechanical systems) is rather poor for tasks like pose estimation, which requires double integration.
- **Fast dynamics:** The fast and unstable dynamics of UAV enhance the importance of reliable perception and motion control tasks. Furthermore, the landing dynamics is critical due to the *ground effect*, a disturb that jeopardizes simple models.
- **Constant motion:** In contrast to ground vehicles or surface vessels that may stop and reset in case of task failure, aircraft are in constant motion - even when hovering. Thus, both perception and motion control tasks must be actively running throughout the entire operation.

1.3 Proposed Solution

Visual State Estimation

The first step in the landing process is deciding where to land (GARCIA-PARDO; SUKHATME; MONTGOMERY, 2002). The average GPS errors for civilian applications are 4.7 m (vertical) and 3.4 m (horizontal) (William J. Hughes Technical Center, 2014). This may be acceptable for navigation at high altitudes, but it is prohibitive for precise manoeuvres like landing. Moreover, GPS performance degrades in bad weather conditions and signal deteriorates or may even be lost in *urban canyons*[2] (LANGE; SUNDERHAUF; PROTZEL, 2009). On the other hand, as long as there is enough light on the scene, camera sensors have a high potential for environment perception (YANG; SCHERER; ZELL, 2012). Therefore, vision-based state estimation is suitable for precise manoeuvres in a non-structure environment.

Vision-based landing requires either a landmark or a high-texture obstacle-free flat region. Solutions addressed in this work relies on prior known landmarks, which provide absolute pose estimation. In contrast to high-texture solutions, landmarks can be easily employed as identifiers. Advantages includes landing on a ground or surface vessel that carries the landmark. However, since landmarks may not be visible throughout the entire operation, optical flow - which requires high-texture regions - continuously estimates the

[2] Urban canyon or street canyon is a place surrounded by tall buildings or any other urban structure. GPS operating in urban canyons are subject to multi-path effects and signal occlusion.

velocity of the vehicle. For reliable state estimation, an Extended Kalman Filter (EKF) fuses landmark, optical flow, and IMU data.

This work considers only visual and inertial data. More recently, different sensors have been merged in a single platform for environment perception. Singh et al. (2016), for instance, combine visual and laser range data for a landing strategy that detects safe zones and avoids wires.

Motion Control

This work addresses a strategy similar to the one proposed in (SARIPALLI; SUKHATME, 2003) for the motion control problem during landing – first minimize the horizontal error, and then, descend to a pre-defined minimum safety height. After that, the power decreases on open loop. The vehicle acquiesce it has touched the ground using IMU data, turning the motors off. Both horizontal and vertical errors are forced to the origin using PID (proportional-integral-derivative) control law. However, in contrast to the aforementioned work, a virtual target point (VTP) strategy ensures gently actuation, avoiding saturation on the attitude commands. The PID controller is tested for other manoeuvres such as hovering, waypoint navigation and target tracking.

Contribution

The main contributions of this dissertation are:

- Discussing a framework for inertial-visual based navigation and guidance, which comprises take off, hovering, landing and position control;
- Benchmarking five different landmark based solutions for absolute pose estimation. Simulation and real world data assess the performance of each method in respect to accuracy and processing time;
- A virtual quadrotor model integrated with ROS. The model is suitable for different algorithms other than the ones discussed in this dissertation, e.g. failure recovery, collision avoidance and coordination. It can be employed within research or classroom environment for quick validation and test.

Code developed in this dissertation is available for free under GPLv3 license at the LabRom Github repository.

1.4 Dissertation Outline

This document is organized as follow:

- **Chapter 2:** introduces background that supports the remainder of this document;

- **Chapter 3:** discusses material, methods and system architecture;
- **Chapter 4:** addresses the motion control problem;
- **Chapter 5:** addresses the state estimation problem;
- **Chapter 6:** discusses closed-loop results;
- **Chapter 7:** concludes this document and proposes future work.

Chapter 2

Preliminaries

This chapter introduces basic concepts that support the remainder of this work. Section 2.1 presents the notations, homogeneous transform and coordinate frame system. Section 2.2 addresses the basics on perspective and sigle-view point imaging models.

2.1 Basic Definitions

2.1.1 Notation

The formal notation of this work is as follow. Mathematical variables are written in italics. Upper case calligraphic letters represent sets. Scalars are typed in lower case and vectors in lower case bold. $\boldsymbol{x} = [x_1, x_2, .., x_n]^T \in \mathbb{R}^n$ represents a n-dimensional column vector and x_i its ith element. The superscript symbol T indicates the transpose. All the elements of the vectors $\mathbf{1}$ and $\mathbf{0}$ are equal to 1 and 0, respectively. $\mathbf{0}$ is known as the null vector.

The absolute value of a scalar γ is $|\gamma|$. The p-norm of a vector $\boldsymbol{x}$ is defined as

$$\|\boldsymbol{x}\|_p = (|x_1|^p + ... + |x_n|^p)^{1/p}, \quad 1 \leq p \leq \infty$$

and unless otherwise stated, $\|x\|$ refers to the 2-norm:

$$\|\boldsymbol{x}\|_2 = (|x_1|^2 + ... + |x_n|^2)^{1/2} = (\boldsymbol{x}^T\boldsymbol{x})^{1/2}$$

Upper case letters are reserved for points and matrices. The matrix $A_{m\times n} \in \mathbb{R}^{m\times n}$ contains m-rows and n-columns. The element a_{ij} sits in the ith row and jth column. The transpose matrix A^T contains the elements of A reflected over its main diagonal. A square matrix has the same numbers of rows and columns ($m = n$). The identity matrix $I_{n\times n}$ contains 1's in the main diagonal ($a_{ii} = 1, \forall i$) and the other elements are 0 ($a_{ij} = 0, \forall i \neq j$). The inverse of a matrix, denoted as A^{-1}, is unique and respects the property $A^{-1}A = I$, where I is an identity matrix with proper dimension. A matrix is said to be singular if it is not invertible, and non-singular otherwise. The existence of an inverse may be verified by the determinant of a matrix – if $det(A) = 0$, then A is singular.

Definition 2.1. ***Moorie-Penrose pseudo inverse:*** *Consider a non-square matrix* $A_{m\times n}$. *The pseudo-inverse or left generalized inverse form is*

$$A^+A = I_{n\times n},$$

where $A^+ = (A^TA)^{-1}A^T$. *And the right generalized form is*

$$AA^+ = I_{m\times m},$$

where $A^+ = A(AA^T)^{-1}$.

In particular, a square matrix A can be classified as:

- **Symmetric:** A symmetric matrix is equal to its transpose ($A = A^T$);
- **Skew-symmetric:** A skew-symmetric matrix is equal to the negative of its transpose ($A = -A^T$);
- **Orthonormal:** The column vectors that define an orthonormal matrix have unit length norm and they are orthogonal to each other. Two important properties are 1) $A^{-1} = A^T$ and 2) the product of two orthonormal matrices is also an orthonormal matrix;
- $\mathcal{SO}(\mathbf{n})$: The special group $\mathcal{SO}(n)$ is composed of the n-dimensional orthogonal matrices with determinant $+1$.

2.1.2 Homogeneous Coordinates

Homogeneous transform augments the dimension of vectors and points. This is done by adding an extra coordinate. Consider the vector $\boldsymbol{v} = [v_1, ..v_n] \in \mathbb{R}^n$, its homogeneous form is $\tilde{\boldsymbol{v}} = [\tilde{v}_1, ..\tilde{v}_n, \tilde{v}_{n+1}] \in \mathbb{R}^{n+1}$, where

$$v_i = \frac{\tilde{v}_i}{\tilde{v}_{n+1}}, \quad \forall i = 1..n$$

It is common practice to set $\tilde{v}_{n+1} = 1$, such that $\tilde{\boldsymbol{v}} = [v_1, ..v_n, 1]$. Homogeneous transform is not unique – if $\tilde{\boldsymbol{v}}$ is the homogeneous form of $\boldsymbol{v}$, then $\lambda\tilde{\boldsymbol{v}}, \forall\lambda \neq 0$ is also a valid representation.

2.1.3 Coordinate frame

Coordinate frames are useful for describing position and orientation of objects. Coordinate frame axes must be linear independent, but not mutually orthogonal. However it is convenient for them to hold the latter propriety. Fig. 3 illustrates 3 coordinate frames. Upper case letter between brackets stands for coordinate frame origin. The greek letter ξ – reads *ksi* – represents the relation between two frames. A leading superscript denotes

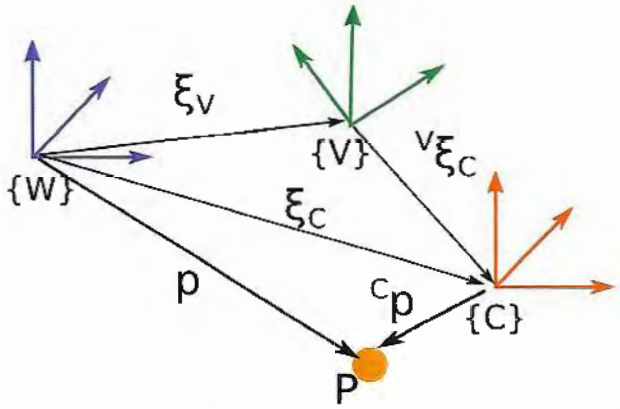


Figure 3: Coordinate frames for a robot equipped with a camera. Let $\{W\}$ corresponds to the inertial frame, $\{V\}$ vehicle frame and $\{C\}$ camera frame.

the reference frame and a trailing subscript indicates the frame being described. For example, ${}^{V}\xi_{C}$ describes the camera frame ($\{C\}$) in respect to to the vehicle frame ($\{V\}$). Vectors and points must be described in a given reference frame, which is indicated by a leading superscript. The following operations hold:

$$\begin{aligned} \xi_C &= \xi_V \oplus {}^{V}\xi_C, \\ \ominus {}^{V}\xi_C &= {}^{C}\xi_V, \\ {}^{V}\boldsymbol{p} &= {}^{V}\xi_C \cdot {}^{C}\boldsymbol{p}, \end{aligned} \tag{1}$$

where the first expression is a composition, the second an inverse transformation, and the latter a vector frame transformation. Frame notation is omitted when either clearly known within the context, or referencing the inertial world frame.

The relative pose of a coordinate frame in 3D is completely described by a translation and a rotation. There are many ways to represent it, such as the 4x4 homogeneous transformation matrix:

$${}^{A}\xi_B \sim {}^{A}T_B = \begin{bmatrix} {}^{A}R_B & {}^{A}\boldsymbol{t}_{\{B\}} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix},$$

where ${}^{A}T_B \in \mathcal{SE}(3)$ is a transformation matrix from $\{B\}$ to $\{A\}$, ${}^{A}R_B \in \mathcal{SO}(3)$ is a rotation matrix from $\{B\}$ to $\{A\}$ and ${}^{A}\boldsymbol{t}_{\{B\}}$ the translation from the origin of $\{B\}$ to $\{A\}$, expressed in $\{A\}$. The generic properties stated in (1) can rewritten as follow:

$$\begin{aligned} T_C &= T_V \, {}^{V}T_C, \\ [{}^{V}T_C]^{-1} &= {}^{C}T_V, \\ {}^{V}\boldsymbol{p} &= {}^{V}T_C \, {}^{C}\tilde{\boldsymbol{p}}, \end{aligned}$$

where ${}^{C}\tilde{\boldsymbol{p}} = [x, y, z, 1]^T$ is the homogeneous transform of ${}^{C}\boldsymbol{p} = [x, y, z]$.

2.2 Camera Model

2.2.1 Perspective Model

Consider the 3D world point $\tilde{x} = [x, y, z, 1]^T$ and its projection in the 2D image plane $\tilde{p}_i = [\lambda u, \lambda v, \lambda]^T$, both described in homogeneous coordinates. For pinhole cameras, the relationship between both points is described by the perspective imaging model:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{\rho_u} & 0 & u_0 \\ 0 & \frac{f}{\rho_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{C}R_W & {}^{C}\boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{2}$$

where λ is an unknown scale factor, f is the focal length, (ρ_u, ρ_u) scale factors associated to the pixel size, (v_0, u_0) the camera optical center coordinate, ${}^{C}R_W$ and ${}^{C}\boldsymbol{t}$ the rotation and translation of the camera w.r.t. the world frame, respectively. In a compact fashion:

$$\tilde{p}_i = K \ {}^{C}T_W \ \tilde{x}, \tag{3}$$

where K is the camera intrinsic parameters matrix and ${}^{C}T_W$ is the camera extrinsic parameters matrix.

2.2.2 Single Viewpoint Model

Following the classification enunciated in (SWAMINATHAN; GROSSBERG; NAYAR, 2003), wide-angle cameras can be described by the single viewpoint imaging model, which introduces radial and tangential distortion. The non-distorted coordinate of an individual point (u, v) can be straightforward computed for a calibrated camera:

$$(u, v) = (u_d - \delta_u, v_d - \delta_v), \tag{4}$$

where (u_d, v_d) are the distorted coordinates of point (u, v). (δ_u, δ_v) are the distortion components of each pixel parametrized as:

$$\begin{bmatrix} \delta_u \\ \delta_v \end{bmatrix} = \begin{bmatrix} u(k_1 r^2 + k_2 r^4 + k_3 r^6 + ...) \\ v(k_1 r^2 + k_2 r^4 + k_3 r^6 + ...) \end{bmatrix} + \begin{bmatrix} 2p_1 uv + p_2(r^2 + 2u^2) \\ p_1(r^2 + 2v^2) + 2p_1 uv, \end{bmatrix} \tag{5}$$

where r is the distance from the principal point, k_i and p_i are radial and tangential distortion polynomial coefficients, respectively. There are free available toolboxes that estimate the intrinsic and distortion parameters, e.g. (BOUGUET, 2015), in a process known as camera calibration. Typically the distortion is modeled using $(k_1, k_2, k_3, p_1, p_2)$ (CORKE, 2011). For most applications discussed in this work, a set of pixels rather than the whole image is employed. Therefore, applying (4) for specific pixels can be computationally more efficient than undistorting the image.

Chapter 3

System Overview

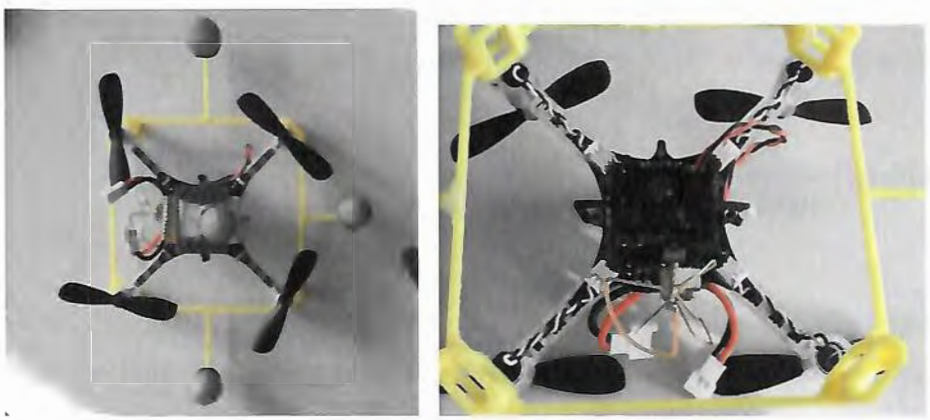
This chapter discusses material, methodology and system architecture. Section 3.1 introduces hardware, middleware and software employed for accomplishing the proposed work. Then, Section 3.2 discusses steps taken for testing and validating algorithms in a safe fashion. Finally, Section 3.3 covers system architecture and functions developed in this dissertation.

3.1 Material

3.1.1 Hardware

Crazyflie 2.0

Algorithms developed in this work were tested in the mini-quadrotor Crazyflie 2.0, manufactured by BitCraze. This 27 g open source vehicle was designed for both hobbyist and code developers. Its coreless DC-motors allow a maximum take-off weight of 42 g.



a b

Figure 4: Crazyflie 2.0 setup for experiments with Optitrack (a). Bottom view of the vehicle equipped with FX798t camera combo(b).

It is equipped with accelerometers, gyro, magnetometer, pressure sensor, bluetooh and a low-latency/long-range radio that allows operating and logging data from a remote PC. The manufacturer's radio, called Crazyradio, works reliable up to 1 km. During tests, the total flight time per battery pack was about 60 s. The quadrotor is equipped with the mini transmitter camera combo FX798T (Fig. 4b). This 4.7 g camera has an antenna that transmits analog image data. On the other end, attached to a notebook, there is an analog receiver and a digital/analog converter.

