Universidad Politécnica de Madrid
Escuela Técnica Superior De Ingenieros Industriales

Visual Tracking, Pose Estimation, and Control for Aerial Vehicles

A thesis submitted for the degree of
Doctor of Philosophy in Robotics and Automation

Carol V. Martínez Luna
BEng Mechatronics Engineering

2013

Escuela Técnica Superior De Ingenieros Industriales
Universidad Politécnica de Madrid

Visual Tracking, Pose Estimation, and Control for Aerial Vehicles

A thesis submitted for the degree of
Doctor of Philosophy in Robotics and Automation

Author:
Carol V. Martínez Luna
BEng Mechatronics Engineering

Advisor:
Pascual Campoy Cervera
Ph.D. Full Professor Industrial Engineer

2013

Titulo:

Visual Tracking, Pose Estimation, and Control for Aerial Vehicles

Autor:
Carol V. Martínez Luna
BEng Mechatronics Engineering

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid el día 24 de Junio de 2013

Tribunal

Presidente	:	Dr. D. Sergio Domínguez Cabrerizo
Vocal	:	Dr. D. Arturo de la Escalera Hueso
Vocal	:	Dr. D. Pedro José de Melo Texeira
Vocal	:	Dr. D. Rogelio Lozano
Secretario	:	Dr. D^{a}. Ángela Ribeiro Seijas
Suplente	:	Dr. D. Gonzalo Pajares Martinsanz
Suplente	:	Dr. D. José María Armingol

Realizado el acto de lectura y defensa de la tesis el día 15 de Julio de 2013.

Calificación de la Tesis .....

El Presidente: Los Vocales:

El Secretario:

To my dear husband,
to my lovely daughter,
and to my unforgettable son.

Carol

Acknowledgements

I would like to take this opportunity to thank all the persons that have been part of this stage of my life. It was a long and difficult journey, but with the company of many persons it looked faster and easier.

First of all I would like to thank the person that gave me the opportunity to start my PhD: my supervisor Pascual Campoy. Thank you for trusting in me, for helping me become a researcher, for many things I have learned about computer vision and UAVs, and especially for all your support.

I would also like to thank Sergio Dominguez. You have also contributed to my research and teaching career in the computer vision field. I would also like to thank you for all the mornings coffees that helped me forget for a while all the deadlines I always had in mind.

I would like to thank the previous members of the computer vision group: Ivan Mondragón and Miguel Olivares. Thanks for all your help and support during all these years. It has been a pleasure to work with you two. I hope we can meet again. I would also like to thank all the current and previous members of the computer vision group for always being willing to help me.

I would also like to thank everyone at the DISAM: to all the professors, for all the things I learned from you; to Rosa, Tere, Carlos, and Angel, thanks for all your help; and to all my current and previous fellow PhD students, for all the good times we shared.

I would like to thank The Australian Research Centre for Aerospace Automation ARCAA - (Australia) and the University of Bristol, especially professors Luis Mejias and Thomas Richardson, for having received me as a visiting researcher and for all your help and support. It was a pleasure to work with and learn from you.

I would also like to thank the Universidad Politécnica de Madrid and the Banco Santander for the financial support I received for my PhD studies, that made it possible for me to complete them.

Thanks to all the persons that have collaborated in the my work on this thesis who have not been mentioned.

Finally, I would like to thank my family for all their support and good wishes. I would especially like to thank my husband for all his support and for helping me to not surrender; and my lovely daughter and my unforgettable son for all the inspiration and motivation you gave me when I was writing these pages. Thanks to the three of you for always putting a big smile on my face :)

Resumen

El principal objetivo de esta tesis es dotar a los vehículos aéreos no tripulados (UAVs, por sus siglas en inglés) de una fuente de información adicional basada en visión. Esta fuente de información proviene de cámaras ubicadas a bordo de los vehículos o en el suelo. Con ella se busca que los UAVs realicen tareas de aterrizaje o inspección guiados por visión, especialmente en aquellas situaciones en las que no haya disponibilidad de estimar la posición del vehículo con base en GPS, cuando las estimaciones de GPS no tengan la suficiente precisión requerida por las tareas a realizar, o cuando restricciones de carga de pago impidan añadir sensores a bordo de los vehículos. Esta tesis trata con tres de las principales áreas de la visión por computador: seguimiento visual y estimación visual de la pose (posición y orientación), que a su vez constituyen la base de la tercera, denominada control servo visual, que en nuestra aplicación se enfoca en el empleo de información visual para controlar los UAVs. Al respecto, esta tesis se ocupa de presentar propuestas novedosas que permitan solucionar problemas relativos al seguimiento de objetos mediante cámaras ubicadas a bordo de los UAVs, se ocupa de la estimación de la pose de los UAVs basada en información visual obtenida por cámaras ubicadas en el suelo o a bordo, y también se ocupa de la aplicación de las técnicas propuestas para solucionar diferentes problemas, como aquellos concernientes al seguimiento visual para tareas de reabastecimiento autónomo en vuelo o al aterrizaje basado en visión, entre otros. Las diversas técnicas de visión por computador presentadas en esta tesis se proponen con el fin de solucionar dificultades que suelen presentarse cuando se realizan tareas basadas en visión con UAVs, como las relativas a la obtención, en tiempo real, de estimaciones robustas, o como problemas generados por vibraciones. Los algoritmos propuestos en esta tesis han sido probados con información de imágenes reales obtenidas realizando pruebas *on-line* y *off-line*. Diversos mecanismos de evaluación han sido empleados con el propósito de analizar el desempeño de los algoritmos propuestos, entre los que se incluyen datos simulados, imágenes de vuelos reales, estimaciones precisas de posición empleando el sistema VICON y comparaciones con algoritmos del estado del arte. Los resultados obtenidos indican que los algoritmos de visión por computador propuestos tienen un desempeño que es comparable e incluso mejor al de algoritmos que se encuentran en el estado del arte. Los algoritmos propuestos permiten la obtención de estimaciones robustas en tiempo real, lo cual permite su uso en tareas de control visual. El desempeño de estos algoritmos es apropiado para las exigencias de las distintas aplicaciones examinadas: reabastecimiento autónomo en vuelo, aterrizaje y estimación del estado del UAV.

Abstract

The main objective of this thesis is to provide Unmanned Aerial Vehicles (UAVs) with an additional vision-based source of information extracted by **cameras located either on-board or on the ground**, in order to allow UAVs to develop visually guided tasks, such as landing or inspection, especially in situations where GPS information is not available, where GPS-based position estimation is not accurate enough for the task to develop, or where payload restrictions do not allow the incorporation of additional sensors on-board. This thesis covers three of the main computer vision areas: **visual tracking and visual pose estimation**, which are the bases the third one called **visual servoing**, which, in this work, focuses on using visual information to control UAVs. In this sense, the thesis focuses on presenting novel solutions for solving the tracking problem of objects when using cameras on-board UAVs, on estimating the pose of the UAVs based on the visual information collected by cameras located either on the ground or on-board, and also focuses on applying these proposed techniques for solving different problems, such as visual tracking for aerial refuelling or vision-based landing, among others. The different computer vision techniques presented in this thesis are proposed to solve some of the frequently problems found when addressing vision-based tasks in UAVs, such as obtaining **robust vision-based estimations at real-time frame rates**, and problems caused by vibrations, or 3D motion.

All the proposed algorithms have been **tested with real-image** data in on-line and off-line tests. Different evaluation mechanisms have been used to analyze the performance of the proposed algorithms, such as simulated data, images from real-flight tests, publicly available datasets, manually generated ground truth data, accurate position estimations using a VICON system and a robotic cell, and comparison with state of the art algorithms.

Results show that the proposed computer vision algorithms obtain performances that are comparable to, or even better than, state of the art algorithms, obtaining robust estimations at real-time frame rates. This proves that the proposed techniques are fast enough for vision-based control tasks. Therefore, the performance of the proposed vision algorithms has shown to be of a standard appropriate to the different explored applications: aerial refuelling and landing, and state estimation. It is noteworthy that they have low computational overheads for vision systems.

Nomenclature

Unless otherwise noted, the following notation is used:

- Scalar variables are denoted in lowercase, non-boldface type, e.g. x
- Vectors are denoted using lowercase boldface type, e.g. $\mathbf{x}$
- Matrix are denoted using uppercase boldface type, e.g. $\mathbf{X}$
- Images are denoted using uppercase boldface type, e.g. $\mathbf{I}$.

Acronyms

Description of the acronyms used in this thesis:

AAAR: Autonomous Air-to-Air Refuelling

CVG: Computer Vision Group

DOF: Degrees Of Freedom

FOV: Field Of View

FPS: Frames Per Second

GPS: Global Positioning System

IBVS: Image-Based Visual Servoing

ICIA: Inverse Compositional Image Alignment

IMU: Inertial Measurement Unit

KF: Kalman Filter

KLT: Kanade Lucas and Thomasi Algorithm

LK: Lucas Kanade Algorithm [115]

MAV: Micro-Aerial Vehicles.