Optitrack

Optitrack is a motion capture (MoCap) system manufactured by the company NaturalPoint. A setup is available at Institute of Systems and Robotics (ISR/Lisbon), which kindly granted access to author during his internship at Lisbon. The system records the motion of retroreflective markers using the optical-passive technique. Infrared cameras capture markers position and send them to a specialized computer through a giganet connection. The proprietary software accurately computes the pose of markers, as long as enough visual cues are available. Due to its accuracy, motion capture data is assumed to be ground truth. Thus, visual-inertial state estimation algorithms are reported with respect to the MoCap estimation. The Opitrack/Crazyflie setup is shown in Fig. 4a.

Processing Unit

Processing time was evaluated on a Intel Core i7-3770 3.40GHz with 4 GB RAM equipped with a graphical card GeForce GTX 550 Ti/PCIe/SSE2.

Logitech controller

The gamepad Logitech F310 was employed for sending commands to the quadrotor. It contains 13 out 16 push buttons are available for any purpose. It also has two analog thumbsticks, which can be used to manually control the vehicle.

3.1.2 Middleware

ROS

Robot Operating System (ROS) (QUIGLEY et al., 2009) is a middleware that aims at encouraging collaborative robotics software development. It runs on Linux Ubuntu embedded in personal computer (PC) processors or *armhf* architectures. ROS interfaces communication between processes in a transparent fashion. Thereby, roboticists may focus on higher level problems. It helps users in different manners, providing device drivers, message formats, graphs, sensor visualizers and other useful libraries. Research

community plays a major role in ROS development. State-of-the-art solutions are available for free, encouraging code using, discussions and further improvements.

ROS adopts the **publisher-subscriber** architecture. Its organization relies on 4 basic elements: nodes, topics, messages, services.

- **Nodes:** A node can be a publisher, subscriber or, more commonly, both. A publisher advertises messages in topics, while the subscriber subscribes and receives new messages from topics. Most nodes receive messages, run routines and publish the outcome;
- **Topics:** Like pipelines, topics have two end. On one side, it is only allowed one publisher, which sends messages. On the other end, there are n subscribers that receive messages. Each topic has a message type and a name;
- **Messages:** Messages contain the meaningful data on a well defined format. Subscribers most known the format a priori;
- **Services:** While messages are continuously published under a topic for possible multiple subscribers, services are usually a one shot action for a specific action or query.

3.1.3 Software

V-REP

In general, implementation of algorithms are plagued by many sources of glitches, from conceptual to round-off errors due numerical precision. Therefore, at early development stage, running field test with an UAV can be hard, frustrating and potentially hazardous. Withing this context, virtual simulators are a suitable option for coding, debugging, assessing performance and improving algorithms before flights. Also, it permits performing multiple test in a relatively short time interval when contrasted to field tests.

There are robotic simulators available under free software licenses, including Gazebo by OSRF, V-REP by Coppelia Robotics and Morse by LAAS-CNRS Open. These three platforms provide the required dynamic simulation, sensory information, virtual graphics and software integration for validating the proposed solution. Decision for a specific simulator was based in following criteria:

- **Easiness for developing objects in the simulator environment**. This is important for landmark designing and reconstructing real-world scenes;
- **ROS integration**. Since real quadrotor is ROS-friendly, codes developed for the virtual vehicle can be re-used. This functionality aims at avoiding re-coding errors;

- ❑ **Technical support.** Tutorial, proper documentation and discussion forums may play an important role for overcoming problems and sharing solutions.

V-REP was chosen as the simulation platform. Besides fulfilling the basic requirements, it has a clear and user-friendly graphical user interface (GUI), and long-term support from Coppelia Robotics. For more information and simulators comparison see (SANTOS, 2014). The virtual quadrotor designed in this work is shown in Fig. 5. Its inputs are the same as a regular quadrotor – attitude angle and collective thrust. It is equipped with a down-looking camera and IMU.

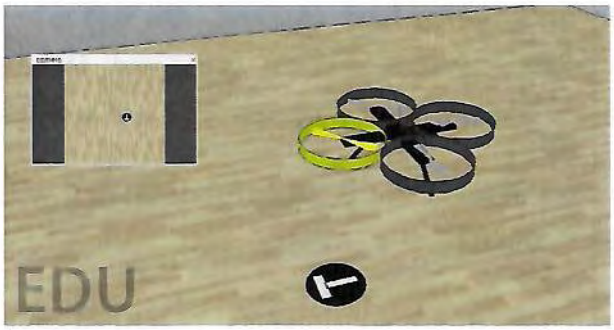


Figure 5: V-REP quadrotor simulation.

OpenCV

Open Source Computer Vision Library (OpenCV) is an open source C++/Python computer vision and machine learning library. It supports Windows, Linux, Mac OS, iOS and Android. OpenCV contains more than 2500 optimized functions implemented. This dissertation employs OpenCV 2.4.8, which comes fully integrated with a fresh ROS Indigo installation.

Eigen

Eigen is a linear algebra library for C++. It has data structure for vectors, matrix, quarternion, transformation and so on. It claims to be fast and reliable. These are important aspects due numerical limitation and computational constraints. The material developed in this work runs on Eigen 3.3.

3.2 Methodology

Algorithms were first developed and tested in simulation. Then, up to parameters tuning, solutions were deployed on the real vehicle. The motion control was validated in two steps:

1. **Software-in-the-loop**: Validation using a virtual vehicle. Simulator provides feedback for the control loop;

2. **Quadrotor/MoCap** : Validation using a Crazyflie quadrotor. The motion capture system provides feedback for the control loop.

State estimation algorithms were evaluated in a similar fashion:

1. **Software-in-the-loop**: Validation using virtual camera and IMU. Inertial and image data are generated at the same rate, i.e. simulation step. Ground truth is supplied by the simulator;

2. **Hand-held camera**: A hand-held camera (FX798T) attached to an IMU gathers image frames in a real world scenario. Ground truth is supplied by the motion capture system.

Notice that at this point the motion control is not plagued by the state estimation error, since feedback comes from an accurate source. Final experiments assess closed-loop performance running the developed visual-inertial state estimation as feedback for the control loop.

1. **Crazyflie**: Closed-loop state estimation and motion control are tested using a Crazyflie quadrotor.

3.3 Architecture

Testing the landing requires the vehicle to be airborne. Therefore, though focuses was given to the landing manoeuvre, this work explores other basic automatic capabilities, namely take-off, hovering, waypoint navigation, and target tracking. As the complexity increased, the need for a system architecture rose. The proposed architecture is shown in Fig. 6 as a direct graph composed of states (vertices) and transitions (edges). A dashed line indicates system is reachable from any other state to its right. States are described on Table 1 and transitions on Table 2. Some concepts were inspired by the work by Hoenig et al. (2015).

There are two different kinds of state: regular and critical. Critical states such as hard land or emergency are reachable from any other state in the graph. Moreover, when emergency state is called, motors are turned off and system must be rebooted. If the system is airborne, then soft land is also reachable from any other state.

Transitions can be classified either as user triggered or automatic. User triggered transitions are sent to the system using a gamepad or keyboard. The default configuration for the Logitech gamepad is as show in Fig. 7. The actions are further split into regular or critical actions. Automatic transitions are issued by the system when a success or a

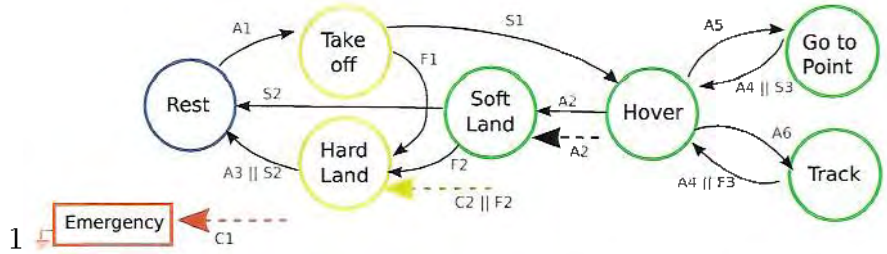


Figure 6: State machine. Dashed lines indicates transition is available for states on its right side.

failure is detected. There is no automatic transition that leads to the emergency state. In case of two or more transitions are fired at the same time, the following priority is respected: critical, failure, action, success.

The following remarks are made:

- Take-off action applies 80% of theoretical hovering thrust. The thrust increases 20% per second. The take-off succeeds if state estimation is available. The action aborts after 5 seconds.
- Once available, state estimation can be either poor or good. The estimation is considered poor if associated uncertainty is greater than a given threshold or estimated height is negative when taking into consideration the associated uncertainty.
- Landing is detected using IMU acceleration or state estimation when available. When the vehicle touches the ground, there is a negative peak in the z-axis acceleration. Axis and sign depends on how the IMU is mounted. Alternatively, if the vehicle is below 0.1 m, the land is considered to have succeed and motors turn off.
- The soft land manoeuvre consists in minimizing horizontal error w.r.t to the landing site and then vehicle descends. The hard land sets the collective thrust to 85% of theoretical hovering thrust, i.e. it is an open-loop manoeuvre.

Table 1: Reachable states for supervisory state machine.

State	Description	State Estimation	Closed-loop Control
Regular operation			
Rest	Motors off	-	-
Take Off	Increase thrust incrementally	No	No
Soft Land	Align and descend	Yes	Yes
Hover	Stabilize at flight	Yes	Yes
Go2Point	Fly to a specified 3D point	Yes	Yes
Track	Track a known target	Yes	Yes
Critical operation			
Hard Land	Set thrust $\sim$ 85% of quad's mass	-	No
Emergency	Turn motors off	-	No

Table 2: Transitions for supervisory state machine. Transitions marked with * can be fired from multiple states.

User triggered				Automatically			
Action		Critical		Success		Failure	
ID	**Description**	**ID**	**Description**	**ID**	**Descritption**	**ID**	**Description**
A1	Do take off	C1*	Do emergency land	S1	State estimation available	F1	Take off timeout
A2*	Soft Land	C2*	Do hard land	S2	Land detected	F2	Poor state estimation
A3	Restart			S3	Arrived at destination	F3	Lost track of target
A4	Hover						
A5	Go to point						
A6	Track target						

- States shown in Fig. 6 correspond to high level capabilities that may share a common controller. For example, Go2Point and Hover employs the same PID position controller.

While some of these values where tuned in flight tests, they can also be defined by the user for a specific quadrotor.

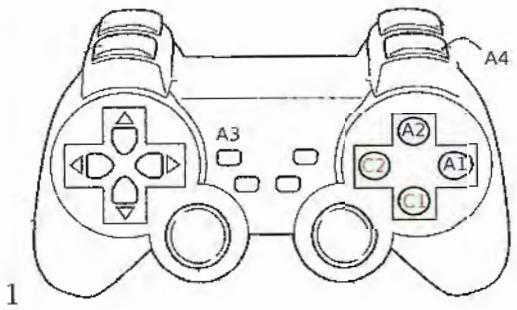


Figure 7: Default buttons configurations for logitech gamepad.

Chapter 4

Motion Control

This chapter addresses the motion control problem. Section 4.1 introduces coordinate frames, UAV kinematic and dynamic equations. Section 4.2 formulates the motion control problem. A literature review is carried out in Section 4.3. Section 4.4 discusses the proposed solution.

4.1 UAV system

4.1.1 Coordination Frames

A rigid body moving freely in the 3D space has 6 degrees of freedom (DoF): three along the translational axes – forward, lateral and vertical translations – and three around the rotational axes – roll, pitch and yaw Cardan angles. The latter are also called Tait-Bryan or Euler angles. Consider the world fixed frame $\{W\}$ and the body-fixed frame $\{B\}$ as shown in Fig. 8.

The airborne system community follows the coordinate system convention known as NED (North, East, Down). However, for ROS compatibility reasons, this work follows

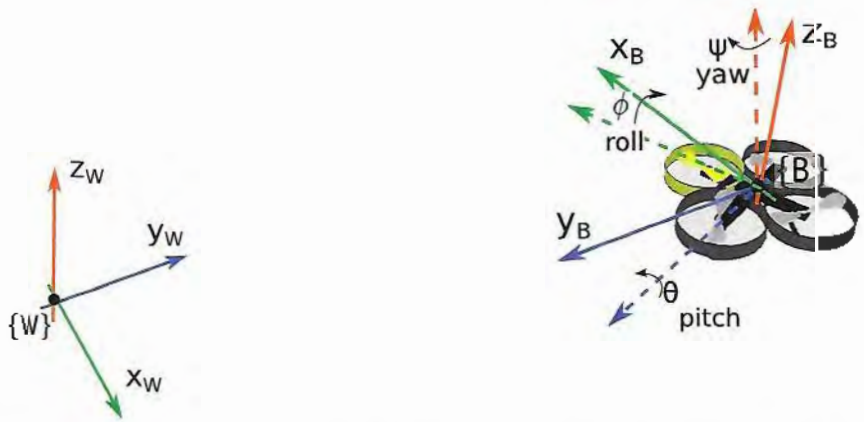


Figure 8: UAV coordinate system. Dashed x and y axes are parallel to the plane defined by the x and y world frame axes.

the East, North, Down (ENU) convention (see ROS REP 103). For the frame $\{W\}$:

- $\{x_W\}$ points East;
- $\{y_W\}$ points North;
- $\{z_W\}$ points Up.

The frame $\{B\}$ is attached to the vehicle. Its origin rests in the center of gravity (CG) of the aircraft, such that two of its axis are aligned with main inertial axes of the quadrotor:

- $\{x_B\}$ aligned with the front and rear rotors.
- $\{y_B\}$ aligned with the left and right rotors.
- $\{z_B\}$ orthogonal to the plane defined by x_B, y_B axes.

The translation, velocity and orientation of the vehicle are described using the following notation:

- $\boldsymbol{p} = [x, y, z]^T$ is the position of the origin of $\{B\}$ w.r.t. $\{W\}$;
- $\boldsymbol{\eta} = [\phi, \theta, \psi]^T$ is the orientation of $\{B\}$ w.r.t. $\{W\}$, represented using XYZ Cardan angles;

The aircraft community employs the word *attitude* for describing the roll (ϕ), pitch (θ) and yaw(ψ) angles.

4.1.2 Kinematic Model

According to the notation introduced in Section 4.1.1, the velocity and acceleration of the vehicle described in the inertial frame is

$$\dot{\boldsymbol{p}} = R_B \ {}^B\dot{\boldsymbol{p}},$$
$$\ddot{\boldsymbol{p}} = R_B \ {}^B\ddot{\boldsymbol{p}},$$

where $R_B \equiv R_B(\phi, \theta, \psi)$ is the matrix that performs roll, pitch and yaw rotations, respectively. Let $c(\cdot) \equiv \cos(\cdot)$ and $s(\cdot) \equiv \sin(\cdot)$, then

$$R_B = R_z(\psi)R_y(\theta)R_x(\phi) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\theta & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\theta & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{6}$$

4.1.3 Dynamic Model

A well known quadrotor morphology consists in four motors equally placed a distance d from the CG of the vehicle in a cross-shape configuration as illustrated in Fig. 9. Theoretically, in this configuration, during trimmed flight, gyroscopic and aerodynamic torques cancel out (CASTILLO; LOZANO; DZUL, 2004). Applying Newton's Second Law of motion yields:

$$\boldsymbol{F} = m\ddot{\boldsymbol{p}},^{1}$$

where $\boldsymbol{F} = [F_x, F_y, F_z]^T$ is the net force acting on the quadrotor. Now, consider there are only two external forces actuating on the vehicle: thrust and weight, i.e. drag forces are neglected. The four propellers produce a net thrust ${}^{B}\boldsymbol{T} = [0, 0, T]^T$. The weight $\boldsymbol{W} = [0, 0, -mg]^T$ is a function of the mass of the quadrotor m and the acceleration of gravity g, which are considered constants. For the net force, follows:

$$\begin{aligned}
\boldsymbol{F} &= \boldsymbol{T} + \boldsymbol{W}, \\
\boldsymbol{F} &= R_B {}^{B}\boldsymbol{T} + \boldsymbol{W}, \\
\boldsymbol{F} &= R_B \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}
\end{aligned}$$

Substituting the rotation matrix (6), yields:

$$F_x = m\ddot{x} = T\cos\psi\sin\theta\cos\phi + T\sin\psi\sin\phi \tag{7a}$$

$$F_y = m\ddot{y} = T\sin\psi\sin\theta\cos\phi - T\cos\psi\sin\phi \tag{7b}$$

$$F_z = m\ddot{z} = T\cos\theta\cos\phi - mg \tag{7c}$$

[1] Exceptionally, the dynamic model does not follow the vector notation introduced in Section 2.1. Instead, following the literature, force vectors are represented in upper case bold letters.

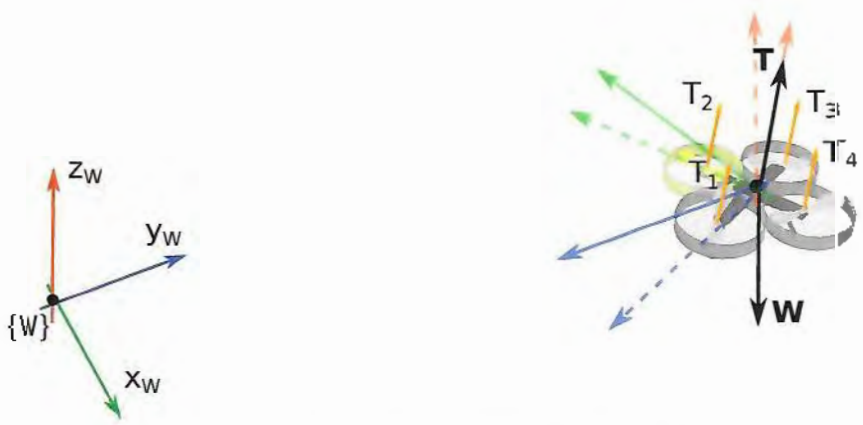


Figure 9: Force diagram for a quadrotor.

4.2 Problem Statement

Assume an autopilot takes control over the attitude stabilization problem. Therefore, given a collective thrust, roll, pitch and yaw angle command the vehicle reaches the desired command in a finite time, which is considered negligible. Next, the motion control problem is formally stated:

Problem Statement 4.1. ***(Set-point)*** *Let* $\mathbf{r} = [\mathbf{p}^T, \psi]^T \in \mathbb{R}^4$ *be the current position and orientation of the vehicle and* $\mathbf{r}_d = [\mathbf{p}_d^T, \psi_d]^T \in \mathbb{R}^4$ *the desired set-point. Consider the control input* $\mathbf{u} = [T^*, \boldsymbol{\eta}^{*T}]$, *where* T^* *is the collective thrust and* $\boldsymbol{\eta}^*$ *attitude angles inputs. Design a feedback control law for* $\mathbf{u}$ *such that the vehicle converges to the desired set-point, i.e.* $\|\mathbf{r} - \mathbf{r_d}\| \rightarrow \mathbf{0}$ *as* $t \rightarrow \infty$.

In general, most quadrotors take yaw rate as input command, rather than yaw angle. In this case, a proportional controller can take a desired yaw angle into yaw rate command. Also, notice that a quadrotor is an underactuated vehicle. Its final pose is restricted to $\phi = \theta = 0$. However, during a path, these angle can and shall assume other non-null values.

4.3 Literature Review

Three possible strategies for the motion control problem were initially considered: set-point assignment, trajectory-tracking and path-following. In the first method, the roboticist is not concerned about the path the vehicle travels to reach a goal point. The latter two approaches require a previous path-planning phase that takes into consideration the path to be travelled to reach the spatial goal. The trajectory-tracking formulation parametrizes a desired trajectory in time. Meanwhile, the path-following concerns in driving the vehicle to a desired path with a given speed assignment, without temporal constraints (AGUIAR; HESPANHA, 2007). The main advantages of path-following over trajectory tracking are no requirement to move to the trajectory starting point and less likely to saturate control signals (VANNI, 2007). .

This work explores set-point strategies. It is a versatile capability that supports different higher level actions such as landing, waypoint navigation, hovering and tracking. Also, it can be adapted in a path-following strategy assuming path has an associated dynamic. The set-point strategy is illustrated in Fig. 10. Consider the landing pad is placed in the origin of W. Then, landing command corresponds to $\mathbf{r}_d = \mathbf{0}$. A pose controller might lead the vehicle in a path similar to the one represented in green doted lines. This is a potentially dangerous path for the perception system, as the camera may lose sight of the landing target. The turnaround consists in splitting the landing in a two set-point assignment. The first aims at aligning the vehicle with the landmark, while keeping a constant height. The second task drives the vehicle towards the ground, while holding

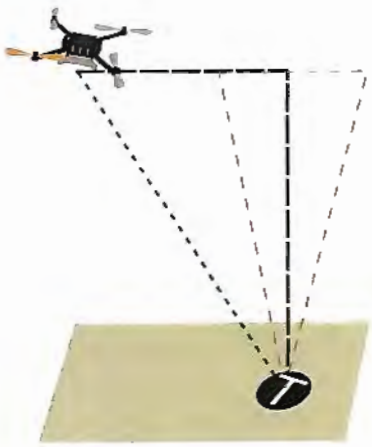


Figure 10: Set-point strategy visual example

horizontal position. The work in (YANG; SCHERER; ZELL, 2012) defines a truncate cone (dashed red lines) that the vehicle must fly within during descending procedure.

The strategy devised by Blösch et al. (2010) takes into account the non-linearities of the model by mapping linear acceleration to system input. Also, the authors observe that the roll and pitch dynamics are fast enough such that it can be modelled as second-order systems. Taking into account the identified delay, a linear gaussian regulator (LQR) associated with a feedforward action drives the vehicle to the desired position. The yaw angle is controlled independently. In a similar fashion, Kendoul, Yu e Nonami (2010) considers the delay that the attitude controller (autopilot) introduces in the control loop. Then, given a bounded trajectory, global asymptotically stability is guaranteed using independent PID controllers for the horizontal and altitude errors. A PID control yaw is also addressed by Mahony, Kumar e Corke (2012). The position and velocity error are projected in the body fixed frame. Model dynamic inversion is employed in (ACHTELIK et al., 2013). The authors apply feedback linearization for the translation and yaw variables. Then, classic control techniques known for linear systems can be applied. The orientation error is minimized using a PI controller and LQR/pole placement assures the closed loop system is stable and converges to the origin.

There are other techniques discussed in the literature which are not addressed here, e.g. fuzzy-logic (OLIVARES-MéNDEZ et al., 2010), backstepping and sliding mode control (BOUABDALLAH; SIEGWART, 2005).

4.4 Proposed Solution

4.4.1 Feedback Linearization

The position of the vehicle is described by (7), a nonlinear differential equation. The control inputs, i.e. collective thrust, roll, pitch and yaw rate, will be manipulated such that the position of the vehicle can be explicitly described by a linear system with respect

to the forces. First, the net thrust is straightforward derived from Eq. (7c):

$$T = \frac{mg - F_z}{\cos\theta\cos\phi}.$$

For the pitch, add (7a) and (7b):

$$\begin{aligned} F_x\cos\psi + F_y\sin\psi &= T\cos\psi\sin\theta\cos\phi\cos\psi + T\sin\psi\sin\phi\cos\psi + \\ &\quad T\sin\psi\sin\theta\cos\phi\sin\psi - T\cos\psi\sin\phi\sin\psi \\ &= T\sin\theta\cos\phi(\cos^2\psi + \sin^2\psi) \\ &= T\sin\theta\cos\phi\ \frac{\cos\theta}{\cos\theta} \\ &= T\cos\phi\cos\theta\tan\theta \\ \frac{F_x\cos\psi + F_y\sin\psi}{T\cos\phi\cos\theta} &= \tan\theta, \end{aligned}$$

which may be combined with Eq. (7c), yielding

$$\theta = \tan^{-1}\left(\frac{F_x\cos\psi + F_y\sin\psi}{F_z - mg}\right)$$

Repeating similar steps, but this time subtracting (7b) from (7a):

$$F_x\sin\psi - F_y\cos\psi = T\sin\phi,$$

and, once again, substituting Eq. (7c):

$$\begin{aligned} F_x\sin\psi - F_y\cos\psi &= \frac{F_Z + mg}{\cos\phi\cos\theta}\sin\phi \\ \frac{F_x\sin\psi - F_y\cos\psi}{F_Z + mg}\cos\theta &= \tan\phi, \end{aligned}$$

such that the roll angle is given as

$$\phi = \tan^{-1}\left(\frac{F_x\sin\psi + F_y\cos\psi}{F_Z + mg}\cos\theta\right)$$

Since Eq. (7) has three equations and four unknowns, it is classified as an underdetermined system. In particular, yaw angle can be arbitrarily set. Thus, the algebraic system has infinite solutions. Assuming the attitude control dynamic is fast and smooth enough, control inputs are:

$$T^* = \frac{mg + F_z}{\cos\theta^*\cos\phi^*}, \tag{8a}$$

$$\phi^* = \tan^{-1}\left(\frac{F_x\sin\psi - F_y\cos\psi}{F_Z + mg}\cos\theta^*\right), \tag{8b}$$

$$\theta^* = \tan^{-1}\left(\frac{F_x\cos\psi + F_y\sin\psi}{F_z + mg}\right), \tag{8c}$$

$$\psi^* = \psi \tag{8d}$$

The model described by (8) transforms the non-linear system in four decoupled linear systems, which can be controlled independently using classic linear techniques. The yaw angle can be arbitrarily set and the position of the vehicle $\boldsymbol{p}$ is a second order system given by Newton Second Law:

$$\ddot{x} = \frac{1}{m}F_x \qquad \ddot{y} = \frac{1}{m}F_y \qquad \ddot{z} = \frac{1}{m}F_z$$

Fig. 11 depicts the control loop for the derived model.

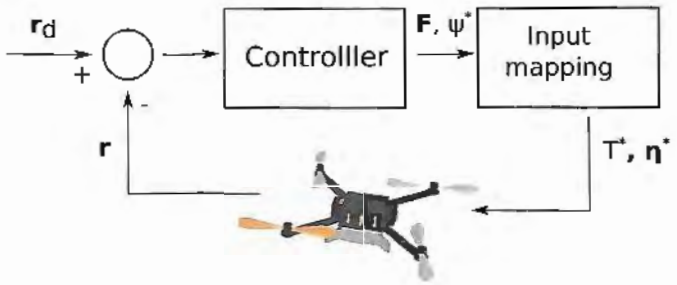


Figure 11: Control loop

4.4.2 Set-point Control Law

The closed loop system (8) with unity gain is marginally stable. It has two pure imaginary poles at $-i\sqrt{\frac{1}{m}}$. The system can be brought to stability using a PD controller:

$$\ddot{\mathbf{r}} = K_p(\mathbf{r}_d - \mathbf{r}(t)) + K_d(\dot{\mathbf{r}}_d - \dot{\mathbf{r}}),$$

where $K_p \in \mathbb{R}^{4\times4}$ and $K_d \in \mathbb{R}^{4\times4}$ are diagonal positive definite matrices. respectively. However, modelling and parameter uncertainty are likely jeopardize the PD controller. Thus, an integrative component that improves performance is added. The final set-point control law is given by:

$$\ddot{\mathbf{r}}(t) = K_p(\mathbf{r}_d - \mathbf{r}(t)) + K_i\sum_N(\mathbf{r}_d - \mathbf{r}(t)) + K_d(\dot{\mathbf{r}}_d - \dot{\mathbf{r}}(t)), \tag{9}$$

where $K_i \in \mathbb{R}^{4\times4}$ is the diagonal integrative gain matrix, positive definite. Excessive actuation is avoided using anti-windup and saturating the position error using a virtual target point. The latter is placed a given distance from the vehicle in the direction of the goal. This is similar to the carrot chasing path-following algorithm shown in (SUJIT; SARIPALLI; SOUSA, 2014).

4.4.3 Landing Manoeuvre

The landing manoeuvre is performed in two steps. The first step is inspired in (SARIPALLI; SUKHATME, 2003), which suggests minimizing the horizontal error w.r.t. to the landing target before decreasing altitude. The second step, descend and correct, is similar

to the strategy devised by Yang, Scherer e Zell (2012). The vehicle must be within a cone during the descending phase, as show in Fig. 10 in red dashed lines. When the vehicle moves out of bounds, it ceases the descend manoeuvre until horizontal error is within an acceptable value. The algorithm is explained in Algorithm 1. R_{min} is the radius of the circular section that intersects the cone at the minimum allowable flying height z_{min} before motors are turned off.

- **Minimize horizontal error**: The vehicle flies at a constant height until horizontal error is below a given threshold (T^{hor}_{error}). A dynamic threshold is derived considering the second order euclidean cone.
- **Descend and correct**: The vehicle descends at a given speed (v_{desc}) until achieve z_{min}. The vehicle aborts descending phase if horizontal error increases above T^{hor}_{error}. Once horizontal error decreases, landing manoeuvre continues.

4.4.4 Tracking

Tracking a target or object is an interesting feature in different applications, e.g. follow a leader in a multi-vehicle operation or track a suspicious target in a surveillance task. The set-point controller addressed in this works allows target tracking. The problem is simplified assuming the target is restrict to move in a planar surface. Let the pose of the tracking object be defined as $\mathbf{r_T} = [x_T, y_T, \phi_T] \in \mathbb{R}^3$. Then, the desired set-point controller becomes $\mathbf{r_d} = [x_T, y_T, z, \phi_T]$. The vehicle altitude is kept constant during tracking.

Algorithm 1 Descend and correct (Landing Manoeuvre)

Initialize: $\mathbf{r}(x, y, z, \psi), \mathbf{r_d}(x_d, y_d, z, \psi_d), v_{desc}, \Delta t, R_{min}, z_{min}$
$airborne \leftarrow true$
while $airborne$ **do**
  if $z > z_{min}$ **then**
    $e_{hor} \leftarrow \sqrt{(x_d - x)^2 + (y_d - y)^2}$
    $R \leftarrow z(R_{min}/z_{min})$
    if $e_{hor} \leq R$ **then**
      $z_d \leftarrow z_d + v_{desc}\Delta t$
    end if
  else
    $airborne \leftarrow false$
  end if
  $poseController(\mathbf{r}, \mathbf{r_d})$
  $\mathbf{r} \leftarrow readCurrentPose()$
end while

Chapter 5

State Estimation

This chapter focuses on the state estimation problem. Section 5.1 formally introduces the problem to be tackled. Then, Section 5.2 reviews solutions available in the literature. Based on the literature review, some strategies are chosen for testing. Proposed framework and implementation details are discussed in Section 5.3.

5.1 Problem Statement

In (GARCIA-PARDO; SUKHATME; MONTGOMERY, 2002), authors point out that the first step in the landing process is deciding where to land. Then, for a safe land, the relative pose between the vehicle and the landing site must be known accurately (SATO; AGGARWAL, 1997).

Problem Statement 5.1. *Define the 6 DoF pose of a body moving freely in the 3D space at time $t = t_k$ as $\mathbf{q}(t_k) = [\boldsymbol{p}^T, \boldsymbol{\eta}^T]^T \in \mathbb{R}^6$, where $\boldsymbol{p}$ stands for the translation vector and $\boldsymbol{\eta}$ the attitude angles. Also, let $\hat{\mathbf{q}}(t_k) = [\hat{\boldsymbol{p}}^T, \hat{\boldsymbol{\eta}}^T]^T \in \mathbb{R}^6$ be the correspondent pose estimation. Design a state estimation method that computes $\hat{\mathbf{x}} = [\mathbf{q}^T, \dot{\mathbf{q}}^T]^T$, such that $\|\mathbf{x} - \hat{\mathbf{x}}\| \approx \mathbf{0}$, $\forall t$.*

The state is defined here as the pose and velocity of the vehicle. However, the definition of state can be extended to other quantities such as sensor biases and calibration (CADENA et al., 2016). Focuses is given to translation, rather than rotation estimation. Inertial measurement unit can accurately estimate orientation and angular velocity.

5.2 Literature Review

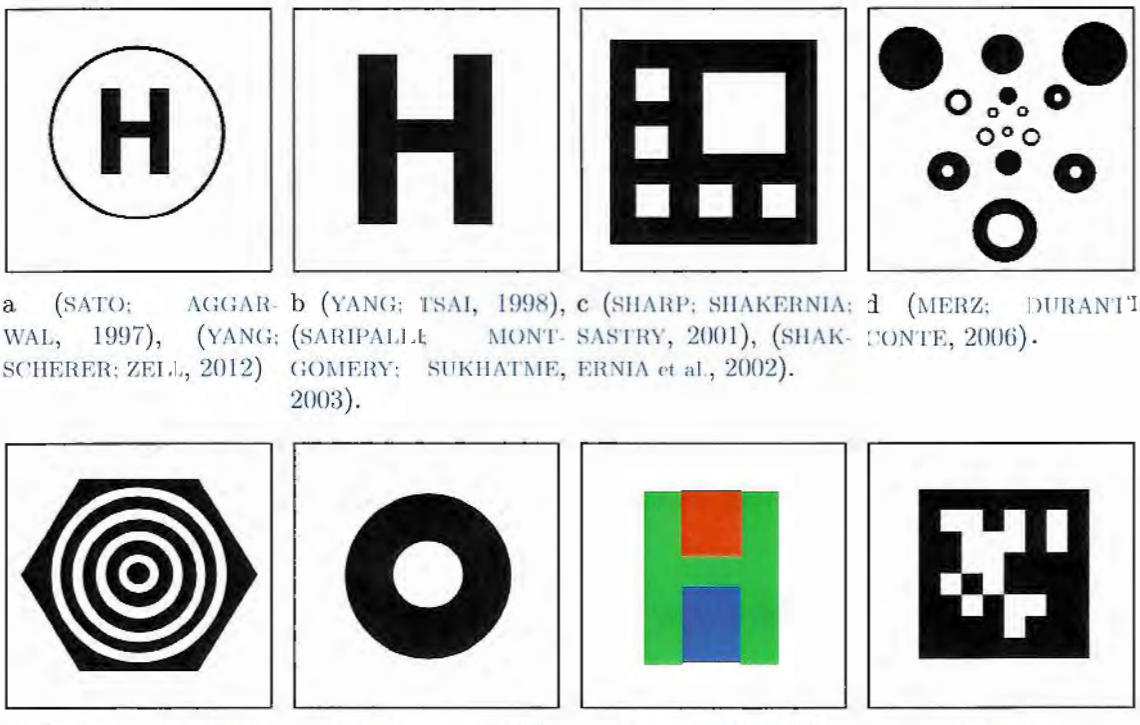
The literature review for the state estimation problem was narrowed down to helicopters and quadrotors using vision sensors for landing procedure. Regarding the landing site, some authors propose landing the vehicle in a high-texture flat region, while other resort to a prior-known landmark. The main difference between both approaches are the interconnection between the motion control and state estimation tasks. In general,

high-texture methods fall in the context of *image-based visual servoing* (IBVS), in which the control error signal is described in terms of image parameters. Most IBVS solutions employ optical flow, a technique that estimates the camera or pixel velocity, rather than the camera pose. Consequently, state estimation and motion control tasks are tightly coupled. Alternatively, for landmark-based solutions control and state estimation tasks are clearly decoupled and the closed-loop architecture is classified as *pose-based visual servoing* (PBVS).