ME: Mean Error

MSE: Mean Square Error

PBVS: Position-Based Visual Servoing

PID: : Proportional-Integral-Derivative controller

RANSAC: RANdom SAmple Consensus

RC: Radio Controlled

ROI: Region Of Interest

RMSE: Root Mean Square Error

SIFT: Scale-Invariant Feature Transform

SLAM: Simultaneous Localization And Mapping

SURF: Speeded Up Robust Features

UAS: Unmanned Aircraft System

UAV: Unmanned Aerial Vehicle

VTOL: Vertical Take-Off and Landing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Currently, there is great interest in the operations of manned and unmanned aerial vehicles for both civilian and military purposes. Operations conducted with those vehicles face different challenges and problems. In this thesis, we present contributions we have proposed that serve to overcome challenges of both manned and unmanned aerial vehicles. Because of their additional complexity and challenges, especial attention will be pay to Unmanned aerial vehicles (UAVs), also known as Unmanned Aerial Systems (UAS), or recently called by the U.S. Air Force as Remotely Piloted Aircraft Systems (RPAS).

Unmanned Aerial Vehicles (UAVs) were originally conceived with the purpose of reducing life risks in military missions. UAVs, which are also called drones, are formally defined as aircrafts without an on-board pilot. They are remotely controlled by a pilot from a ground station, or autonomously based on on-board sensors.

Although UAVs have been mainly used in military applications, in the last decades an increasing interest of using these vehicles in civilian applications has started to emerge, accompanied by an increase in the number of light UAVs (< 150 kg) offered in the market and a decrease of their price (e.g. AR. Drone €300 [7]). In this sense applications such as inspection, search and rescue (e.g. earthquake in Mirandola, Italy, 2012 [105]), surveillance [74, 191], aerial photography, and firefighting[147], among others, are some examples of the possible civilian market uses for these systems. As a CNN journalist wrote in June 2013, "skies over civilian heads will soon be busy with unmanned vehicles" [144].

The main advantage of using UAVs in those applications instead of piloted aircraft comes from the main objective that was considered when UAVs were created: "with UAVs there is no risk of loss of life". They can conduct repetitive flights during long periods

of time, dangerous maneuvers such as night operations, especially at low altitudes, or flights in hazardous areas such as contaminated areas [68], and all of these tasks can be conducted at low prices and without compromising human lives.

Nowadays, the main problem of the application of UAVs in the civilian market is related to the reliability of these systems to operate in complex environments (e.g. see&avoid capabilities or collision avoidance systems, secure data links, among others). Their immature reliability and the lack of aviation regulations of this kind of aircrafts make the incorporation of these systems in the civilian airspace not possible so far.

Although most of the commercial systems have solved the control problem, state estimation is still an open research area. Outdoors, commercial systems base the pose estimation on GPS and IMU (Inertial Measurement Unit) sensors. However, there are not reliable solutions for indoors navigation yet, but instead there are specific solutions to this problem that are based on laser [28], on vision [91], and on SLAM (Simultaneous Localization and Mapping) algorithms [12].

In this sense, computer vision is playing an important role in solving the indoors navigation problem [91], and is also playing an important role outdoors (e.g. collision avoidance [129]). In both scenarios, vision can be used as a main or complementary sensor to improve UAVs' capabilities (e.g. vision-based landing [161], visual inspection [138]), or to cope with vulnerabilities of other on-board sensors (e.g. pose estimation algorithms based on visual information to deal with GPS fallouts [29, 122], Inertial Navigation System -INS- drift). As a consequence of this, **vision-based algorithms to control UAVs have become an attractive and interesting research area** [113], where robust and fast pose estimation and visual tracking algorithms are still required to be developed. That is why the main goal of this thesis is to contribute to the state of the art by proposing strategies that allow the use of vision algorithms in the field of UAVs.

Vision for UAVs is the main research area at the Computer Vision Group of the Universidad Politécnica de Madrid (UPM) [52]. Different theses have been developed in the group [135, 140, 128] to present solutions to this challenging problem, based on the use of cameras as low cost sensors in order to obtain important information of the mission and of the vehicle's behavior, which combined with the UAV's low level controller allow to enrich UAVs' functionalities using the extracted visual information as a visual feedback to the flight controller.

This thesis deals with **two possible configurations of the cameras: a camera on-board the UAV, and an external multi-camera system** (a trinocular system located on the ground). In both configurations, visual tracking, pose estimation, and visual control algorithms are proposed. The thesis focuses on presenting novel solutions for solving the problem of tracking objects on-board UAVs, for estimating the pose of the UAV based on the visual information collected by cameras located either on the ground or on-board; and also focuses on applying these proposed techniques for solving different problems, such as visual tracking for aerial refuelling, or vision-based landing, among other applications.

1.1. Problem Statement

Understanding the 3D world information from the 2D visual information recovered by a camera is not an easy task. Factors such as illumination changes, point of view changes, hardware configuration, or motion, among other factors, can affect the way the information is projected in the image (e.g. they can affect the appearance of objects), and therefore can affect the algorithms used for understanding such information.

During the last 10 years at the Computer Vision Group of the UPM, research efforts have been focused on applying computer vision in unmanned aerial vehicles (UAVs). The development of visual algorithms for UAVs can be considered a challenging task. Real-time requirements (e.g. for control tasks), outdoors operations (non-structured environments), vehicle vibrations, and limited computational capacity on-board, are some examples of the problems found when working with UAVs.

One of the main problems when addressing vision-based tasks in UAVs is to obtain robust visual estimations at real-time frame rates (approximately > 8 frames per second, fast enough to close the control loop). Vision also has the disadvantages of high processing requirements and susceptibility to environmental conditions such as clouds, fog, and variable lighting conditions. If these difficulties are overcome, the recovered visual information can be used in a variety of vision-based applications (e.g. vision-based landing or visual inspection), where robust visual tracking, pose estimation, and control algorithms are required.

VISUAL TRACKING is the core of more complex tasks such as pose estimation and control. It consists in determining the trajectory of an object in the image plane [207]. In general terms, there are two ways to tackle the visual tracking problem: using indirect methods, i.e. using features [186]; or using direct methods [94]. In feature-based approaches, a sparse set of features of the object of interest are extracted and tracked through an image sequence by matching features between frames. Feature-based approaches are considered robust to scale changes, rotations and translations, and to illumination variations. However, the extraction of distinctive invariant features and the matching of these features between frames, at real-time frame rates, are known as difficult problems in computer vision that still need robust solutions to be proposed.

Conversely, direct methods estimate the motion of the object by minimizing an error function that is based on the image brightness (intensity values of the pixels) of the region of interest [181]. In direct methods, the detection and matching steps are merged. However, **the drawbacks of direct methods are that they rely on some constraints** [94] **that are difficult to preserve**, specially when working outdoors (e.g. brightness constancy constraint, spatial coherence, or small motion), and that their speed is highly dependent on the number of pixels in the template (e.g. in the region to be tracked or to be registered) and the registration method used, being it sometimes difficult to achieve real-time frame rates. That is why in the majority of situations, feature-based methods are preferable to direct methods. In order to handle some of the previously mentioned constraints, and to increase the robustness and efficiency of direct methods, multi-resolution (MR) approaches were proposed in [18], whereupon the use of direct methods becomes attractive, considering that they make it possible to handle large ranges of motion and also to improve the speed of the algorithm.

An important question at this point is to know **which method is the most suitable for our applications on-board UAVs: a feature-based method or a direct method?** In previous works at the Computer Vision Group of the UPM [36, 142, 141], we have used feature-based methods [186] to track planar scenes on-board UAVs. From these works, we have seen that in the application of tracking on-board UAVs the adopted feature-based strategies are very sensitive to strong motions (e.g. vehicle vibrations and fast 3D changes), being it difficult to find a compromise between achieving real-time and accurate estimations (defining a specific number of good features to track without increasing the processing time). Additionally, it has also been observed that when using feature-based methods under strong motions, the accumulation of errors makes the tracking algorithm fail after just a few frames, making on-line tests difficult.

Based on these results, **in this thesis direct methods [94] are explored as an alternative solution to solve the tracking problem in UAVs**. They have proved to obtain robust motion estimations (recovering motion with subpixel precision), due to the amount of information (every pixel in the image) used in the motion estimation process. Nevertheless, for using these methods in real-time applications, it is still required to find solutions to handle the constraints of direct methods (brightness constancy constraint, spatial coherence, small motion), and to improve the typical high computational demands of these methods. Additionally, all of the proposed solutions must be oriented to also solve the problems that arise when these kinds of methods are applied for tracking on-board aerial vehicles.

On the other hand, for some vision-based applications, not only robust tracking algorithms are required. In this thesis, **POSE ESTIMATION** techniques are proposed and used for what is known as visual odometry (estimating 3D motion based on image sequences by integrating the relative translation and rotation between frames) and for vision-based control tasks using visual servoing techniques (using visual information as feedback of control loops).

Visual pose estimation is the process of estimating the position and orientation of an object with respect to a reference frame, based on image information. Outdoors, pose estimation of UAVs is commonly accomplished through the intervention of GPS and IMU sensors. Such systems give position, velocity, and attitude information of the UAV that is used for positioning and navigation tasks. That is why any loss of GPS signal could cause serious problems in UAV operations. The GPS position is used to correct IMU (Inertial Measurement Unit) data from drift in order to obtain the UAV's state. However, GPS-based position estimation is not always an available measurement (e.g. indoors); and when it is available, it is not always an accurate and reliable measurement (e.g. at low heights). The waypoint navigation accuracy of our systems can vary between ± 1 and ± 2 m (DGPS is not implemented in our systems); and the altitude accuracy is around ± 0.5 m, under good GPS reception conditions. Although the previously mentioned values are considered sufficiently accurate for navigation in a variety of situations, **low height positioning tasks (< 3 m) such as landing require a more accurate and precise position estimation of the UAV**.