5.2.1 Landmark

A landing target designing criteria is enunciated in (MERZ; DURANTI; CONTE, 2006): minimum size, easy recognition, reliable pose estimation from different distance ranges and minimal asymmetry. There are many possible layouts suggested by different authors. Fig. 12 helps visualizing landmarks that are discussed next.

Motivated by the fact that heliports in Japan consists in a character "H" inside a circle (Fig. 12a), Sato e Aggarwal (1997) estimate the position and orientation of a camera from a circle. From any plane other than the circle plane, a circle is projected into an ellipse. Authors estimate ellipse parameters and the radius of the circle using successive images, and on-board linear and angular velocity measurements. An extended Kalman Filter estimates the system state and accounts for error measurements. A specific



a (SATO; AGGARWAL, 1997), (YANG; SCHERER; ZELL, 2012)

b (YANG; TSAI, 1998), (SARIPALLI; MONTGOMERY; SUKHATME, 2003).

c (SHARP; SHAKERNIA; SASTRY, 2001), (SHAKERNIA et al., 2002).

d (MERZ; DURANTI; CONTE, 2006).

e (LANGE; SUNDERHAUF; PROTZEL, 2009).

f (EBERLI et al., 2010).

g (BI; DUAN, 2013).

h (OLSON, 2011).

Figure 12: Landing targets

camera motion and a careful experimental setup is required. Reported accuracy is 10% of the radius value.

The H-pattern (Fig. 12b) yields eight vanishing lines, which intercept one another generating 16 corners. Based on the projection model, Yang e Tsai (1998) exploits the directions of the lines for computing the orientation of the vehicle. Its relative position is obtained using the target collinear points and the known distance between them. Taking 0.66 s, algorithm accuracy is 2 degree and 1.1 pixel at 50 meters height.

Sharp, Shakernia e Sastry (2001) address a lightweight algorithm based on corner extraction of a landmark composed of 6 square blobs (Fig. 12c). Image processing extracts 4 corners per square. Then, resorting to linear optimization an initial guess for the camera pose is calculated. Pose estimation is refined using nonlinear optimization for minimizing reprojection error. The latter requires a good initial guess to avoid local minima. At 30 Hz, the accuracy is 5 cm (translation) and 5 deg (rotation). The aforementioned work is further extended in (SHAKERNIA et al., 2002). Instead of running the estimation step based in one frame, it uses multiple views from a single camera. Since features extracted lies on the same plane, authors employ a specialized eight-point algorithm based on homography constraints. The performance is quite similar to the first solution, however it overcomes the local minima problem. The accuracy reported is 7 cm for the translation estimation and 4 deg to the rotation.

The milestone work by Saripalli, Montgomery e Sukhatme (2003) presents a fully autonomous landing system for an UAV using vision and a behavior-based controller to follow a desired path. The vision algorithm requires a distinguishing geometric shape landmark such as the H pattern. It calculates the perimeter, area and Hu invariant image moments for recognizing the landing target. The same measurements are used for estimating the x-y coordinates of the target and its orientation in respect to the vehicle. Then, the aircraft aligns with the pad and descends with constant speed. A sonar provides height measurements. The landing error, not the vision accuracy, is 6 degree (orientation) and 40 cm (translation). Saripalli e Sukhatme (2003) boost the results obtained for landing on a moving target. A Kalman Filter is designed for tracking the target and cubic polynomials generates the desired trajectory.

Merz, Duranti e Conte (2006) propose a target composed of five equilateral triangles in different scales (Fig. 12d). In turn, three circles lay in the corner of each triangle. Thus, considering the camera intrinsic parameters are known, the three circles define uniquely the pose of the camera. This is achieved minimizing the reprojection error of the circles, which are seen as ellipses from any projection plane except the one directly from above. The system accuracy is around 1 degree (rotation).

The size of the target size has a direct impact on the estimation performance. The bigger the landmark, the better it is recognized from high heights. However, at very low heights a big target does not fit on the image. The solution addressed in (LANGE:

SUNDERHAUF; PROTZEL, 2009) tackles this problem using n unique concentric rings (Fig. 12e). For height estimation, authors assume the vehicle is flying parallel to the ground. Inertial measurement allows correcting projection errors. Results suggest that 4 concentric rings make it possible estimating the pose from 2 m to 0.2 m height. The average error is 5 cm.

In (EBERLI et al., 2010), authors estimate the camera pose from the elliptical contours of a circle in a perspective projection (Fig. 12f). The landmark consists of two concentric circles. The projection of the center of the inner circle allows to disambiguate the multiple solutions. However, when the image plane lies in the normal axis of the blob, a singularity arises. Hence, though theoretically capable of estimating pitch and roll angles, the IMU provides both angles as an input for the algorithm. The average root mean square (RMS) error is 6 cm for the the height estimation.

Yang, Scherer e Zell (2012) devise a methodology for obtaining the 6 DOF pose of a camera from a "H" pattern surrounded by a circle. A neural network classifies the connected components in a image as circle, letter "H" or other. The right match corresponds to a "H" component inside a circle. Since it is a combined match, false positive rate is low. The translation, roll and pitch angles are obtained from the projected ellipse. When solving for the pose, a sign ambiguity arises. Authors propose choosing the roll and pitch solution that are the most similar to angles reported by the IMU. However, when the roll or pitch angles are close to zero, this solution may fail and lead to large errors. The yaw angle is computed based on the 'H' character. The RMS error is 4 cm for the translation, 1.3 degree for roll and pitch angles and 4.8 degree for yaw angle.

The work described in (BI; DUAN, 2013) exploits the advantage of RGB images. The target is a green H with red and blue rectangles on the top and bottom edge, respectively ((Fig. 12g)). The colors are used for determining the orientation of the vehicle in respect to the camera. Sonars provide height estimation. Author claims that the algorithm is lightweight and robust to different lighting conditions. Accuracy is not reported.

Best known for its application in augmented reality (AR), fiducial markers may be successfully employed for the landing problem. The most prominent works regarding fiducial markers are (FIALA, 2005) (ARTag) and (OLSON, 2011) (AprilTag), which define a fiducial mark as a low payload pattern that must be accurately detected and localized even at low resolution images. Both works employ complex and robust techniques to achieve low false positive rate against different lighting conditions. Fiala (2005) uses digital coding theory, check sum and forward error correction to distinguish up to 2002 tags. Details of ARTag are not public. In (OLSON, 2011), a line detector resorting to a low pass filter and image gradient computation is designed. Then, a graph is built, where the vertices are the pixels and the edges the gradient difference between the linked pixels. The graph is sorted and clustered according to its gradient for line identification. The next task consists in detecting the 4-sided shape line segments, which the authors solves using

depth search. After that, authors solves the *Perpesctive-n-Point* (PnP), which consists in computing the 3D pose of the camera from a set of n 2D points. This problem can be solved using homography matrix. Both works focus more in the detection problem, than pose estimation. Accuracy is not clearly reported.

Table 3 summarizes the main characteristics of each work. Notice that the update rate is not a good measurement for performance as it depends on the image resolution and processing power. Both evolved dramatically over the years.

Table 3: Landmark based solutions review. A slash (-) indicates the field was not found/available or does not apply.

Work	DoF	Accuracy		Processing time (ms)	Processor	UAV tests	Techniques
		Transl (cm)	Rot (deg)				
(SATO; AGGARWAL, 1997)	6	-	-	-	-	No	Conic theory
(YANG; TSAI, 1998)	6	-	2	660	Pentium 166 MHz	No	Vanishing point detection
(SHARP; SHAKERNIA; SASTRY, 2001)	6	5	5	<30	Pentium 233 MHz	Yes	PnP
(SHAKERNIA et al., 2002)	6	7	4	<10	Pentium 233 MHz	Yes	Multiple views
(SARIPALLI; MONTGOMERY; SUKHATME, 2003)	4	-	-	<10	-	Yes	Reprojection
(SARIPALLI; SUKHATME, 2003)	4	-	-	<10	-	Yes	Reprojection and Kalman filter
(MERZ; DURANTI; CONTE, 2006)	6	-	1	<50	Pentium III 700 MHz	Yes	PnP
(LANGE; SUNDERHAUF; PROTZEL, 2009)	3	5	-	<40	Pentum IV 2.4 GHz	Yes	Conic theory and Reprojection
(EBERLI et al., 2010)	3	6	-	16	Pentium M 1.86 GHz	Yes	Conic theory and Reprojection
(YANG; SCHERER; ZELL, 2012)	6	4	4.8	18	-	Yes	Conic theory and Reprojection
(BI; DUAN, 2013)	3	-	-	<35	-	Yes	Reprojection

5.2.2 Optical Flow

Optical flow is the apparent motion of image patches (dense methods) or points (features methods) between consecutive image frames. Optical flow has been adopted by MAV roboticists in different applications, including:

- Landing on high-texture platforms (RUFFIER; FRANCESCHINI, 2004), (HERISSé et al., 2012);
- Lightweight low level strategy for stabilizing the quadrotor when a high level framework running on-board or off-board fails, e.g. GPS-based localization and visual simultaneous localization or mapping (WEISS et al., 2012), (GRABE; BüLTHOFF; GIORDANO, 2012b);
- Obstacle avoidance (ZUFFEREY; FLOREANO, 2005).

In (RUFFIER; FRANCESCHINI, 2004), authors assume that the camera optical axis is perpendicular to the ground throughout the entire flight. This is motivated by the fact

that optical flow is maximized when the point being observed lies in a plane perpendicular to the image plane. The algorithm is applied to a tethered rotorcraft. Herissé et al. (2012) de-rotate the estimated average optical flow using angular velocity provided by an IMU. A PI nonlinear controller assures hovering and landing capabilities on a moving platform. Authors address a formal proof of the system stability. However, experiments show that the height of the vehicle oscillates. Authors suggest that computational delay does not allow high gains that would guarantee asymptotically stability. The results are valid as long as the platform does not rotate.

Grabe, Bülthoff e Giordano (2012b) developed a closed-loop velocity control using optical flow. Their framework is built on top of continuous homography constraint, which assumes constant camera motion and that tracked points lie in a single plane. Experiments show that de-rotating the optical flow greatly improves its performance. However, estimating the normal plane using the gravity vector reported by the accelerometer has no major impact. The former work is further extended in (GRABE; BüLTHOFF; GIORDANO, 2012a), which lifts the planar structure constraint. Planarity measurements determine whether a new point should be incorporated to the dominant plane or not. Then, optical flow is computed taking into consideration points that belong to the dominant plane.

Weiss et al. (2012) proposed a speed estimation module based on the optical flow of at least two features. The module has many capabilities: sensor self-calibration, SLAM initialization and works as fall back solution when SLAM fails. The 8-point algorithm (LONGUET-HIGGINS, 1987) is reduced based on IMU angular speed and the assumption that the camera trajectory is smooth, i.e. continuous. An EKF framework incorporates optical flow normalized estimated velocity and IMU data to recover the unknown scale factor, as well as sensor calibration (biases and drift).

The main difference between previous methods are how each method recovers the optical flow from the 3D geometry of the scene. According to the epipolar geometry, a essential matrix, denoted E, describes a rotation and a translation:

$$E = S(\mathrm{t})R, \tag{10}$$

where $\mathbf{t} \in \mathbb{R}^3$ is the translation and $R \in SE(3)$ the rotation matrix between two consecutive frames. Since the translation vector can be recovered up to a scale, it is a 5-DoF problem. There are well known solutions to find the essential matrix, such as the eight-point or the five-point algorithm (MA et al., 2003). The minimum number of point correspondence required decreases when additional information is known a priori, e.g. rotation between frames. These algorithms are usually employed in a RANSAC scheme (FISCHLER; BOLLES, 1981; NISTÉR, 2005). RANSAC is a probabilistic method that effectively removes outliers, i.e. wrong point correspondences.

The homography matrix H is a degenerated case for points that lie in the same plane in the 3D world. It describes the relation between corresponding points in different frames,

such that points in frame {1} can be projected into frame {0} accordingly:

$$\mathbf{p}_0 = H\mathbf{p}_1. \tag{11}$$

The minimal solution requires four point correspondences. A famous solution is the Direct Linear Transformation (DLT) algorithm. Similarly to the essential matrix, minimum number of points can decrease if information is known a priori. For more information about epipolar geometry, essential and homography matrix, the reader is referred to (MA et al., 2003) and (HARTLEY; ZISSERMAN, 2004). Finally, notice that optical flow estimates velocity, thus, position estimation purely based on optical flow is prone to drift.

5.3 Proposed Solution

In this work, motion control and state estimation tasks are independently addressed. This is not a requirement for autonomous landing, but for aforementioned reasons it was the chosen strategy. In that context, there is a preference for landmark-based approaches. However, the landmark may not be visible throughout the entire landing procedure. Therefore, optical flow continuously provide velocity measurements.

5.3.1 Landmarks

Four different landmark strategies based on the literature review were implemented and compared. Landmarks are shown in Fig. 13. The methods herein addressed may be divided in the following steps: 1) image pre-processing, 2) segmentation and 3) pose estimation. (Fig. 14)

Figure 13: Modified H, circular and proposed landmark.

The **pre-processing step** consists in achieving a binary image from a gray scale image. The simplest strategy is setting a fixed threshold value. However, this method is not immune to changes in lighting. The OpenCV library offers an adaptive threshold method. Each pixel is classified based on the average or the weighted sum of the pixels contained in a sliding window. This method is considerably heavy because the average

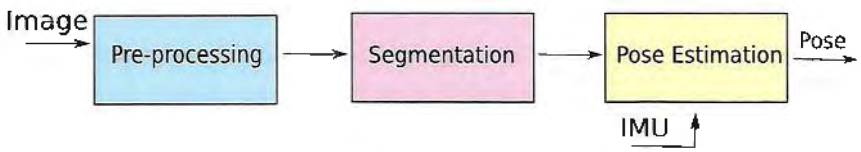


Figure 14: Perception algorithm pipeline

intensity value is computed for each pixel neighborhood. The solution adopted in this work consists in subdividing the image in n-regions (partitions) and computing the average pixel value for each region. Pixels are classified in respect to the average intensity of its containing region. The number of partitions is a function of the length of the region of interest. However, once the landmark is correctly detected, n is set to 2. This choice is adequate because most landmarks are almost symmetric. Once the landmark is detected, the image is cropped. See algorithm 2 for more details. Experimental results show that limiting the search into a region of interest significantly improves the processing time.

Algorithm 2 Grayscale to black and white image

Initialize: $n, rows, cols, C$
$p\_rows \leftarrow rows/n$
$p\_cols \leftarrow cols/n$
loop $w \leftarrow 0$ **to** n
  loop $k \leftarrow 0$ **to** n
    % Compute average pixel intensity and thresh value for current partition
    $sum \leftarrow 0$
    loop $i \leftarrow w * p\_rows$ **to** $(w+1) * p\_rows$
      loop $j \leftarrow w * p\_cols$ **to** $(k+1) * p\_cols$
        $sum \leftarrow sum + grayImg(i,j)$
      end loop
    end loop
    $avg \leftarrow sum/(p\_rows * p\_cols)$
    $thresVal \leftarrow avg - C$
    % Apply threshold to current image patch
    loop $i \leftarrow w * p\_rows$ **to** $(w+1) * p\_rows$
      loop $j \leftarrow w * p\_cols$ **to** $(k+1) * p\_cols$
        $bwImg(i,j) \leftarrow grayImg(i,j) \leq threshVal$
      end loop
    end loop
  end loop
end loop

The **segmentation step** aims at extracting meaningful information from an image. First, contours of each image blob[1] are extracted. The distortion model described in (4) is applied for these contours points to take lens distortion into consideration. Then, a set of dynamic and invariant characteristics, called descriptor, is extracted from each contour. The invariant component encompasses Hu moments, normalized area and normalized shape. The dynamic component includes the geometric center of the blob, its diagonal length, orientation and pixel area. The dynamic set is employed once the target has been detected. It is subject to changes according to the point of view and distance from the target. Meanwhile, the invariant set is used throughout the entire operation for detecting the target. It is less variant to the point of view and distance from the target.

[1] A blob is a set of connected pixels that share common characteristics, e.g. color or intensity value

For the sake of simplicity, consider that the landmark lies in the origin of the world frame $\{W\}$, as shown in Fig. 15a. The pose estimation step recovers the pose of the camera in respect to the landmark ξ_C. The segmentation step provides the geometric center of the landmark projection described in the image frame (u_L, v_L). Using homogeneous coordinates, this point can be reprojected in the camera frame using the inverse of (3):

$$^{C}\mathbf{n} = K^{-1} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix},$$

where K is the intrinsic parameter matrix and $^{C}\mathbf{n} = [x_C, y_C, 1]$ is the normalized vector that describes the position of the landmark in the camera frame. Suppose roll, pitch and yaw angles are known, e.g. previously computed or provided by an external sensor. Then, the translation from the camera to the world frame, up to a scale, is:

$$\mathbf{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} = -R_C K^{-1} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix},$$

where R_C is the rotation matrix that comprises roll, pitch and yaw angles (see (6)). Let d_{pixel} be the length of the landmark diagonal in pixels, d the known diagonal length in meters and f the focal length in pixels. Then, the scale λ is given by:

$$\lambda = \frac{d}{d_{pixel}} \frac{f}{n_3}.$$

Now, the pose ξ_C can be described using a transformation matrix T_C:

$$T_C = \begin{bmatrix} R_C & \mathbf{t}_{\{C\}} \\ \mathbf{0} & 1 \end{bmatrix},$$

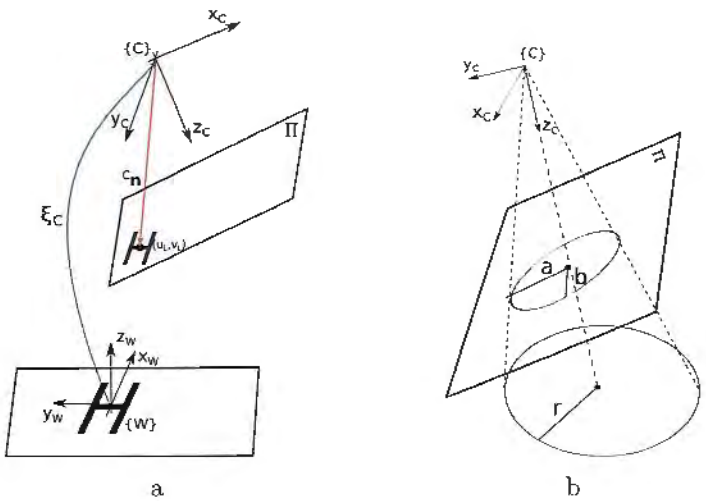


Figure 15: Landmark projection in the image plane (a). In particular, projection of circle into an ellipse (b)

where

$$\hat{\mathbf{p}} = \mathbf{t}_{\{C\}} = -\lambda R_C K^{-1} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix}. \tag{12}$$

Strategy I - Modified H

This strategy is inspired in the work by Saripalli, Montgomery e Sukhatme (2003). However instead of using a 'H' character, it employs a modified H. The horizontal bar is moved upward. The asymmetric pattern allows computing the camera orientation $\hat{\psi}$ in respect to the target from $[-\pi, \pi]$, while using a symmetric shape gives orientation from $[0, \pi]$. It assumes the camera is always facing downward, parallel to the target plane.

Strategy II - Modified H (IMU)

This is very similar to *Strategy I*, but it employs IMU data. Consequently, it does not assume that the camera is parallel to the landmark, but that it is attached to the quadrotor instead. The roll and pitch angle inputs provide a more consistent transformation from the camera frame to the landmark frame.

Strategy III - Circle (IMU)

Resorting to a circular landmark, this strategy is based on the work developed in (YANG; SCHERER; ZELL, 2012) and (EBERLI et al., 2010). Once the contours of the projected ellipse have been detected, they are de-rorated. Then, major and minor axes of the ellipse are computed. Eberli et al. (2010) employ coordinates of the outer and inner circle centers to disambiguate among the possible solutions. However, this 5-DoF algorithm performs rather poor when the image plane is almost perpendicular to the landmark. Therefore, IMU roll, pitch and yaw angles are fed as input for the pose estimation system.

Figure 15b shows visually the projection of a circle into an ellipse when the image plane is not parallel to the circular landmark. The scale can be obtained from the signature of the circle:

$$\lambda = r\frac{a}{b^2},$$

where r is the radius of the circle, a and b are the major and minor axis of the ellipse, respectively. The estimated position $\hat{\boldsymbol{p}}$ is obtained using (12). However, this time, the estimated height is $\hat{z} = \lambda$.

Strategy IV - Proposed (IMU)

The proposed landmark (see. Fig. 13) exploits the advantage of the circle (precise translation estimation) and non-symmetric shapes like the modified H-pattern (precise orientation estimation). When searching for the blob, a contour hierarchy is built. Therefore, the searching problem limits to finding one of the patterns (faster) or both (slower, but theoretically more robust to false positive).

5.3.2 Optical Flow

The optical flow solution derived in this work is based on the continuous homography matrix. When the depth difference between points is much smaller than the distance between the points and the camera, the assumption that points lies in a single plane is less punitive. This holds for quadrotors flying at high altitudes with a downward looking camera. In addition, although not implemented in this work, a RANSAC scheme can be employed to pick a dominant plane. The proposed solution is inspired on (GRABE; BÜLTHOFF; GIORDANO, 2012b). The algorithm pipeline is show in Fig. 16.

The first step is extracting features from an image frame. Features are salient regions such as a corner or a patch around it. There are many feature extractors available, e.g. SIFT (LOWE, 2004), FAST (ROSTEN; PORTER; DRUMMOND, 2010), and ORB (GALVEZ-LÓPEZ; TARDOS, 2012). The proposed solution employs the lightweight Shi-Tomasi feature detector[2](SHI; TOMASI, 1994), which is computational cheaper than the aforementioned feature extractors. The Shi-Tomasi detector computes the correlation matrix, also known as structure tensor, that describes the intensity of pixels in a patch. Eigenvalues of the correlation matrix are computed and analysed whether the minimum eigenvalue is above a user-defined threshold or not. The threshold defines what is a good feature. Each feature is associated to a pixel coordinate, which is further refined using interpolation[3]. Pixel coordinates are undistorted using (4).

The next step is finding features correspondences between two consecutive image frames. In general, optical flow resort to tracking, rather than matching. Matching requires a good descriptor, which can be computational expensive. A famous and widely employed feature tracking algorithm is the Lucas-Kanade Tracker (LKT)[4] (LUCAS; KANADE, 1981). The LKT algorithm assumes small displacement between consecutive frames, such that search for corresponding features can be restricted to a region around the previous coordinate the feature was detected. Moreover, LKT assumes pixels around a feature have the same flow. Thus, the problem is posed into an over constrained algebraic form and solved in a linear least square sense. The method becomes more robust considering multiple resolution of the image, i.e. pyramids.

[2]OpenCV function *goodFeaturesToTrack*
[3]OpenCV function *cornerSubPix*
[4]OpenCV function *calcOpticalFlowPyrLK*

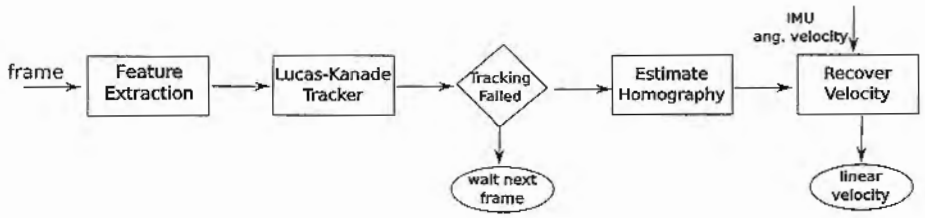


Figure 16: Continuous homography optical flow pipeline

Suppose a reasonable number of features were detected and tracked. Let ${}^{C}\mathbf{x}, {}^{C}\mathbf{p} \in \mathbb{R}^3$ be a 3D scene point and its associated tracked feature described in the camera frame, i.e. ${}^{C}\mathbf{x} = \lambda {}^{C}\mathbf{p}$. Also, let ${}^{C}\mathbf{v}, {}^{C}\mathbf{w} \in \mathbb{R}^3$ be the point linear and angular velocity, respectively. The apparent velocity of ${}^{C}\mathbf{x}$ due camera motion is:

$$ {}^{C}\dot{\mathbf{x}} = S({}^{C}\mathbf{w}){}^{C}\mathbf{x} + {}^{C}\mathbf{v} \tag{13} $$

$$ {}^{C}\dot{\mathbf{x}} = \dot{\lambda}{}^{C}\mathbf{p} + \lambda {}^{C}\dot{\mathbf{p}}, \tag{14} $$

where ${}^{C}\dot{\mathbf{p}}$ is the known optical flow given by the LKT and the camera intrisic parameters.

The homography constraint assumes all 3D points belong to the same plane, a distance d from the camera. Let $\mathbf{n} \in \mathcal{S}^2$ be the orthonormal vector that describes the plane that contains the features. Then, by simple projection it is known that $\mathbf{n}^T {}^{C}\mathbf{x} = d$. Therefore (13) can be manipulated such that:

$$
\begin{aligned}
{}^{C}\dot{\mathbf{x}} &= S({}^{C}\mathbf{w}){}^{C}\mathbf{x} + {}^{C}\mathbf{v}\,1, \\
{}^{C}\dot{\mathbf{x}} &= S({}^{C}\mathbf{w}){}^{C}\mathbf{x} + {}^{C}\mathbf{v}\frac{1}{d}\mathbf{n}^T {}^{C}\mathbf{x}, \\
{}^{C}\dot{\mathbf{x}} &= (S({}^{C}\mathbf{w}) + {}^{C}\mathbf{v}\frac{1}{d}\mathbf{n}^T){}^{C}\mathbf{x}, \\
{}^{C}\dot{\mathbf{x}} &= H\,{}^{C}\mathbf{x},
\end{aligned}
$$

where $H \in \mathbb{R}^{3\times3}$ is called continuous homography matrix. Now, isolating the known optical flow in (14):

$$
\begin{aligned}
{}^{C}\dot{\mathbf{p}} &= \frac{{}^{C}\dot{\mathbf{x}}}{\lambda} - \frac{\dot{\lambda}}{\lambda}{}^{C}\mathbf{p}, \\
{}^{C}\dot{\mathbf{p}} &= \frac{H\,{}^{C}\mathbf{x}}{\lambda} - \frac{\dot{\lambda}}{\lambda}{}^{C}\mathbf{p}, \\
{}^{C}\dot{\mathbf{p}} &= H\,{}^{C}\mathbf{p} - \frac{\dot{\lambda}}{\lambda}{}^{C}\mathbf{p},
\end{aligned}
$$

multiplying both sides by the skew-symmetric matrix $S({}^{C}\mathbf{p})$:

$$ S({}^{C}\mathbf{p}){}^{C}\dot{\mathbf{p}} = S({}^{C}\mathbf{p})H\,{}^{C}\mathbf{p} - S({}^{C}\mathbf{p})\frac{\dot{\lambda}}{\lambda}{}^{C}\mathbf{p}, $$

and since for any vector $S(\mathbf{x})\mathbf{x} = \mathbf{0}$, the former equation yield the continuous homography constraint (GRABE; BÜLTHOFF; GIORDANO, 2012a):

$$S({}^{C}\mathbf{p})H\,{}^{C}\mathbf{p} = S({}^{C}\mathbf{p}){}^{C}\dot{\mathbf{p}}, \tag{15}$$

where both the optical flow (${}^{C}\dot{\mathbf{p}}$) and point coordinate (${}^{C}\mathbf{p}$) are known. The homography matrix can be computed from (15) using a 4-point algorithm. A DLT algorithm following (MA et al., 2003) was implemented[5]. The algorithm is not discussed here. The continuous homography matrix encodes the velocity of the camera:

$$H = S({}^{C}\mathbf{w}) + {}^{C}\mathbf{v}\frac{1}{d}\mathbf{n}^{T} \tag{16}$$

As long as the transformation from IMU to the camera is known, the angular velocity of the camera can be assumed to be known a priori. Thus (16):

$$H - S({}^{C}\mathbf{w}) = {}^{C}\mathbf{v}\frac{1}{d}\mathbf{n}^{T},$$
$$\bar{H} = {}^{C}\mathbf{v}\frac{1}{d}\mathbf{n}^{T},$$

where $\bar{H} \in \mathbb{R}^{3\times3}$ is the derotated continuous homography matrix. Since $\frac{1}{d}\mathbf{n}\,{}^{C}\mathbf{v}^{T} = \bar{H}^{T}$, the unit normal vector $\mathbf{n}$ spans the matrix $\bar{H}$ and can be straight forward recovered from the singular value decomposition $\bar{H} = U\Sigma V^{T}$ as the first column of V. Assuming the camera is facing downward and the 3D points lies in the ground, the sign ambiguity can be solved forcing N_z to be positive. Now, the linear velocity can be obtained up to a scale factor d:

$$\frac{{}^{C}\mathbf{v}}{d} = \bar{H}\mathbf{n}.$$

The vehicle velocity can be obtained transforming ${}^{C}\mathbf{v}$ from the camera frame {C} to the body frame {B}. The scale can be given by the user and then propagated by the optical flow estimation, or, more robustly, provided by an external sensor, e.g. sonar.

5.3.3 Extended Kalman Filter

In the previous Sections two sources of state estimation were addressed: landmarks and optical flow. This Section presents a data fusion framework for combining both sources in a statistical fashion, namely an Extend Kalman Filter (EKF).

The Kalman filter was first proposed by Kalman (1960) and it is further explored in (THRUN; BURGARD; FOX, 2005). In particular, EKF approximates a nonlinear system to a linear system using a first-order Taylor polynomial expansion. This work takes advantage of the ROS package *robot_localization* (MOORE; STOUCH, 2014). Thus, instead of deriving

[5]DLT is implemented in labrom_optical_flow::continuous_homography class.

the equations and proving solution convergence, the remainder of this Section focuses on explaining the role that covariance matrices play on the solution convergence. Before going any further, the reader is introduced to the process model and measurement model. The process model is defined as:

$$\mathbf{x}(t_k) = f(\mathbf{x}(t_{k-1})) + \mathbf{w}(t_{k-1}),$$

where $f(.)$ is the 3D nonlinear kinematic model of the vehicle and $\mathbf{w}$ the normally distributed process noise. The measurement model is given by:

$$\mathbf{z}(t_k) = h(\mathbf{x}(t_k)) + \mathbf{v}(t_k),$$

where $\mathbf{z}$ is the measurement, h the function that maps the current state into a measurement, and $\mathbf{v}$ the normally distributed measurement noise.

There are two main steps in EKF: prediction and update. The prediction is done using the process model and it is also called state propagation. When acceleration is not known, it is assumed the constant velocity model. The prediction step follows:

$$\hat{\mathbf{x}}(t_k) = f(\hat{\mathbf{x}}(t_{k-1})) \tag{17}$$

and the associated estimate error covariance P, given by the uncertainty propagation law:

$$\hat{P}(t_k) = FP(t_{k-1}))F^T + Q, \tag{18}$$

where F is the Jacobian of $f(.)$ and Q the process noise covariance. In the prediction step, the uncertainty grows unbounded, i.e. estimation is less reliable over time. In the update step, the filter incorporates measurements from sensors:

$$\hat{\mathbf{x}}(t_k) = \hat{\mathbf{x}}(t_k) + K(\mathbf{z}(t_k) - h(\hat{\mathbf{x}}(t_k))) \tag{19}$$

where K is the Kalman gain, which can be algebraically obtained as:

$$K = \hat{P}(t_k)H^T(H\hat{P}(t_k)H^T + R)^{-1}, \tag{20}$$

where H is the Jacobian of $h(.)$ and R is the measurement covariance.