Vision-based pose estimation of UAVs is still an open research area [21]. Most of the algorithms proposed in the literature have been based on feature-based methods. **In this thesis, direct methods are explored to solve the pose estimation problem in**

UAVs. The pose information estimated by the visual algorithms will be used for vision-based applications. Therefore, the proposed algorithms should comply with the real-time requirements of that kind of applications.

A camera is a light-weight and a low-cost sensor which can be used as a main or complementary sensor for UAVs. Therefore, computer vision can play an important role to solve the pose estimation problem; and in the application concerned in this thesis, it can be applied in the following scenarios:

- When the GPS data is not available, another source of information is required to correct IMU drift.
- In some vision-based applications, such as visual inspection or landing, relative pose estimation between an object of interest and the UAV is required.
- For low height positioning tasks (< 3 m) such as landing, accurate position estimation of the UAV is required.

In sum, this thesis deals with computer vision for UAVs, where cameras can be used as a main or complementary sensor. The information recovered by this sensor is intended to be used for different vision-based applications, where robust real-time visual tracking, pose estimation, and visual control strategies are required. In all these areas there is still work to do; and proposing or applying tracking strategies capable of overcoming the problems mentioned in this section is still required. In this thesis, solutions to some of these problems are presented.

1.2. Thesis Objectives

The main objective of this thesis is **to provide UAVs with an additional source of information, based on vision, which is extracted by cameras located either on-board or on the ground**, in order to allow UAVs to develop visually-guided tasks, such as landing or inspection, especially in situations where GPS information is not available, where GPS-based position estimation is not accurate enough for the task to develop, or where payload restrictions do not allow the incorporation of additional sensors on-board.

This thesis covers three of the main computer vision areas: visual tracking and visual pose estimation, which are the bases of a third one called visual servoing, which, in our application, focuses on using visual information to control UAVs. In this sense, the thesis focuses on presenting novel solutions for solving the tracking problem of objects when using cameras on-board UAVs, for estimating the pose of UAVs based on the visual information collected by cameras located either on the ground or on-board, and also focuses on applying these proposed techniques for solving different problems, such as visual tracking for aerial refuelling or vision-based landing, among others.

The objectives formulated in this thesis can be summarized as follows:

1. To contribute to the state of the art by **proposing strategies** that allow the use of **computer vision algorithms in the UAVs field**.

2. To explore the use of **direct methods for vision-based applications for aerial vehicles**.

3. To present a **solution for tracking planar structures (or structures that can be assumed to be planar) using cameras on-board UAVs**, based on direct methods, that can deal with the problems of this kind of methods and be robust to large frame-to-frame motions. It should also be able to recover simple and complex motion models (e.g. the translation or the homography transformations) at real-time frame rates, and to recover information that can be used in different vision-based applications for UAVs.

4. To present **solution to estimate the pose of the UAV**. These solutions should comply with the real-time and accuracy requirements of the application (e.g. landing, aerial refuelling, visual odometry).

5. To apply **computer vision techniques for augmenting UAV capabilities**. This implies the proposal of applications and techniques where cameras can be used as a main or complementary sensor to guide UAVs.

1.3. Thesis Proposals

In the next paragraphs, proposals of this work to achieve objectives 3 to 5 will be described. Since **objectives 1 and 2** have a general nature, they are implicitly covered with the fulfillment of the other objectives.

1.3.1. Proposal Regarding Objective 3

"To present a solution for tracking using cameras on-board UAVs": visual tracking algorithms are the bases of many computer vision-based applications. From previous works conducted in this area at the Computer Vision Group [36, 142, 141], it has been seen that feature-based tracking strategies are very sensitive to strong motions, being it difficult to strike a balance between achieving real-time and accurate estimations. Direct methods, on the other hand, can be used as alternative solutions. However, the constraints of direct methods (brightness constancy constraint, small motion) and the high computational demands make direct methods far from being used in real-time applications such as the ones discussed in this thesis. In the literature, multi-resolution (MR) approaches [18] were proposed to help dealing with the small frame-to-frame motion constraint of direct methods. However, in our application, constant vehicle vibrations, low computational capacity available on-board, and delays in the communications (when images are processed on the ground), are problems that make the MR strategies insufficient to cope with the large frame-to-frame motion problem and therefore insufficient to allow direct methods to properly perform the tracking task using cameras on-board UAVs.

We propose a tracking strategy for planar structures (or structures that can be assumed to be planar) that is based on direct methods, taking advantage of the facts that such methods use the available information of every pixel in the image,

i.e. its intensity values, to estimate motion; and that as a consequence of this, robust motion estimations can be obtained.

In order to overcome some of the drawbacks of direct methods mentioned in Section 1.1, we propose to use the efficient Inverse Compositional Image Alignment Algorithm **(ICIA)** [14] **as image registration technique, to deal with the efficiency problem of direct methods**; and we also propose to extend this algorithm with a hierarchical strategy in terms of image resolution and number of parameters estimated in each resolution. We have called it: the Hierarchical Multi-Parametric and Multi-Resolution strategy (HMPMR). The **HMPMR strategy is proposed to improve the tracking task in situations where MR approaches are insufficient to cope with large frame-to-frame motions**. With the resultant algorithm, the **HMPMR-ICIA**, a solution to the small frame-to-frame motion constraint of direct methods and real-time frame rates are obtained.

1.3.2. Proposal Regarding Objective 4

"To present a solution to estimate the pose of the UAV": vision is an attractive light and low-cost solution to obtain the pose of the UAV when GPS information becomes unavailable (e.g. indoors) or unreliable (e.g. low heights), and also when relative position information with respect to an object of interest is required. **We propose to use vision for the pose estimation of UAVs using both on-board cameras and an external camera system**. We propose **three pose estimation approaches**: one for estimating the state of UAVs based on visual odometry using an on-board camera, a second one for estimating the relative pose between UAVs and an object of interest using also an on-board camera; and a third one for estimating the pose of UAVs using a ground multi-camera system.

The first algorithm is based on the information of an on-board camera to estimate the state of the UAV based on visual odometry. As can be seen in Section 2, most of the work presented in the literature in this topic makes use of features to determine the motion of the UAV (rotation and translation). However, why instead of using the appearance of the scene to derive features should we not use the appearance of all the pixels in the image to directly estimate the motion? Based on this question, **this thesis proposes a visual odometry algorithm based on direct methods**. We propose to use both images from an on-board camera and the HMPMR-ICIA algorithm to find the inter-frame motion, whereupon robust motion estimations at real-time frame rates can be obtained. The algorithm that is proposed recovers the motion between frames (rotation and translation) by decomposing the frame-to-frame homography obtained by the HMPMR-ICIA algorithm, which is applied to a patch that covers around 80% of the image. When visual estimation is required (e.g. GPS drop-out), this motion is integrated with the previous known estimation of the vehicle's state, obtained from the on-board sensors (GPS/IMU), and the subsequent estimations are based only on the vision-based motion estimations. Additionally, in order to deal with the high computational cost of the proposed approach, we propose to vary the number of pixels that intervene in the motion estimation process, in each MR level.

For the second algorithm, **we propose to use a pose estimation algorithm for**

planar objects, based on direct methods, in order to estimate the relative position of the UAV with respect to the object that is tracked. Many vision-based applications, such as landing or visual inspection, require relative pose estimations between an object of interest and the UAV. The pose estimation algorithm assumes that there is a visual tracking algorithm (we propose to use the HMPMR-ICIA) in charge of estimating the position of the object of interest in the image plane, and also assumes that the camera is calibrated and that the size of the object of interest is known. Therefore, with the 2D position of the object in the image plane and the known 3D position of the object, a homography transformation is calculated and decomposed in order to obtain the 3D position of the UAV with respect to the object that is being tracked. Thus, by using direct methods, robust relative position estimations at real-time can be obtained.

The third pose estimation algorithm that is proposed deals with the pose estimation problem for landing tasks. **We propose a ground multi-camera system for estimating the pose of UAVs**. The system is composed of three cameras located on the ground that are in charge of recovering the information of key features on-board the UAV in order to estimate the 3D position of the vehicle. The system requires robust feature extraction, tracking, and 3D reconstruction algorithms. To achieve this, we propose to use state of the art algorithms, and to focus on the problem of obtaining robust 3D information at real-time frame rates. The proposed multi-camera system focuses on landing tasks. This kind of tasks require a more accurate and precise position estimation of the UAV than the one obtained by GPS information (GPS horizontal and vertical accuracy of 2 m, and ± 0.5 m, respectively). With the proposed system, **we offer a novel low-cost vision-based platform for autonomous landing tasks** with the advantage that the installation of additional on-board sensors for conducting basic tasks such as landing is not required, and therefore the limited capacity on-board can be well exploited for other sensors needed for the mission.