In a broad sense, the Kalman filter is a low pass filter. In (19), K determines the belief on new measurements. If K is large, state is greatly modified by income measurements. This may lead to large discontinuities in state estimation when error is large. However, if K is low, new measurements do not have a major impact in current estimation, leading to smoother state estimation. The process covariance matrix Q and measurement covariance R determines the optimal Kalman gain computed in (20). If Q is large, that is the process is highly uncertain, P is large. Consequently, K is also large and new measurements have a large weight. If R is low, that is new measurements are accurate, then K is also large.

In general, estimation uncertainty decreases when update is an absolute measurement, e.g. landmark or GPS. However, due to low sensor rate and processing time, update step runs at low rates. Meanwhile, prediction can be done at high frequencies. Thus, uncertainty increase in between update steps. In this work, **inertial data** and **optical flow** provide acceleration and velocity estimation continuously at 25 Hz during the prediction step. Meanwhile, **landmarks** – Apriltag due to its low false positive rate – are used during the update step at low frequencies (around 2 Hz) and discontinuously.

Chapter 6

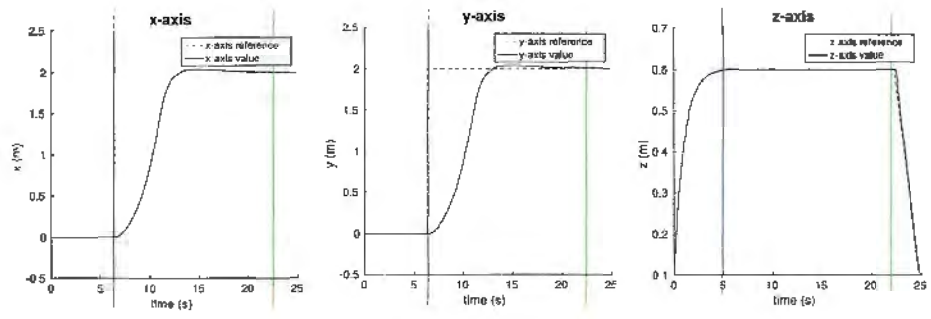
Results

This chapter presents results collected in simulation and real world environment. Motion control results using ground truth is discussed in Section 6.1. Then, state estimation results are presented in Section 6.2. Finally, with both motion control and state estimation independently validated, closed loop visual take-off, hovering and landing is shown in Section 6.3. The simulation was extensively employed for debugging and testing. However, for the sake of simplicity, the simulation results are briefly presented. Focuses is given to results collected with the Crazyflie quadrotor. Data are reported with respect to the world frame, unless otherwise stated. Results are described in terms of root mean square error (RMSE) and standard deviation.

6.1 Motion Control

6.1.1 V-REP

V-REP simulation results for take-off, way-point navigation and landing routines are illustrated in Fig. 17. While the vehicle lies at the origin, a take-off request is issued. At $t = 5\,s$, take-off finishes and the vehicle stabilizes (see Fig. 17c). Few seconds later the vehicle receives a waypoint request $\mathbf{r_d} = [2.0, 2.0, 0.6, 0.0]$. The waypoint request time is



a b c

Figure 17: Take-off, navigation and landing results in simulation: x (a), y (b), z (c) axes

represented by the blue line. The horizontal error exponentially converges to zero. There is a small overshoot, which is usually acceptable for quadrotors applications. At $t = 22\,s$, the quadrotor receives a soft land command (green line). The quadrotor descends with constant velocity until the minimum allowable flight altitude ($z_{min} = 0.1\,m$) is detected. Notice that there is no velocity controller. However, a virtual target that moves with constant downward speed has a similar effect in a well tuned system.

6.1.2 Crazyflie and Optitrack

Take-off and hovering performance is shown in Fig. 18. The vehicle lies on the ground and receives an order to take-off and hover at point $\mathbf{p} = [0, 0, 0.75]^T$ During take-off thrust increases until quadrotor is above 0.1 m. The vehicle drifts on the x-axis (Fig. 18a) more than on the y-axis (Fig. 18b). The main reason for that is the non-uniform weight distribution: markers are placed in symmetric positions along the y-axis such that the weight is well balanced; in the x-axis markers weight distribution is not symmetric. The overshoot in the z-axis was a desired behaviour to quickly overcome ground effect. RMSE w.r.t. hovering position was 0.03±0.02 m (x-axis), 0.04±0.03 m (y-axis), and 0.05±0.03 m (z-axis). The integrative component of the controller assures robustness against model uncertainty, i.e. weight distribution. Fig. 18c shows the measured thrust over time. The thrust measurement is a mapping from the PWM input. During take off, there is a thrust peak (almost 0.36 N). Afterwards, thrust seems to slowly increases. However, what is observed is motors PWM input increasing, rather than the thrust. This is due to the integrative component of the PID, which compensates for battery discharge – as battery discharges, motors drain more current to generate same amount of thrust. Fig. 18d shows the vehicle behaviour in the x-y plane during hover.

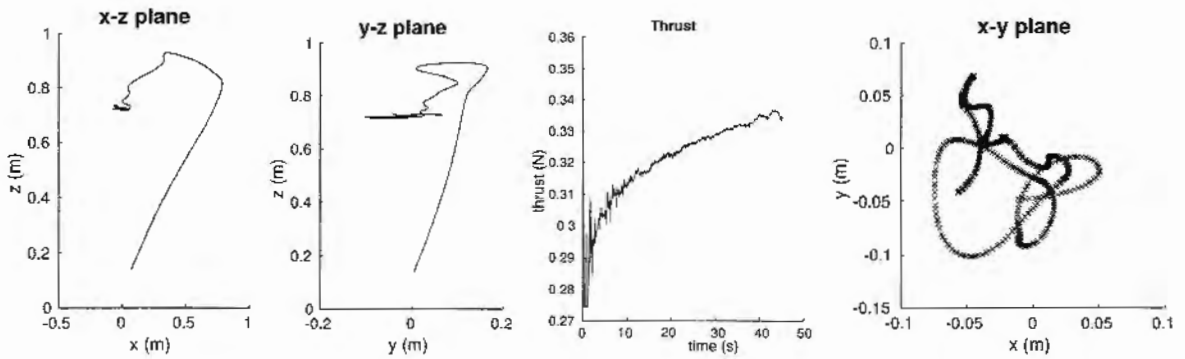


Figure 18: Crazyflie take-off and hovering. View from x-z plane (a), y-z plane (b) and measured thrust (c). x-y plane view when quadrotor has stabilized (d).

Once the vehicle is airborne, four waypoints corresponding to the vertices of a square are sent to the mission planner. Fig. 19a shows waypoints in red circles and robot trajectory. Waypoints are at the same height ($z_d = 0.6$ m) and automatically change once

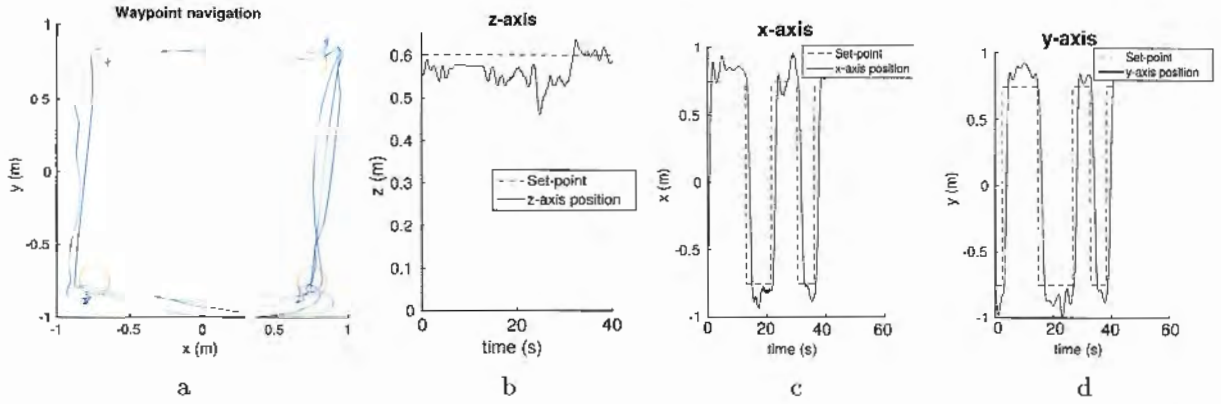


Figure 19: Crazyflie waypoint navigation. Waypoint as red circles in a top-view perspective (a). Reference and robot position for the x (b), y (c) and z (d) coordinates.

the vehicle is less than 0.1 m away. The vehicle travels at almost constant height (see Fig. 19b). The assessed RMSE on the z-axis is 0.05 ± 0.02 m. Fig. 19c, 19d depict the set-point and vehicle position in x, y axes during waypoint navigation. There is some overshoot, which can be improved tuning controller parameters if accuracy rather than speed is required.

Tracking performance was evaluated using a virtual target that moves around a circle at constant angular speed $w = 2\pi\frac{1}{8}$. Fig 20a shows the evolution of the target and vehicle. The vehicle begins at the origin of the circular pattern and converges to the circle keeping an offset from the target. Fig. 20b shows the tracking error after a long run, i.e. vehicle distance to the target. The assessed tracking RMSE is 0.42 ± 0.07 m.

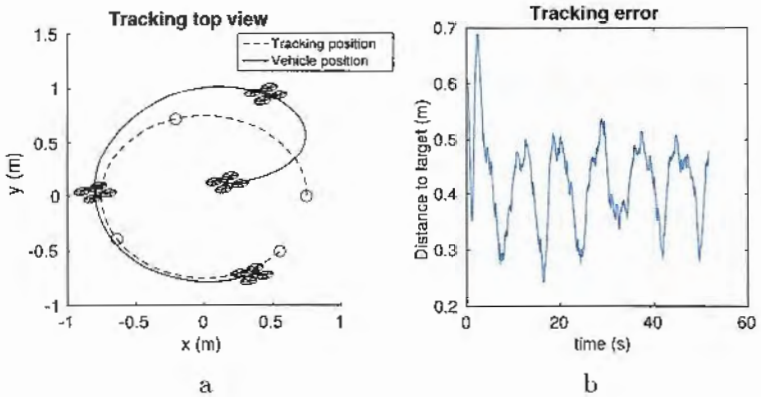
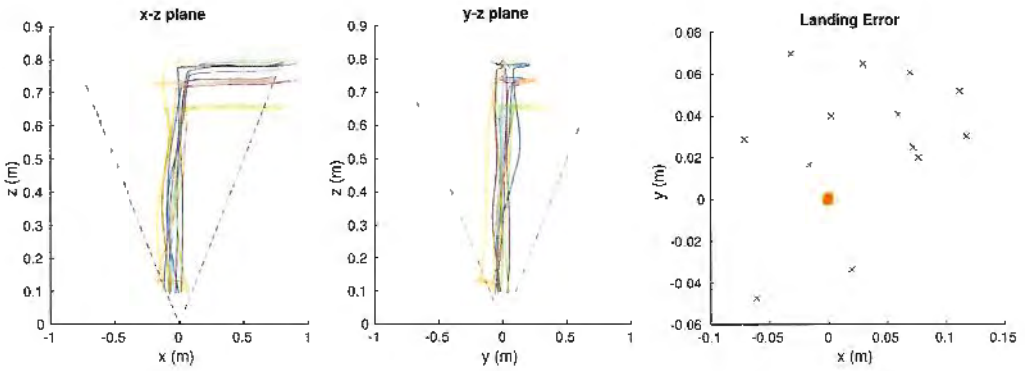


Figure 20: Crazyflie tracking a virtual target that moves in a circular pattern (a). Tracking error (b)

Finally, but not least, the landing performance is assessed. A total of 13 landing trials were carried out. The vehicle is hovering 0.75 m away from the desired landing site when the landing command is issued. First, the horizontal error is minimized as shown in Fig. 21a, 21b. Once the distance to the landing site is smaller than 0.1 m, the descending phase takes place. Fig. 21b shows that for two attempts the vehicle had to abort the



a b c

Figure 21: Crazyflie landing evaluation for 13 trials. x-z axes (a), y-z axes (b) profile w.r.t. to landing origin, and landing error (c). Dashed lines correspond to horizontal error vehicle must be within to descend.

descending phase to correct the horizontal error. Motors thrust decrease in an open loop fashion when markers are 0.1 m above the ground – CG of markers is approximately 0.03 m above motor mounts. Fig. 21c depicts the final error measured after the vehicle touches the ground. RMSE is 0.08 ± 0.02 m .

6.2 State Estimation

6.2.1 V-REP

6.2.1.1 Landmark

In addition to the four strategies discussed in Section 5.3.1, the ROS wrapper for AprilTag (WILLS, 2014) was also evaluated. Each solution was tested using a set of aggressive landing approaches as described in Table 4. As show in Fig. 22a, results for modified H without IMU estimation are poor because the assumption that the image plane is parallel to ground is weak. When IMU data aids estimating the camera angle performance for x, y-axis has a major improvement. Camera altitude estimation, that is z-axis, is better when taking into consideration conic theory (circle and proposed landmark) and the PnP formulation (AprilTag). Although AprilTag does not perform well for x, y-

Table 4: Agressive landing description, period T = 10 s and radius of trajectory R = 0.1 m. Translation described in world frame.

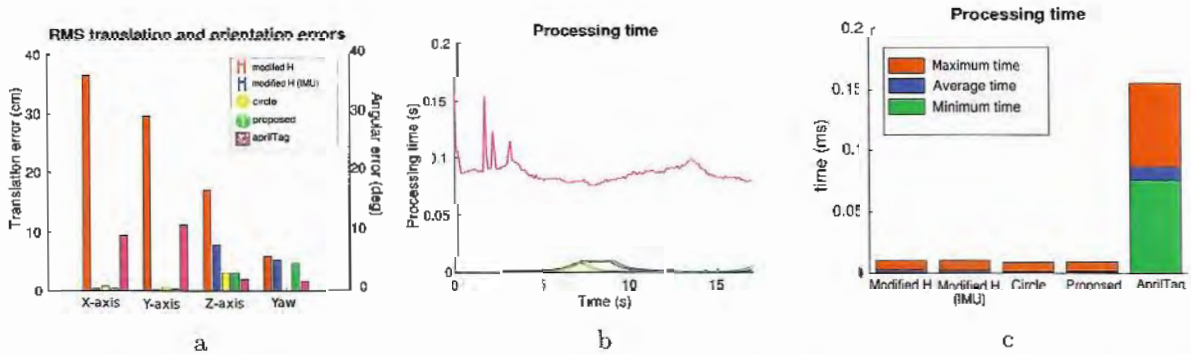
Translation	Rotation	Description
$x = 0$ $y = 0$ $z = 1.2 + \sin(2 * \pi f t)$	$\phi = -20 \cos 2\pi f t$ $\theta = -20 \cos 2\pi f t$ $\psi = 2\pi f t$	Rotating and varying roll and pitch angles in closed trajectory while moving the camera up and down in respect to landmark

Table 5: Landmak RMS error and processing time evaluation in simulation.

Strategy	IMU	RMS Error				Processing time (ms)		
		x (cm)	y (cm)	z (cm)	yaw (deg)	Min	Average	Max
Modified H (I)	No	36.5	29.7	17	5.9	0.5	2.8	10.6
Modified H (II)	Yes	0.4	0.2	8	5.3	0.5	2.5	10.6
Circle (III)	Yes	0.7	0.6	3	-	0.4	1.4	8.9
Proposed (IV)	Yes	0.4	0.4	3	4.8	0.5	1.5	9.4
AprilTag	No	9.5	11.3	2	1.6	75.4	87.0	156.0

axis translation estimation, it does perform slightly better than other methods for z-axis and yaw angle estimation.

Processing time is also evaluated. Notice that processing time is a function of the number of pixels being processed. After the landmark has been detected, the analysis takes into consideration a region of interest around the last position the landmark was observed. The more pixels a landmark occupies in an image, the slower is the algorithm. This is valid for the implementations herein devised, as shown in Fig. 22b. The peak time occurs around 7.5 s, when the camera is the closest to the landmark. AprilTag presents an almost constant processing time during the entire scenario, which is considerably slower when contrasted to other methods. Fig. 22c puts in perspective maximum, average and minimum processing time. In average, AprilTag is about 30 times slower than the other strategies. Results are summarized in Table 5. The proposed solutions show the best overall performance. The downside on the proposed solution was the false positive rate experienced in experiments, which was considerable higher than the one experienced for AprilTag.

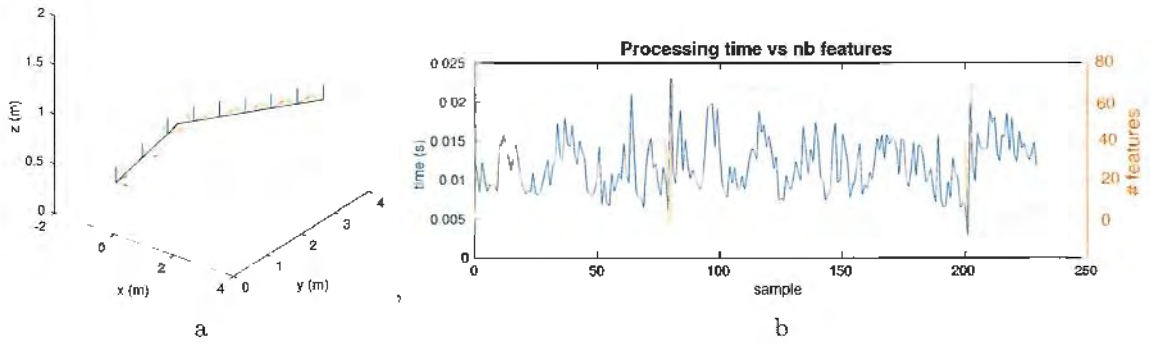


a b c

Figure 22: Landmark evaluation in simulation. RMSE error (a), processing time over time (b) and processing time bar graph (c).

6.2.1.2 optical flow

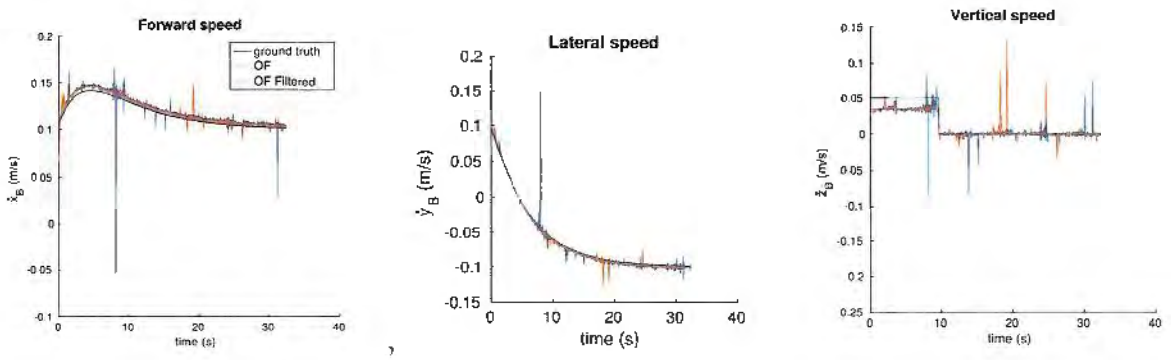
The continuous homography optical flow algorithm was tested using a trajectory as shown in Fig. 23a. The vehicle starts 0.5 m above the ground and rises until 1 m. While translating on both horizontal axes and rotating. Fig. 23b shows the processing



a b

Figure 23: V-REP optical flow evaluation. Vehicle trajectory (a) and processing time (b).

time and number of features. The average processing time is 12 ± 3.4 ms. Taking into consideration iterations that provided a valid estimation, maximum processing time is 23 ms and the minimum is 7 ms. Processing time peaks occur when the number of features drops below the minimum value – 40 features in this example – and new features have to be extracted from the image. Algorithm performance is shown in Fig. 24. As explained in Sec. 5.3.2, optical flow estimates linear velocity up to an unknown factor - depth of the scene. The scale can be recovered using a sonar sensor or a previous known landmark. In our experiments, ground truth is employed. Results show that velocity estimation quickly converges. Discontinuity on the estimation are removed using a first order low pass filter with cut-off frequency at 20 Hz. The measured RMSE was 0.005 ± 0.004 m/s, 0.004 ± 0.004 m/s and 0.012 ± 0.011 m/s for x, y and z-axes, respectively.



a b c

Figure 24: Optical flow evaluation in simulation. Ground thuth (black), optical flow estimation (blue) and filtered optical flow estimation (red) for forward (a), lateral (b) and vertical (c) vehicle speed.

6.2.1.3 EKF - Data fusion

The data fusion framework results are shown in Fig. 25. The measurement covariance matrix were set using the standard deviations previously obtained. Apriltag was employed

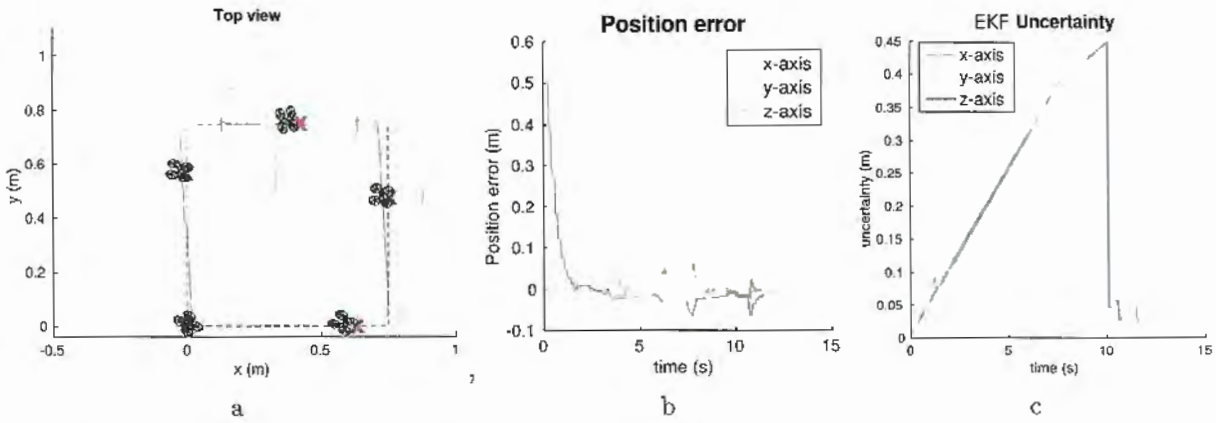


Figure 25: EKF evaluation in simulation: inertial data, landmark and optical flow data fusion. Top-view (a), estimation error (b) and estimation uncertainty (c). Vehicle travels at 0.3 m/s. Red ellipses denote associate uncertainty.

in simulation because later it will be also used as the landmark in experimental tests. Since in simulation it is possible to have a large planar area, continuous homography optical flow was tested by means of a navigation task. The tag was place at the world frame origin. The quadrotor flies around a square as shown in Fig. 25a in dotted lines, rotating 90 degrees between every two corners of the square. At the beginning, the estimated altitude error is large, but quickly drops (see Fig. 25b). As the vehicle moves away from the origin, it loses track of the landmark. Consequently, the filter relies on optical flow estimation. Although uncertainty grows, estimation drifts at low rates. After a complete lap, both error and uncertainty drops as the vehicle re-detects the tag. The RMS error was 0.03 ± 0.02 m , 0.001 ± 0.01 m and 0.02 ± 0.01 m for the x, y and z-axes, respectively.

6.2.2 Camera and IMU

6.2.2.1 Landmarks

Landmarks evaluation with a handheld camera[1] are shown in Fig. 26. Although not exactly the same, trajectories of the camera are very similar for the five methods. Height varies between 0 m to 1.5 m, motion was as smooth as possible and attitude angles were kept small. Fig. 26a illustrates the trajectory for the modified H landmark. Color brightness indicates height, from low (dark) to high (bright). The results obtained for translation estimation (Fig. 26c - 26e) validate simulation tests. However, both modified H and the proposed method outperformed AprilTag in respect to yaw estimation. The estimated roll and pitch angles RMS error using AprilTag were 5 degree and 4.6 deg, respectively. Table 6 summarizes the results achieved in respect to RMS error. Observing the recorded image data, it is possible to notice that the camera being used is highly plagued by motion blur, which deeply jeopardizes the pose estimation methods.

[1]Landmarks were evaluated with Vicon motion capture system and Philips SPC1030NC webcam.

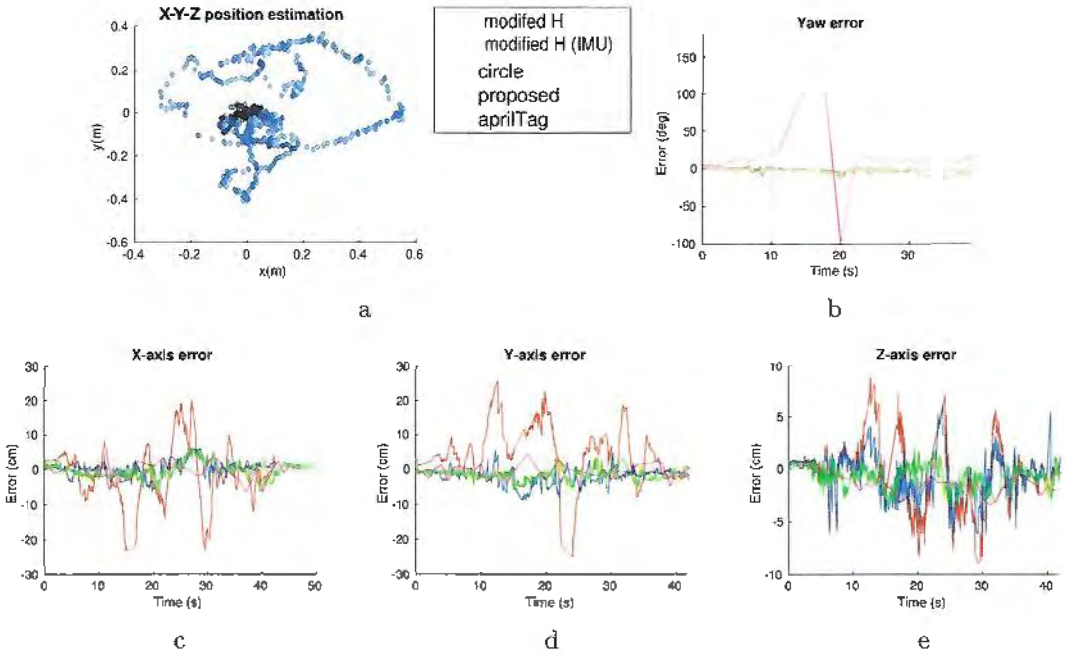


Figure 26: Hand-held camera experiments. Results described in respect to world frame. The The color code follows the one described in Fig. 26a

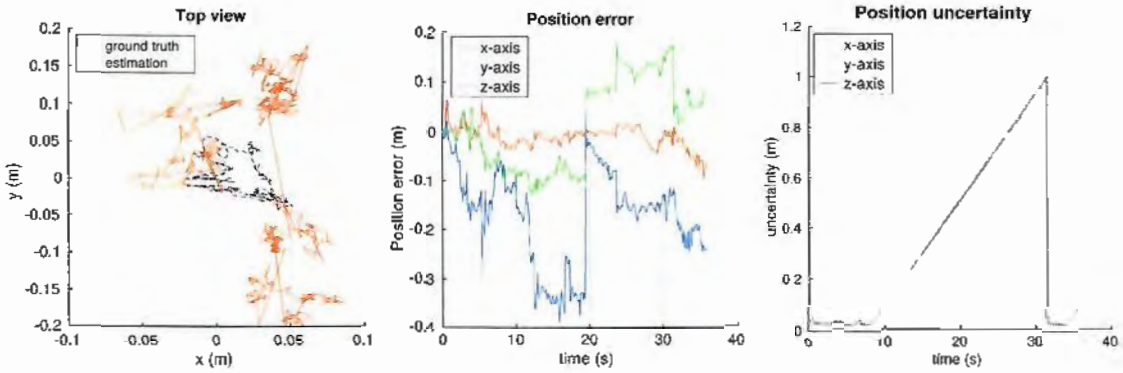
Table 6: RMS error for hand-held camera experiments

Strategy	IMU	RMS Error			
		x (cm)	y (cm)	z (cm)	yaw (deg)
Modified H (I)	No	7.0	7.74	2.26	5
Modified H (II)	Yes	2.1	2.5	1.7	4.9
Circle (III)	Yes	2.2	2.3	1.4	-
Proposed (IV)	Yes	2.2	2.3	1.4	3.2
AprilTag	No	2.5	2.6	2.1	45.4

6.2.2.2 Optical Flow/EKF - Data Dusion

The Optitrack software does not estimate velocity and pose differentiation does not provide reliable velocity estimation due to noise. Thus, instead of testing optical flow separately, this Section discusses results for EKF framework fed by optical flow, landmark and inertial data. AprilTag was the selected landmark. Although slow, it has a low false positive rate.

The testing arena was cluttered and there was not much planar space available – a requirement for the continuous homography optical flow without dominant plane selection. Within the available space, take-off, hover (at 0.75 m) and landing tests were conducted. The total time was approximately 40 s. Results are show in Fig. 27. The camera observes the landmark during the first 10 s. AprilTag process is intentionally halted, so that visual estimation relies purely on optical flow. As show in Fig. 25b and Fig. 27c, both error and uncertainty grows. Before landing, Apriltag node is turned on. The error for the



a b c

Figure 27: EKF evaluation for camera FXT98/IMU setup: inertial data, landmark and optical flow data fusion. Top-view (a), estimation error (b), estimation uncertainty (c). Vehicle simulates take-off, hovering and landing. At t=20 s there is a discontinuity in the ground truth data.

y-axis decreases, but the same behaviour is not followed by the other translational axes. Evaluated RMSE was 0.02 ± 0.02 m, 0.09 ± 0.04 m and 0.2 ± 0.10 m for the x, y and z-axes, respectively. The measurement covariance was set according to the values obtained in simulation. Most of the time, error was within covariance bounds.

6.3 Visual Closed Loop Autonomous Landing

Both motion control and state estimation have been independently validated in real world scenario in the previous sections. Now, closed loop system with visual feedback is evaluated for take-off, hovering and landing manoeuvres using the quadrotor and FXT98 camera kit. A total of five runs were done. Fig. 29 shows the top-view error w.r.t. to the landing target origin. Notice that estimated landing error was smaller than the assessed error using ground truth data. This is expected since estimation itself has an associated error. The RMS landing error was 0.17 ± 0.19 m and 0.15 ± 0.21 m for the x and y axes,

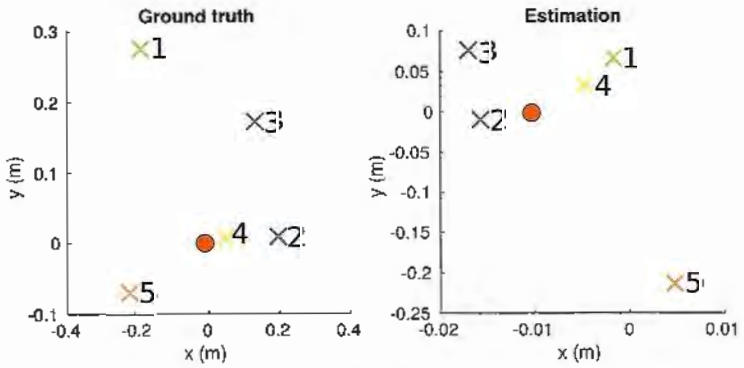


Figure 28: Vision-based landing for crazyflie/FXT98 - top view for 5 landing trials. Assigned numbers correspond to ground truth (a) and estimated landed position (b) across both images.

respectively. In the 5 trials, more than 5897 data points were collected. The RMS error during the entire flight was 0.12 ± 0.06 m, 0.11 ± 0.09 m and 0.16 ± 0.16 m for the x, y and z-axes, respectively. The measurement covariance was inflated after previous tasks, such that about 0.3% of estimation were not within uncertainty bounds.

The closed loop system is further analyzed in two specific scenarios. In the first scenario, the vehicle detects the landing target throughout the entire flight, except when it is on the ground. Results are shown in Fig. 29. The vehicle manages to keep a constant altitude and the translation error bounded. The height estimation presents some rough changes which are motivated by discontinuities in the landmark pose estimation. In the second scenario, Fig. 30, the landmark detection node is killed soon after take off. The main difference between both scenarios – beside expected uncertainty growth and position drift due to accumulated error – is the lack of discontinuities. Optical flow provides velocity estimation – a relative, rather than absolute measurement.

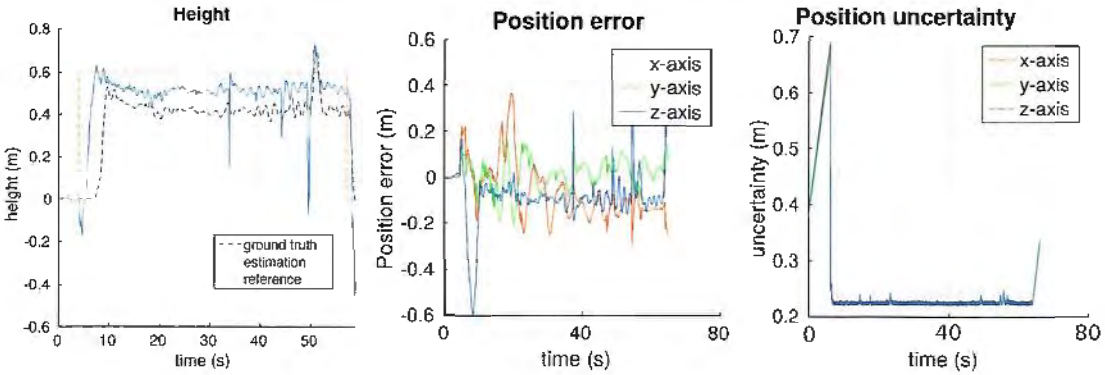


Figure 29: Crazyflie autonomous take-off, hovering and landing evaluation with visual feedback. Landmark is visible throughout the entire flight. Height (a), estimation error (b) and estimation uncertainty (c).