1.3.3. Proposal Regarding Objective 5

"Objective 5: to apply computer vision techniques for UAVs": one of the goals of this thesis is to contribute to the state of the art by proposing strategies that allow the use of computer vision algorithms in field of UAVs. In this sense, in this thesis we propose to use vision for different applications: aerial refuelling, pose estimation, and landing. **We also use the different proposed algorithms with visual servoing techniques** to estimate references to the flight controller for achieving **visually-guided tasks**.

1.4. Contributions and Outline of the Thesis

This thesis shows how computer vision can play an important role to acquire significant information for UAVs. **The thesis focuses on presenting novel solutions for** solving the **tracking** problem of objects using cameras on-board UAVs; for **estimating the pose** of the UAVs based on the visual information collected by cameras located either on the ground or on-board; and also focuses on applying these proposed techniques for solving different problems, such as visual tracking for **aerial refuelling, or vision-based**

landing, among others.

Other important features of this thesis are that all the algorithms presented in this thesis have been **tested with real-image data**, and that different evaluation mechanisms have been used to analyze the performance of the proposed algorithms: simulated data, images from real-flights, publicly available datasets, manually generated ground truth data, accurate position estimations using a VICON system and a robotic cell, and comparison with state of the art algorithms.

Specific contributions of this work and the outline of the thesis are presented bellow.

Chapter 2: State of the Art

- A literature review of the computer vision techniques in the areas of visual tracking, pose estimation, and applications of computer vision for UAVs, is provided in this chapter. The contributions of this thesis to the state of the art are also outlined in this chapter.

Chapter 3: Visual Tracking

- Two state of the art visual tracking techniques are reviewed: the CAMSHIFT and tracking based on image registration. In this chapter, the 2D tracking-by-registration algorithm, based on direct methods, is introduced.

Chapter 4: Hierarchical Multi-Parametric and Multi-Resolution Tracking Strategy (HMPMR)

- A hierarchical strategy (**the HMPMR strategy**) for overcoming the small frame-to-frame motion constraint of direct methods is proposed.
- A detailed analysis of the HMPMR strategy, **characterizing the different components of this framework** (image pyramid, influence of the combination of parameters in the MR structure, and criteria to select the MR and the MP hierarchies) is presented.
- An extension of the ICIA algorithm [14] is proposed. The ICIA algorithm is extended with the HMPMR strategy (**the HMPMR-ICIA**). Therefore, robust real-time estimations are obtained.
- The HMPMR-ICIA algorithm is applied for **tracking objects using cameras on-board UAVs**. The advantages of using the HMPMR strategy for improving the tracking task are demonstrated in different scenarios and compared with state of the art algorithms.
- A visual tracking strategy for solving the **drogue tracking problem during Air-to-Air Refuelling tasks** is proposed. The proposed strategy is based on the HMPMR-ICIA. The robustness of the HMPMR-ICIA algorithm under the adverse conditions of the task (including rapid motions, large changes in scale, and significant occlusions) is demonstrated during different tests conducted in a realistic laboratory environment with actual flight hardware (a probe and a drogue) and simulated aircraft motion data, recreating an automated refuelling approach.

Chapter 5: Monocular On-Board Pose Estimation

- A **visual odometry algorithm**, for estimating the UAV state, based on direct methods, is proposed.
- The HMPMR-ICIA is applied to estimate real-time frame-to-frame homographies. Advantages of using the HMPMR-ICIA and the visual odometry strategy to estimate the pose of an aerial vehicle are demonstrated in tests with real data collected during flights.
- A position estimation algorithm is used to estimate the **relative position between a camera and an object of interest**. The algorithm is applied using a camera on-board a UAV; and also for Air-to-Air Refuelling tasks where fast, reliable, and accurate relative position estimations are required in order to achieve a successful capture of the drogue.

Chapter 6: Ground Multi-Camera Pose Estimation

- A novel external **multi-camera system** for estimating the pose of UAVs is proposed.
- Real flight tests demonstrate how the multi-camera system improves the position estimation, especially at low heights.

Chapter 7: Visual Servoing for UAVs

- Visual servoing techniques are reviewed and formulated for vision-based applications using cameras on-board UAVs.
- The HMPMR-ICIA is applied to track objects using a camera on-board a UAV. Visual servoing techniques are applied to conduct **vision-based positioning and landing tasks**.
- The novel external multi-camera system is presented as a **landing platform for UAVs**. 3D information extracted from the trinocular system is used to send vision-based commands to the flight controller.

Chapter 8: Conclusions and Future Work

- Conclusions about the different proposed techniques are presented, and the direction of future work is discussed.
- Publications derived from this work are presented.

Chapter 2

State of the Art

Computer vision implemented in UAVs covers object detection and tracking [133, 72], pose estimation [4], navigation [90, 1], obstacle detection [127], collision avoidance [131], autonomous landing [161, 121, 142], surveillance [36, 132], and Simultaneous Localization and Mapping SLAM [8, 34], among other areas in which vision is used as a main or complementary sensor to estimate the vehicle's state or to improve the vehicle's capabilities.

Most vision-based tasks are composed of 3 stages: the detection stage, in charge of initializing the task (detecting the position of the object of interest in the image plane and extracting the appropriate characteristics according to the task); the tracking stage, which is the core of most vision-based applications and is in charge of locating the object of interest in each frame; and the analysis stage, which is completely dependent on the application, and is in charge of answering the question of what to do with extracted visual information?

This chapter reviews algorithms presented in the literature concerning the tracking and the analysis stages. The algorithms that are reviewed correspond to the the three main areas studied in this thesis: visual tracking, visual pose estimation, and visual control. The latter area focuses on the use of vision for guiding UAVs. Special attention is paid to contributions that deal with aerial vehicles. Additionally, **this chapter presents the contributions of this thesis and indicates their relevance in the context of state of the art approaches**.

2.1. Visual Tracking

In the literature, different methods have been presented for tracking objects. The general idea of tracking is that, given an image sequence, the tracking algorithm must be able to identify the position of an object, either in the image plane or in the 3D space, by making assumptions about the objects, the camera, and the environment. Some of these assumptions are, for example, that the object is static, that the motion of the object in the image plane is due to motions of the camera, that the appearance of the object does not change, that the background is static, or the assumption that the scene is planar.

A general classification of tracking methods can be based on the space where the tracking task is conducted. It can be considered that there are two classes of such methods: 2D tracking and 3D tracking. In 2D tracking, the image projection of an object is followed, assuming that the 3D motion of the object can be modeled by a 2D transformation in the image plane (e.g. a translation, an affine, or a homography transformation). This kind of methods does not require the recovery of the actual position in space of the object, although it can be found with the 2D position estimated by the tracking algorithm based on some assumptions, e.g. knowing that the tracked object is planar [172, 30, 142].

On the other hand, 3D tracking algorithms directly recover the pose of the object (its 3D motion). Some of these methods rely on predefined markers such as [173, 159], and others on natural features such as edges [57], texture [46], or SIFT features [205]. These methods are based on the 2D-3D matching: matching 2D features with 3D features of the model (e.g. a CAD model), in order to estimate the motion of the camera or the object. Therefore, the position and orientation of the features can be known all the time [108].

Due to the amount of information contained in an image, tracking is conducted by imposing constraints in order to simplify the task. These constraints are based on all the knowledge we can have about the object to track (size, color, shape, motion, etc.) and about the camera (motion, calibration parameters, etc.). Therefore, if for example we know the object's color, the tracking tasks can be constrainted to track only a specific color, for instance.

The different approaches presented in the literature can be classified depending on how the following questions are solved [207]: which object representation is used for tracking? Which image features are used? How the motion appearance and shape of the object are modeled? The answers to all these questions depend on the application (off-line, on-line); on the kind of object to track; on the degrees of freedom of the object or of the camera; or on who is moving: the object, the camera, or both. In [207], a good survey of tracking algorithms is presented.

There are different ways the tracking task can be conducted. Two common ways are: based on image registration (tracking-by-registration), and based on detection algorithms (tracking-by-detection) [101]. In [50], the term tracking-by-registration was introduced, and although tracking and registration (estimating the transformation that aligns two images of the same scene estimating pixel-to-pixel correspondence [216]) are usually seen as two different strategies, they can be used simultaneously for object tracking, in which case the object of interest is tracked by iteratively registering pairs of consecutive images.

Image registration algorithms can be classified according to the kind of information used: based on direct methods (based on shape and appearance, e.g. a template) [94, 26];

or based on features (e.g. points) [186, 114]. Feature-based approaches [186] extract local features such as lines, edges [143, 215], or points [80, 212, 114, 16], among other features, and track them by matching features between frames. These methods rely on the detection of distinguishable image features and the matching method employed (e.g. such as RANSAC [64, 179]). They tend to be robust to scale changes, rotations and translations, and to illumination variations.

Conversely, direct methods [94], also called pixel-based methods, estimate the transformation that aligns the images by minimizing an error measurement that is based on the intensity values of the image. In this thesis, a tracking-by-registration algorithm based on direct methods is used not only for tracking objects but also for more complex tasks such as pose estimation and control.

The image registration process consists in aligning two images: the reference image (that contains the object to track) and the current image, by finding the transformation that best aligns them. This transformation is normally found iteratively by minimizing the sum of squared differences (SSD) between the reference image and the current image [181] using different minimization methods. Nonetheless, the gradient descent optimization method (based on a first order Taylor series approximation of the SSD) is one of the most widely used approaches because of its efficiency [15, 115, 78].

Gradient descent approaches can be classified depending on the update rule of the parameters. In [15], the different approaches were classified as follows: forwards additive [115], forwards compositional [171], inverse additive [78], and inverse compositional [14]. In the inverse compositional approach, the roles of the image and the template are inverted, whereupon a more efficient algorithm is obtained because the Jacobian of the warping function and the inversion of the Hessian matrix are calculated once at the beginning of the tracking task and every time the template image changes (not in every iteration, as the forwards additive approaches do).

Under large frame-to-frame motions, these image registration techniques require the use of hierarchical methods to help finding the transformation that aligns the reference and the current images. We have classified the different hierarchical approaches found in the literature in 5 classes, according to the kinds of hierarchies they have. Some of these approaches use feature-based methods, while others use direct methods. They can be classified as follows:

- The Multi-Resolution approach (MR): in this hierarchical approach, the same motion model is estimated in each level of the image pyramid (the MR pyramid). This scheme has been used for different applications: tracking planar objects [118], detecting and tracking moving objects [96], registration [190], and recovering scene structure [79], among other applications [106].

- The Multi-Parametric approach (MP): in it, a hierarchy of parameters is applied using the same resolution of the image. In the different fields where this strategy has been used, the hierarchy of 2D transformations has always been employed. With this strategy, an initial registration is found by first recovering the translation motion model, in order to reduce the separation of the images; and then the estimation is refined by increasing the complexity of the motion model progressively [38, 96, 157].

- Multi-Resolution with priors (MRp): in this strategy, before applying the MR approach, an initial estimation of the parameters is obtained applying either an initial algorithm [188] or information from additional sensors [93, 169].

- The Hierarchical Multi-Parametric Multi-Resolution approach (HMPMR): in this approach, different parameters are estimated inside an MR pyramid. In [164], this strategy is used for image mosaicing. They use an MP scheme and an MR pyramid. The number of pixels employed in the optimization process depends on the number of parameters estimated. In the case of translation, a region of interest (ROI) of around $\frac{1}{3}$ of the input image is used. For the affine motion model, $\frac{2}{3}$ of the image are used; and the full image is used for the homography.

- Hierarchical Multi-Parametric Multi-Resolution with priors (HMPMRp): where an initial estimation of the motion is obtained by either an initial algorithm or by using additional sensors. Examples of this strategy can be found in [165] and [213] for building image mosaics. In [165], the normalized cross correlation (NCC) method is used to find an initial 2D translation, whereas in [213] the 2D translation is found using the phase correlation method. In both methods, the initial translation is used as an initial estimation of a coarse-to-fine search of more complex motion models in an MR pyramid. On the other hand, in [58] the global motion from image sequences is estimated by first finding the 2D translation using an n-step search matching algorithm, and then using that estimation in an MR pyramid to refine the estimation.

According to this classification, in this thesis we propose an HMPMR strategy to deal with the large frame-to-frame motion for tracking objects on-board UAVs. Visual tracking is a difficult task, especially in our application, where there can be partial or full occlusions of the object, illumination changes, 3D motions, and real-time processing requirements, among other problems. That is why, the selected tracking methods must overcome these challenges to be able to be used in our application field.

Robust visual tracking at real-time frame rates is required by many online tasks on-board UAVs, such as pose estimation, or target following (see Figure 2.4). In the literature, different strategies have been presented to solve the tracking problem in aerial images. Most of those strategies are based on feature-based methods [208, 141, 132, 183, 39, 72], and just a few have explored the use of direct methods [36, 53].

In [161], a helipad (letter H) is tracked from a camera on-board a UAV. The segmentation of the helipad is based on image thresholding, image filtering, and on finding connected components. The segmented helipad is recognized based on comparing current invariant moments with stored ones (found with images collected in prior flights). The respective tracking algorithm was implemented for landing tasks in a helicopter. In [132], the Lucas-Kanade algorithm is used to track four points correspoding to the four corners of a window. The center of the window is used to send vision-based references to the flight controller in order to center the window in the image plane, by moving the helicopter using image-based visual servoing techniques (see Figure 2.2(a)). Similarly, in [141] a feature-based tracking algorithm is used to track a helipad placed on the ground. The helipad was designed to contain distinctive features. The helipad is manually selected

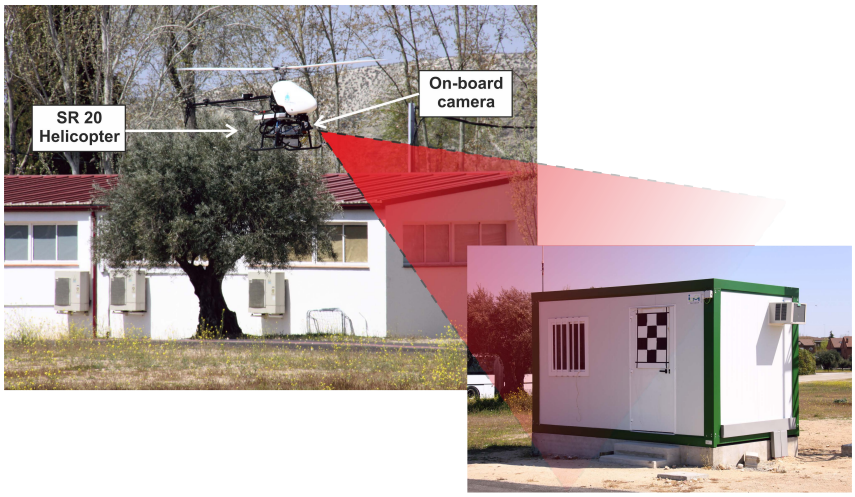
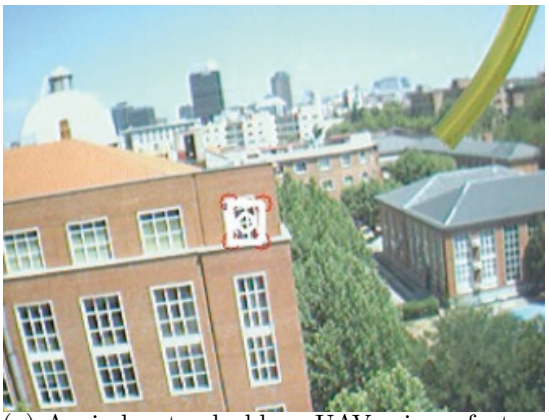


Figure 2.1: Tracking with cameras on-board UAVs. Robust real-time tracking allows to expand the vehicle's capabilities, such as those related to landing, visual inspection, or coping with vulnerabilities of other on-board sensors (e.g. GPS drop-outs).

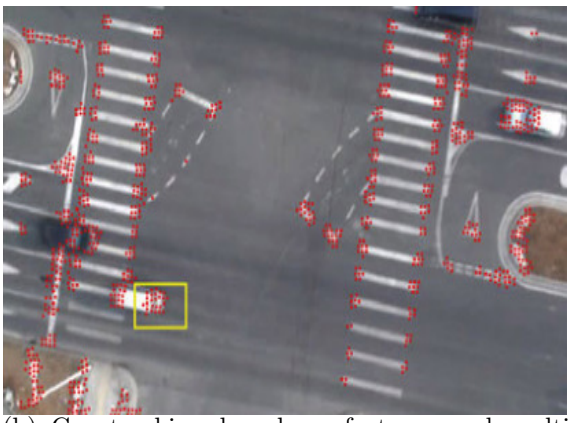
in the first frame, and the KLT algorithm [22] is in charge of tracking it (12 FPS). This tracking algorithm was proposed for vision-based landing tasks using a camera on-board a UAV.

On the other hand, in [53] a template-based tracking algorithm based on direct methods for tracking planar templates was presented. The algorithm uses mutual information to align the images, and was tested off-line with different data sets. Results show that the tracking algorithm is robust and can be applied for real-time applications, such as tracking vehicles in aerial images. In [39], a multi-motion layer analysis is proposed to detect and track vehicles from airbone videos (see Figure 2.2(b)). The KLT algorithm is used for detecting and tracking the objects. Images are registered based on the KLT algorithm and then image differences are used to detect moving objects in the images. When the objects are detected, vehicle motion layers and background layers are created and updated in every frame. In [73], a vision-based line tracking algorithm to estimate the 3D position and orientation of a mini-rotorcraft is proposed. The algorithm estimates the 3D pose based on the vanishing points of the line. A forwards-looking camera is used to track the line, and a downwards-looking camera is used to estimate the translational velocity based on optical flow. Results have shown that the tracking and control algorithms ensure the tracking of the visual reference.

Most of the tracking approaches presented in the literature, as the ones shown in Figure 2.2, are based on features, and some of them use MR [118] and MRp (multi-resolution with priors) [93] strategies. In this thesis we propose to address the tracking problem with direct methods [94] using the Inverse Compositional Image Alignment Algorithm (ICIA) [14], but extending it in a hierarchical strategy in terms of image resolution and number of parameters estimated in each resolution: it is thus a Hierarchical Multi-Parametric and Multi-Resolution strategy (HMPMR). Using this strategy, the tracking task is improved in situations where MR approaches are insufficient to cope with large frame-to-frame motions without compromising the real-time operation required in on-line applications.