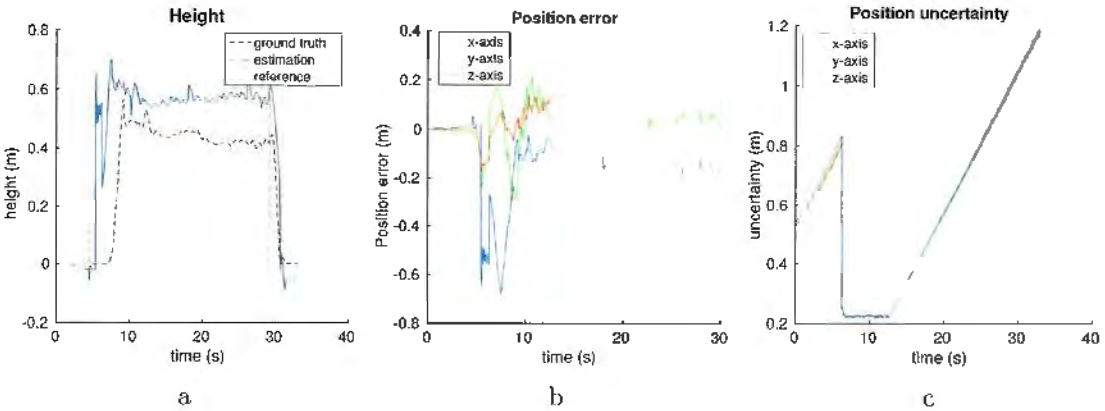


Figure 30: Crazyflie autonomous take-off, hovering and landing evaluation with visual feedback. Landmark is not visible for most of the flight. Height (a), estimation error (b) and estimation uncertainty (c).

Chapter 7

Conclusions

This chapter concludes this document. Section 7.1 remarks the accomplishments achieved during this work. Then, Section 7.2 discusses future work.

7.1 Remarks

The landing problem of mini-UAV with quadrotor configuration unravels in two tasks: motion control and state estimation.

The motion control task tackled in this work focuses on the kinematic problem. A set-point controller based on feedback linearization and PID control law assures the vehicle drives to a desired spatial goal. The landing manoeuvre is solved by first driving the vehicle closer to the landing site, while keeping a constant altitude. Then, vehicle descends until minimum height is achieved. The descend phase can be interrupted if the horizontal error grows over a threshold. In this case, horizontal error must decrease before the vehicle descend once again. Using ground truth data as feedback, the evaluated RMS landing error was $0.08\,m$. The set-point controller was also tested for waypoint navigation and tracking a virtual target.

Vision sensors are the natural choice for solving the state estimation problem for landing manoeuvres. As long as there is enough light, a camera allows precise pose estimation in a variety of scenarios. There are two possible approaches concerning the landing site, namely based on landmarks and on highly textured flat regions. In this dissertation, landmark based technique was preferred. This solution allows to separate perception and motion control tasks in a clear fashion. Five methods for estimating the pose of the quadrotor in respect to a landmark were analyzed using simulated and real world data gathered with a hand-held camera. The proposed 4 DoF pose estimation method, which relies on a circular landmark with an inner non symmetric shape, leads to the best results. It requires accurate angle measurements provided by an onboard IMU. The tests also showed that AprilTag (6 DoF) provides reliable pose estimations with low false positive rate. However, it is considerably slower. Since the landmark may

be partially occluded or not within the field of view of the camera, optical flow provides continuous velocity estimation. The technique implemented was accurate in simulated environment, but its performance could not be directly assessed with real world data. Finally, landmark, optical flow and inertial data were fused in an Extended Kalman Filter framework. Experiments with a MAV shows that state estimation pipeline RMSE is $0.1\,m$.

Both systems were independently validated using ground truth data. In practical experiments, using a small quadrotor and a miniaturized analog camera, the proposed visual closed loop system RMS landing error was $0.15\,m$.

Code developed for this work can be downloaded from LabRoM GitHub repository at `https://github.com/EESC-LabRoM/`. Packages are documented using Doxygen. Use the launch file examples for writing your own customized launch.

7.2 Future Work

The packages developed can be improved in the following way:

- optical flow: The planar scene constraint can be lift using an outlier rejection scheme that selects a dominant plane;
- V-REP model: The virtual model can be an important contribution to the community, specially for those who lack physical resources. Although available at the LabRoM repository, it has to be further documented, e.g. tutorials;
- Parameters: There is a thin line between a general solution that allows user to adapt a package to a specific problem and the curse of parameters tuning. A possible turnaround is automatic parameters tuning;
- Onboard computer/camera: The state estimation pipeline is light enough to be embedded on a onboard computer. Results should improve since delay introduced by analog to digital converter is significant;
- Camera: The analog camera employed is very noisy. It would be interesting to test algorithms using a global shutter CMOS onboard.

Bibliography

ACHTELIK, M. W. et al. Inversion based direct position control and trajectory following for micro aerial vehicles. In: **2013 IEEE/RSJ International Conference on Intelligent Robots and Systems**. 2013. p. 2933–2939. ISSN 2153-0858.

AGUIAR, P. A.; HESPANHA, J. P. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. **IEEE Transactions on Automatic Control**, 2007. v. 52, n. 8, p. 1362–1379, Aug 2007.

BACHRACH, A. et al. Range - robust autonomous navigation in gps-denied environments. In: **Robotics and Automation (ICRA), 2010 IEEE International Conference on**. 2010. p. 1096–1097.

BI, Y.; DUAN, H. Implementation of autonomous visual tracking and landing for a low-cost quadrotor. **Optik - International Journal for Light and Electron Optics**, 2013. v. 124, n. 18, p. 3296 – 3300, 2013.

BLöSCH, M. et al. Vision based mav navigation in unknown and unstructured environments. In: **Robotics and Automation (ICRA), 2010 IEEE International Conference on**. 2010. p. 21–28. ISSN 1050-4729.

BOUABDALLAH, S.; SIEGWART, R. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In: **Proceedings of the 2005 IEEE International Conference on Robotics and Automation**. 2005. p. 2247–2252. ISSN 1050-4729.

BOUGUET, J.-Y. **Caemra Calibration Toolbox for Matlab**. October 2015. Disponível em: <http://www.vision.caltech.edu/bouguetj/calib_doc/>.

BUTLER, R. **Rainforest of Brazil - An Environmental Status Report**. July 2014.

CADENA, C. et al. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. **IEEE Trans. Robotics**, 2016. v. 32, p. 1309–1332, 2016.

CASTILLO, P.; LOZANO, R.; DZUL, A. Stabilization of a mini-rotorcraft having four rotors. In: **Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on**. 2004. v. 3, p. 2693–2698 vol.3.

CHENG, P.; KUMAR, V. An almost communication-less approach to task allocation for multiple unmmaned aeria vehicles. In: **IEEE International Conference on Robotics and Automation**. Pasadena, CA, USA: , 2008. p. 19–23.

CORKE, P. I. **Robotics, Vision & Control: Fundamental Algorithms in Matlab**. : Springer, 2011. ISBN 978-3-642-20143-1.

DECEA. **Sistemas de Aeronaves Remotamente Pilotadas e o Acessso ao Espaço Aéreo Brasileiro**. 2015.

EBERLI, D. et al. Vision based position control for mavs using one single circular landmark. **Journal of Intelligent & Robotic Systems**, 2010. v. 61, n. 1, p. 495–512, 2010.

FAA. **Operation and certification of small unmanned aircraft systems, 2015. Department of Transportation**. 2015.

FIALA, M. Artag, a fiducial marker system using digital techniques. In: **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**. 2005. v. 2, p. 590–596 vol. 2.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM**, 1981. ACM, New York, NY, USA, v. 24, n. 6, p. 381–395, jun. 1981. ISSN 0001-0782.

GALVEZ-LóPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. **IEEE Transactions on Robotics**, 2012. v. 28, n. 5, p. 1188–1197, Oct 2012.

GARCIA-PARDO, P. J.; SUKHATME, G. S.; MONTGOMERY, J. F. Towards vision-based safe landing for an autonomous helicopter. **Robotics and Autonomous Systems**, 2002. v. 38, n. 1, p. 19–29, 2002.

GRABE, V.; BüLTHOFF, H. H.; GIORDANO, P. Robust optical-flow based self-motion estimation for a quadrotor UAV. In: **2012 IEEE/RSJ International Conference on Intelligent Robots and Systems**. 2012. p. 2153–2159.

GRABE, V.; BüLTHOFF, H. H.; GIORDANO, P. R. On-board velocity estimation and closed-loop control of a quadrotor uav based on optical flow. In: **2012 IEEE International Conference on Robotics and Automation**. 2012. p. 491–497.

GRISSO, R. et al. **Precision Farming: A comprehensive Approach**. 2002.

HARTLEY, R. I.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. Second. : Cambridge University Press, ISBN: 0521540518, 2004.

HERISSé, B. et al. Landing a vtol unmanned aerial vehicle on a moving platform using optical flow. **IEEE Transactions on Robotics**, 2012. v. 28, n. 1, p. 77–89, Feb 2012. ISSN 1552-3098.

HOENIG, W. et al. Mixed reality for robotics. In: **IEEE/RSJ Intl Conf. Intelligent Robots and Systems**. Hamburg, Germany: , 2015. p. 5382 – 5387.

JENKINS, D.; VASIGHI, B. **AUVSI's The Economic Impact of Umanned Aircraft Systems Integration in the United States**. March 2013.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **ASME Journal of Basic Engineering**, 1960. 1960.

KENDOUL, F.; YU, Z.; NONAMI, K. Guidance and nonlinear control system for autonomous flight of minirotorcraft unmanned aerial vehicles. **Journal of Field Robotics**, 2010. Wiley Subscription Services, Inc., A Wiley Company, v. 27, n. 3, p. 311–334, 2010. ISSN 1556-4967.

LANGE, S.; SUNDERHAUF, N.; PROTZEL, P. A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments. In: **Advanced Robotics, 2009. ICAR 2009. International Conference on**. 2009. p. 1–6.

LONGUET-HIGGINS, H. C. Readings in computer vision: Issues, problems, principles, and paradigms. In: FISCHLER, M. A.; FIRSCHEIN, O. (Ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. cap. A Computer Algorithm for Reconstructing a Scene from Two Projections, p. 61–62. ISBN 0-934613-33-8.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, 2004. v. 60, n. 2, p. 91–110, 2004. ISSN 1573-1405.

LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: **Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981. (IJCAI'81), p. 674–679.

MA, Y. et al. **An Invitation to 3-D Vision: From Images to Geometric Models**. : SpringerVerlag, 2003. ISBN 0387008934.

MAHONY, R.; KUMAR, V.; CORKE, P. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. **IEEE Robotics Automation Magazine**, 2012. v. 19, n. 3, p. 20–32, Sept 2012.

MERINO, L.; RAMIRO, J. M. D.; OLLERO, A. Cooperative unmanned aerial systems for fire detection, monitoring, and extinguishing. In: VALAVANIS, K. P.; VACHTSEVANOS, G. J. (Ed.). **Handbook of Unmanned Aerial Vehicles**. : Springer Netherlands, 2014. cap. 112, p. 2693–2722.

MERZ, T.; DURANTI, S.; CONTE, G. Autonomous landing of an unmanned helicopter based on vision and inertial sensing. In: _____. **Experimental Robotics IX: The 9th International Symposium on Experimental Robotics**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 343–352.

MOORE, T.; STOUCH, D. A generalized extended kalman filter implementation for the robot operating system. In: **Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)**. : Springer, 2014.

NISTÉR, D. Preemptive ransac for live structure and motion estimation. **Machine Vision and Applications**, 2005. v. 16, n. 5, p. 321–329, 2005. ISSN 1432-1769.

OLIVARES-MéNDEZ, M. et al. Fuzzy controller for uav-landing task using 3d-position visual estimation. In: **Fuzzy Systems (FUZZ), 2010 IEEE International Conference on**. 2010. p. 1–8. ISSN 1098-7584.

OLSON, E. Apriltag: A robust and flexible visual fiducial system. In: **Robotics and Automation (ICRA), 2011 IEEE International Conference on**. 2011. p. 3400–3407.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: **ICRA workshop on open source software**. 2009. v. 3, n. 3.2, p. 5.

ROSTEN, E.; PORTER, R.; DRUMMOND, T. Faster and better: A machine learning approach to corner detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 2010. v. 32, n. 1, p. 105–119, Jan 2010.

RUFFIER, F.; FRANCESCHINI, N. Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction. In: **Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on**. 2004. v. 3, p. 2339–2346 Vol.3. ISSN 1050-4729.

SANTOS, J. **Multi-robot cooperation systems for assembly automobile industry**. Dissertação (Mestrado) — University of Porto, Portugal, 2014.

SARIPALLI, S.; MONTGOMERY, J. F.; SUKHATME, G. S. Visually guided landing of an unmanned aerial vehicle. **Robotics and Automation, IEEE Transactions on**, 2003. v. 19, n. 3, p. 371–380, June 2003.

SARIPALLI, S.; SUKHATME, G. S. Landing on a moving target using an autonomous helicopter. In: **Field and Service Robotics, 2003. Proceedings. IEEE International Conference on**. 2003.

SATO, M.; AGGARWAL, J. K. Estimation of position and orientation from image sequence of a circle. In: **Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on**. 1997. v. 3, p. 2252–2257 vol.3.

SHAKERNIA, O. et al. Multiple view motion estimation and control for landing an unmanned aerial vehicle. In: **Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on**. 2002. v. 3, p. 2793–2798.

SHARP, C. S.; SHAKERNIA, O.; SASTRY, S. A vision system for landing an unmanned aerial vehicle. In: **Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on**. 2001. v. 2, p. 1720–1727 vol.2.

SHI, J.; TOMASI, C. Good features to track. In: **1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition**. 1994. p. 593–600.

SINGH, S. et al. Perception for safe autonomous helicopter flight and landing. In: **Unmanned Aircraft Systems (ICUAS), 2015 International Conference on**. 2016. p. AHS International 72nd Annual Forum and Technology Display.

SUJIT, P. B.; SARIPALLI, S.; SOUSA, J. B. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. **IEEE Control Systems**, 2014. v. 34, n. 1, p. 42–59, Feb 2014. ISSN 1066-033X.

SWAMINATHAN, R.; GROSSBERG, M. D.; NAYAR, S. K. A perspective on distortions. In: **CVPR**. 2003.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)**. : The MIT Press, 2005. ISBN 0262201623.

URSULA, U. A. **URSULA Agriculture Precision Agriculture Solution from UAVs (drones), Satellites and Farm Data.** 2016. Disponível em: <http://www.ursula-agriculture.com/>.

VALAVANIS, K. P.; VACHTSEVANOS, G. J. **Handbook of Unmanned Aerial Vehicles.** : Springer Netherlands, 2015. ISBN 978-90-481-9706-4.

VANNI, F. **Coordination motion control of multiple autonomous underwater vehicles.** Dissertação (Mestrado) — Department of Mechanical Engineering, Insituto Superior Técnico, Universidade Técnica de Lisboa, Portugal, 2007.

WEISS, S. et al. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In: **2012 IEEE International Conference on Robotics and Automation.** 2012. p. 957–964.

William J. Hughes Technical Center. **Global Positioning System (GPS) Standard Positioning Service (SPS) Performance Analysis Report.** July 2014.

WILLS, M. **ROS package aprilTag_ros.** 2014. Accessed: 2015-04-25. Disponível em: <http://wiki.ros.org/apriltags_ros>.

YANG, S.; SCHERER, S. A.; ZELL, A. An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. **Journal of Intelligent & Robotic Systems**, 2012. v. 69, n. 1, p. 499–515, 2012.

YANG, Z.-F.; TSAI, W.-H. Using parallel line information for vision-based landmark location estimation and an application to automatic helicopter landing. **Robotics and Computer-Integrated Manufacturing**, 1998. v. 14, n. 4, p. 297 – 306, 1998.

ZHANG, C.; KOVACS, J. M. The application of small unmanned aerial systems for precision agriculture: a review. **Precision Agriculture**, 2012. Springer, VAN GODEWIJCKSTRAAT 30, 3311 GZ DORDRECHT, NETHERLANDS, 13, n. 6, p. 693–712, 2012.

ZUFFEREY, J. C.; FLOREANO, D. Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control. In: **Proceedings of the 2005 IEEE International Conference on Robotics and Automation.** 2005. p. 2594–2599.