(a) A window tracked by a UAV using a feature-based tracking. Image from [132].



(b) Car tracking based on features and multi-motion layer analysis. Image from [39].

Figure 2.2: Literature review of visual tracking on-board UAVs.

To our knowledge, in the literature this strategy has neither been presented for solving the tracking problem nor for on-line tracking on-board autonomous vehicles. In [165], the HMPMR strategy has been applied for frame-to-frame alignment in image mosaicing. However, there are no evidences of why this strategy was adopted in that area, what criteria were considered for configuring it, or what the advantages of using it are.

2.2. Visual Pose Estimation

Pose estimation is the process of estimating the position and orientation of an object with respect to a reference frame. This estimation can be accomplished based on fusing information from several sensors (e.g. GPS, IMU, Vision, laser, or other sensors). Proposed visual pose estimation methods depend on the kind of visual information that is used to estimate the pose (e.g. based on planar structures [30], on landmarks [195], or based on the detection of the horizon [60]), and on the number of cameras used for estimating this pose (monocular, stereo, multi-cameras [121], [195]). Pose estimation is an essential technique for several vision-based applications, such as Human Computer Interaction [61], vehicle navigation [166], SLAM (Simultaneous Localization and Mapping), etc.

In this thesis, among other things, two pose estimation techniques based on image information are proposed and used in the field of aerial vehicles. The first one is used for what is known as visual odometry [166], estimating 3D motion based on image sequences by integrating the relative translation and rotation between frames. The second technique is used to obtain relative position estimations of the UAV with respect to an object. This technique is used for vision-based control tasks using visual servoing techniques [43]: i.e. using visual information as feedback of control loops.

For outdoors operations, most UAVs use GPS position to correct IMU (Inertial Measurement Unit) data from drift, in order to obtain the UAV's state. That is why any loss of GPS signal will cause serious problems in the UAV operation. GPS estimation is not

a reliable measurement, and indoors GPS signal is unavailable. For this reason, visual navigation is becoming the main topic of research in different robotic vehicles: UAVs, Autonomous Ground Vehicles (AGVs), and Autonomous Underwater Vehicles (AUV). Visual odometry approaches have been proposed as one of the possible solutions for the problems that arise when GPS information is unreliable (e.g. when flying close to obstacles, or during GPS dropouts) or when it is unavailable (e.g. indoors).

Visual odometry is the process of estimating the egomotion (3D motion) of an agent using visual information from one or multiple cameras [166]. It is based on the detection of the apparent motion of the scene, and focuses on the estimation of the 3D motion sequentially (every time a new frame arrives), estimating pose after pose in real-time. Visual odometry has been used in the NASA Mars exploration program [45], and for ground vehicles [146], aerial vehicles [48], and underwater vehicles [49].

In aerial vehicles, the different approaches presented in the literature that use vision as an additional or complementary sensor to estimate the UAV's state, based on visual odometry, can be classified according to the type of information that is recovered to estimate the vehicles' position and orientation. Two types can be identified: those that use features to obtain the UAV's state; and the ones that use the pixels' information (direct methods). Additionally, other classifications can be based on the number of cameras used for estimating the motion: monocular (motion can be estimated up to a scale factor) [33, 48], or multi-camera; and on the motion estimation method used to extract the vehicle's motion, for example when the scene is planar or can be assumed planar and the intrinsic parameters of the camera are known, homography decomposition techniques are commonly used in monocular systems to extract the camera displacement, e.g. [198].

In [33], a homography-based visual odometry technique (assuming planar scenes) is presented. Matched corner features are used to calculate a frame-to-frame homography, and the homography decomposition technique is used to obtain two solutions of the motion estimation. The correct solution is obtained by analyzing the homography with a third view. The algorithm has been evaluated off-line using images from real-flight tests. Another example of a homography-based visual odometry is the work presented in [100]. It is a feature-based pose estimation algorithm for piecewise planar scenes. In it, during a GPS dropout previous GPS data is linked with image data to provide inertial measurements; and simulation results illustrate the performance of the algorithm. On the other hand, in [5] a homography matrix is used directly -without decomposition- in a kalman filter to fuse GPS, IMU, and visual data to obtain the state of a MAV (Micro Aerial Vehicle). Corner features are used to calculate the homography matrices. Both simulated and real data are used to evaluate the algorithm.

In [117], a monocular system based on features is used to estimate absolute position and velocity data when GPS data is unavailable. The algorithm is based on the estimation of the 3D position of features when GPS data is available. The 3D positions and visual line-of-sight measurements of these features are used to estimate the pose when GPS is unavailable. These line-of-sight measurements are used to triangulate the absolute pose of the vehicle. The algorithm has been tested in simulations and in real-flight tests. In [200] (see Figure 2.3), a vision-based strategy for autonomous navigation is presented. The algorithm is based on a camera and inertial sensors, and in order to reduce drift and GPS dependency a visual SLAM algorithm is implemented. The proposed algorithm

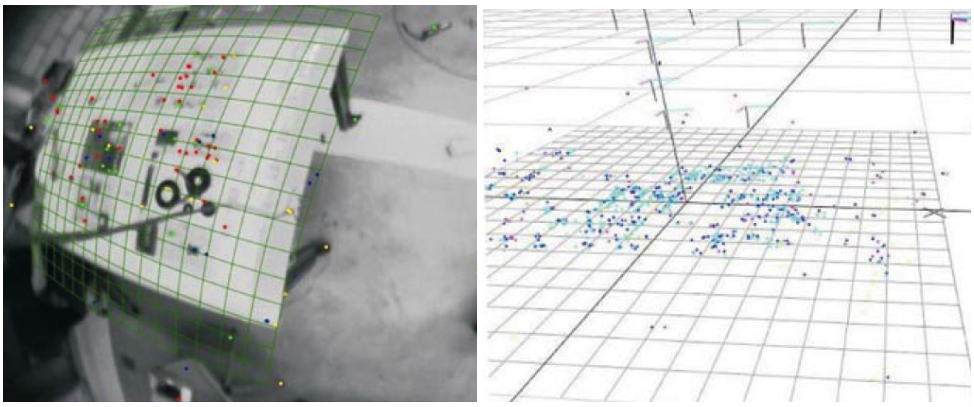


Figure 2.3: Visual SLAM for autonomous indoors navigation. The left image is a screenshot of the visual SLAM algorithm. The right image shows the generated 3D map. Images from [200].

permits to use vision as a main sensor to navigate through an unknown GPS-denied environment, independently of any external artificial aids. Results show that the UAV can conduct autonomous take-off, set-point following, and landing tasks. The pose estimation is obtained with a precision of even a few centimeters.

Visual odometry is also accomplished by matching visual data with a reference map (a terrain map). In [48], a monocular vision-based navigation method that offers a drift-free estimation when GPS signal is unavailable is presented. The algorithm uses two techniques: a position estimation based on visual odometry (using corner features) that drifts, and an image registration technique that matches the current image with a geo-referenced map in order to compensate the drift. The position estimation is derived from the homography matrix considering that the attitude data is obtained from the IMU, and the distance to the plane is obtained from an altimeter. The algorithm has been tested off-line with real flight data, as well as on-line. Off-line results showed a maximum position error of 5 m and a maximum velocity error of 1.5 m/s when comparing visual estimations with GPS/INS measurements. In [112], a vision-based geo-referencing system to reduce long term drift is presented. The visual information is classified and matched against a classified visual information in a pre-existing map. With this matching, absolute locations of the UAV are estimated. The system was tested off-line with real-flight data.

Other strategies are based on binocular systems. The advantage of these systems is that they permit to know the scale factor. One of the first works on vision-based position estimation is the one presented in [4]. The visual odometer system depends on tracking a ground object with stereo cameras. A template matching algorithm is in charge of estimating the relative motion (lateral and longitudinal). This information is fused with inertial data and also with range information obtained with the binocular system, in order to obtain the 3D motion of the UAV. Another stereo visual odometer was proposed in [102]. Vision-based data and inertial data are fused using a Kalman Filter. Features are detected and tracked with the KLT feature-based algorithm, but in this case their 3D position is found by triangulation.

Some widely used pose estimation systems are the motion caption systems, such as VICON [195]. External cameras located inside a flight area are in charge of estimating the pose of the UAV by detecting infrared marks located on the UAV [88]. Fast estimations

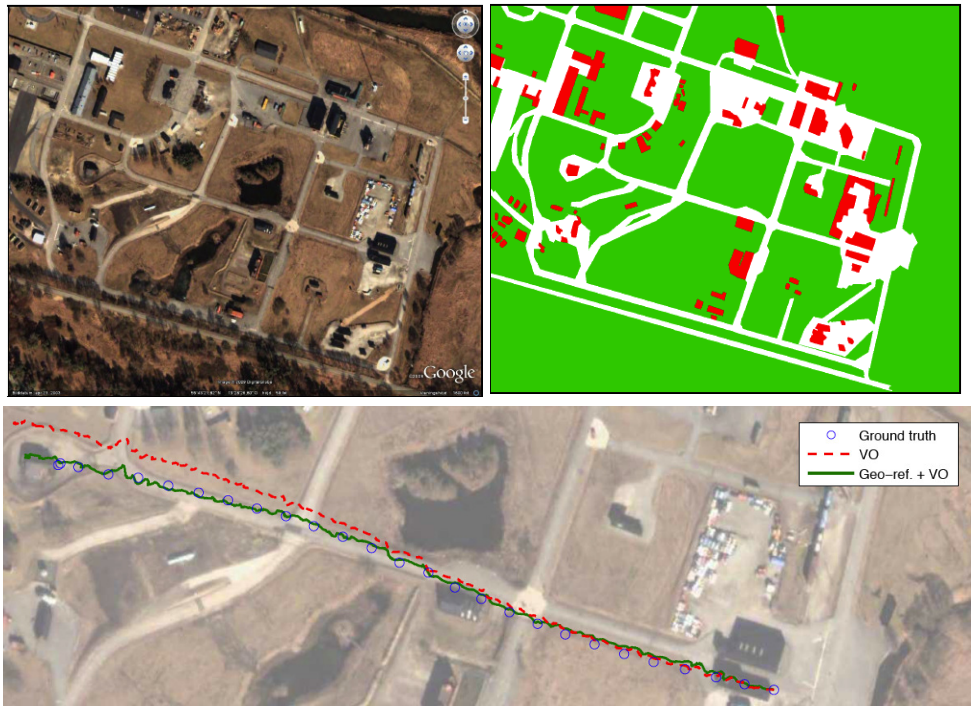
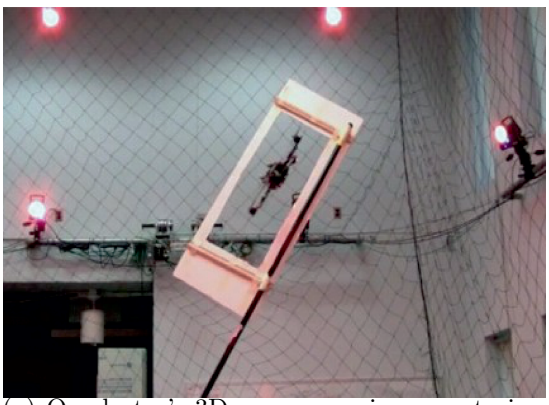


Figure 2.4: Vision-based pose estimation using geo-referenced maps to reduce drift. The left image shows the pre-existing map, and the right image shows the manually classified map (grass, hoses, etc.). The bottom image shows the estimated trajectories. Images from [112].

(100 Hz) with millimeter accuracy are obtained with these systems. However, with these systems, the workspace is limited by the cameras' field of view, and the systems must be calibrated when the position of the cameras changes. In the literature, the VICON system has been used as ground truth of vision-based algorithms [123, 148], and also as state estimation for the flight control algorithm. In the latter case, results from aerobatic and aggressive maneuvers (see Figure 2.5(a)) are presented in [134], and also for cooperative tasks [139, 111].

As can be seen, most of the work presented in the literature makes use of features to determine the homography relationship between images. Under some circumstances, feature-based methods tend to be preferable to direct methods. One of the reasons of this is that direct methods rely on some assumptions, that in some situations are difficult to preserve, specially when working outdoors (brightness constancy constraint, spatial coherence, and small motion).

Motivated by this fact, **in this thesis we address the pose estimation problem using image registration and homography decomposition techniques, but using direct methods instead of feature-based methods**. In [158], a direct method was used in an inertially-aided visual odometry system. However, as the authors say, their results were presented as a proof of concepts, and additional improvements must be incorporated in order to achieve a real-time operation of the algorithm. In this thesis we propose to expand the use of direct methods to obtain real-time pose estimations, taking into account that direct methods can generate more accurate results in the estimation due to the amount of information that is considered in the evaluation of the motion model.



(a) Quadrotor's 3D maneuvers in a constrained environment. Image from [39].

(b) A new paradigm for construction based on quadrotor teams. Image from [111].

Figure 2.5: Multi-camera pose estimation. Approaches based on a VICON system. Pose is estimated by detecting infrared marks at 100 Hz with millimeter accuracy.

To achieve this, we propose a pose estimation strategy based on the decomposition of the homography recovered by the HMPMR-ICIA algorithm, which is used for registering pairs of images. Therefore, the proposed algorithm extends the results obtained in [158], where an algorithm based on direct methods was proposed but its complexity did not allow its real-time operation.

A second pose estimation algorithm is used in this thesis for obtaining relative position estimations which are later used for vision-based tasks. One of the most common tasks that requires relative position estimations is the landing task. The literature review of these approaches is presented in the next subsection, which is dedicated to vision-based applications on-board UAVs. In [89], an omnidirectional vision system is used for positioning tasks performed with an autonomous helicopter. A customized target is detected by Hu's moments. The distance to the target is found based on the area of the target in the image. With this data, vision-based commands are sent to move the helicopter towards the target. An image-based visual servo control algorithm for stationary flight is presented in [76]. An on-board camera is used to detect a target, and the centroid of the target is used to define image-based visual servoing tasks in order to maintain a small quadrotor hovering. On the other hand, in [142] features extracted from a planar pattern located on the ground are used to calculate a frame-to-frame homography. In every image, the 3D relative position between the UAV and the pattern is estimated by composing the relative homographies from the current to the first frame; and then by multiplying them with the homography found between the world and the camera reference frames in the first frame, which is obtained assuming that the dimensions of the pattern are known (3D-2D matching). The final homography is decomposed in order to obtain the relative positions and orientations.

In this thesis, a method similar to the one presented in [142] is presented in Section 5.3, and is used to obtain relative position estimations. The main differences with that proposal are that the tracking algorithm proposed herein is conducted using the HMPMR-ICIA algorithm, which is based on direct methods and permit to conduct the

tracking task robustly, especially under large frame to frame motions; and also that the pose estimation is not conducted composing the homographies from the current to the first frame, but instead in every frame the 2D image coordinates of the object found by the HMPMR-ICIA algorithm are used to calculate the homography between the world and the camera reference frames, which is then decomposed in order to estimate the current pose at real-time frame rates (the frame rate reached by the algorithm depends on the tasks presented in Sections 5.5 and 5.6).

2.3. Vision for UAVs

Image information has been used for different purposes in the field of aerial vehicles. The following paragraphs enlist applications where vision has been used as a main or complementary sensor for conducting such tasks. Some of those applications include: collision avoidance, e.g. [131], where an on-board camera is used to detect potential aircraft collisions for fixed wings aerial vehicles; surveillance by detecting and tracking objects from a UAV [36, 132]; autonomous vision-based take-off and landing tasks [161, 121, 142]; SLAM (Simultaneous Localization and Mapping) [8, 34, 200]; aerial photography [77]; traffic monitoring [83, 154]; attitude estimation based on horizon detection and segmentation [60, 185]; and firefighting [136], among others.

Vision for unmanned aerial vehicles is a constantly growing area. In the last years, a wide variety of approaches that combine different UAV platforms, different sensors, and different applications have been proposed. In the literature, different surveys that try to summarize, organize, and classify the works presented in the literature have been presented [149, 113, 103]. Among them, the most complete paper is the one presented in [103], which describes the state of the art in autonomous rotorcraft UAVs and highlights challenges that still need to be addressed.

In this thesis, we focus on two applications where vision is used to conduct aerial refuelling and landing tasks. In the following sections, we present a literature review of these applications.

2.3.1. Aerial Refuelling

Aerial refuelling, also referred to as air-to-air refuelling (AAR), was first developed in the 1920s. Two approaches are currently used for aerial refuelling [176]: the flying boom, where a retractable boom is extended to be connected to the fuel receptacle of the aircraft to be refueled; and the probe and drogue method, where a flexible hose with a drogue attached to its end is extended and the receiver aircraft maneuvers to insert the probe into the drogue.

In unmanned aerial vehicles (UAVs), aerial refuelling capabilities offer significant benefits. Refuelling operations have historically been conducted as a piloted operation demanding a high level of training and fast reactions, and thus are not appropriate for remotely piloted aircraft controlled over relatively slow data links. The development of capabilities to perform that task relies on two key technologies: position sensing and tracking, in order to allow the receiver aircraft to determine the relative position of the

refuelling drogue; and control strategies, to enable a robust and safe approach and coupling. Due to the inherent difficulty of the task, fast, reliable and accurate relative position estimations are required in order to achieve a successful capture of the drogue.

For position tracking, machine vision in UAVs is a popular method for collating position data. Advantages of using vision systems for AAAR include the potential for installation without modifications of the target aircraft being required; and increasing precision with proximity to the target. Disadvantages of vision systems can include high processing requirements and susceptibility to environmental conditions such as cloud, fog, and variable lighting conditions. In the literature, there have been a variety of computer vision solutions for the aerial refuelling problem. A review of computer vision techniques applied to aerial refuelling until 2012 is found in [109], and here we summarize some relevant works.

In [152], the drogue position is estimated using an infrared camera placed in the receiver aircraft and infrared leds placed on the drogue structure. The pose of the drogue is estimated matching the 2D position of the leds with their known 3D positions in the drogue. Synthetic images are used to test the algorithm. On the other hand, a set of beacons mounted on known positions in the drogue are used in [193] (see Figure 2.6). These beacons are detected by a VisNav sensor placed in the receiver aircraft. A communication link between the sensor and the beacons allows the receiver aircraft to triangulate the pose of the drogue with updated rates of 100 Hz.

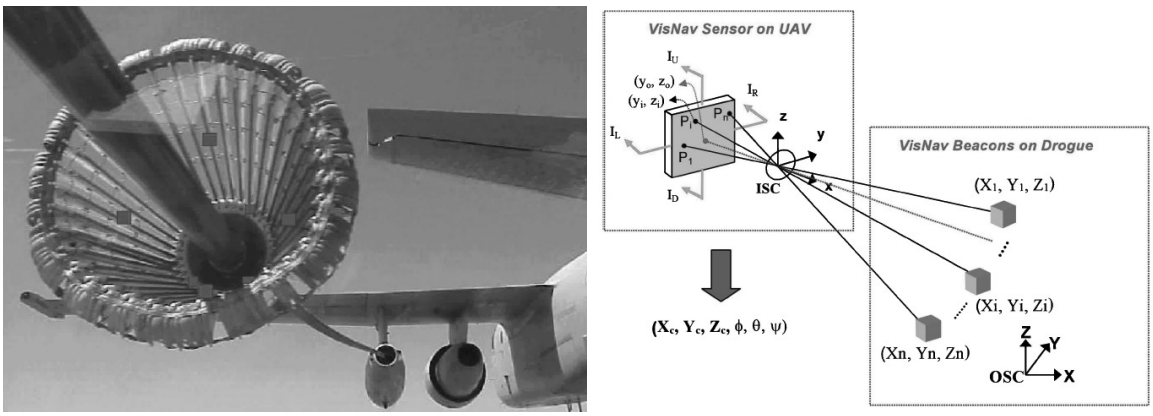


Figure 2.6: Drogue tracking and pose estimation based on VisNav sensor. A set of beacons are mounted on known positions in the drogue. The VisNav triangulates the pose of the drogue (100 Hz). Images from [193].

Passive systems, on the other hand, do not require active cooperation from the target. For the flying boom refuelling method, a vision system based on deformable contour algorithms was proposed in [56] to obtain relative 3D position estimations. A camera was placed on the tanker aircraft looking down towards the receiver aircraft, and it captured images of a passive target painted near the refuelling receptacle. The HSV (hue, saturation, and value) color space was used to increase the robustness under variations in lighting conditions. Synthetic images were used in this case to test the performance of the algorithm. Experiments verify that visual snakes can provide relative position measurements at rates of 30 Hz. Spencer [176] used a corner detection algorithm to extract

both structural and painted features on a tanker. For each video frame, the detected features were compared with known features of the tanker in order to compute 3D pointing vectors, which were used in a Kalman filter-based navigation algorithm to determine the relative position of the tanker.

On the other hand, also concerning the flying boom refuelling method, Vendra *et al* analyzed the performance of well-known corner detectors (SUSAN and Harris) for the use of a machine vision-based approach in the UAV Aerial Refuelling problem [194]. A camera was placed in the receiver aircraft looking upwards, capturing the tanker aircraft, and the feature extractor algorithms detected corners of the tanker aircraft. These corners were matched with a set of known physical features of the tanker (2D-3D match) using a detection and labeling algorithm, and the positions of the matched corners were used by a pose estimation algorithm to evaluate the position and orientation of the receiver aircraft with respect to the tanker aircraft. The paper analyzes the robustness of the corner detection algorithms in the event of image noise, variations in image contrast, and motion blur; and confirms the capabilities of corner detection algorithms for interfacing with detection and labeling, and with pose estimation algorithms in the AAAR problem.

Evidently, most of the existing vision-based approaches for autonomous aerial refuelling have made use of features such as corners, painted marks, and LEDs to estimate the relative position of either the receiver or the tanker aircrafts. Not only do these methods often require the installation of specific hardware, but they may have problems caused by the occlusion of one or more of those features. In this area, this thesis proposes the use of direct methods and hierarchical image registration techniques to solve the drogue tracking problem for autonomous aerial refuelling. **To our knowledge, In the literature direct methods have not been proposed to solve the drogue tracking problem**. In addition, whilst a limited set of studies have tested vision systems under realistic operating conditions, many have relied on simulated visual data to test the algorithms. This thesis proposed and evaluates a vision system for probe and drogue refuelling using a single camera in conjunction with real, full-scale aircraft hardware in a laboratory test environment. The real-time vision-based strategy is based on direct methods and image processing techniques and differs from previous approaches because, by using direct methods, installation of specialized hardware or software is not required, and has the additional advantage that under partial occlusions of the drogue, the algorithm is able to continue with the tracking task. The test environment, developed in Bristol University [20], comprises a robotic cell that simulates the tanker and receiver aircrafts. A drogue is attached to the free end of one robot, and a refuelling probe is attached to a second, track-mounted robot.

2.3.2. Landing

Vision-based autonomous landing for UAVs has been a very active research area, not only for VTOL (Vertical Take-off and Landing) UAVs, but also for fixed wings. Some of the approaches have been sought to detect safety areas to land, and others to detect customized patterns (specific geometry and color) to estimate the relative pose between the vehicle and the area where the UAV should land. Additionally, some of the approaches are based on on-board image data (monocular or binocular) and others are

based on cameras placed on the ground (monocular and multi-camera systems). Most of their results correspond to off-line and simulated tests where the performance of vision algorithms is analyzed, and just a few have conducted real vision-based landings.

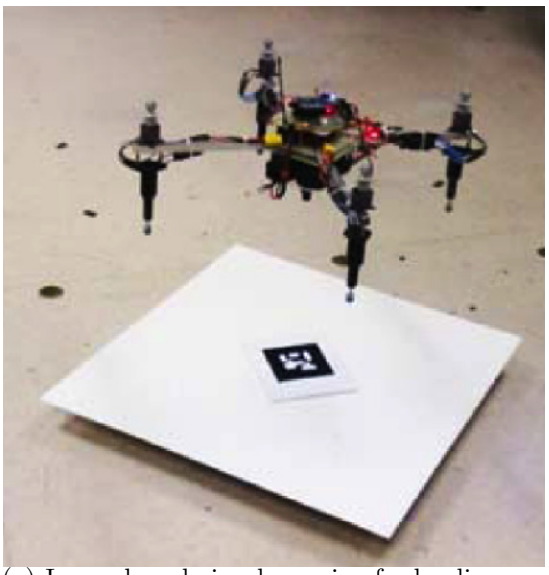
Most of the works presented in the literature use on-board cameras; and for VTOL UAVs, most of these works use customized helipads in order to simplify the detection and the tracking of the helipad in every frame [161, 141, 132, 39].

In [168], a landing approach based on a landing target strategically designed to facilitate the segmentation and feature extraction processes is presented. The algorithm is tested in a stationary flight, obtaining accuracies of 5 cm in position and 5° in orientation. This proposal was extended in [167] using a multiple view approach with a single camera. In that work, vision-based relative position estimations were used to achieve a vision-based hovering above the landing target, but not an automatic landing.

In [162], a successful autonomous landing of a UAV is achieved using the information of an onboard camera and a sonar sensor. The landing strategy is divided into two steps: the first one uses inertial moments as shape descriptors to recognize the helipad, and extracts its position and orientation to move the helicopter over it. Then, in the second step, the helicopter descends using the information of the sonar sensor. The helicopter achieved autonomous landing with a position accuracy of 40 cm and 6° in orientation, both measured in relation to the helipad. This work is then complemented in [160], where the behavior of the algorithm is analyzed in situations where the helipad is momentarily hidden and where the object to track is moved. In the latter situation, the helicopter starts its landing process when the target has stopped. In [137], a Kalman filter is used to fuse information from an on-board camera and an IMU. The vision-based system detects a designed landing pad (black circles with a white background) and estimates the relative pose. Different successful tests were achieved with a precision of 42 cm. In other works, an onboard camera, in conjunction with a ground moire pattern, is used to estimate the vehicle's six degrees of freedom for landing purposes [187]; and in [107]an image-based visual servoing strategy is used to control a quadrotor for tracking and landing on a moving platform.

Stereo vision has also been used to detect the landing pad. In [204], a stereo vision-based pose and motion estimation algorithm to land a UAV on a moving target is proposed. The landing target was designed to facilitate the feature extraction and matching processes. The center of the target is found in the left and right images, and its 3D position with respect to the helicopter is calculated by triangulation. The results were considered accurate enough for a future incorporation of the algorithm in the UAV control loop.

Identifying a safe place to land was the purpose of the work presented in [70]. In that work, a safe landing site detection algorithm for unknown terrains is presented. The algorithm looks for areas clean and big enough to land, assuming that the local ground plane is horizontal and has obstacles on it (cars, boxes, people, rocks, etc) and that the image shows a higher contrast in the boundaries of the obstacles as compared to the boundaries of the terrain texture. A contrast threshold is used to detect the obstacle zones, so that all the circular areas in which the pixels have a level of contrast below the threshold are selected as possible landing zones. The algorithm was tested with experimental aerial images, leaving for future work the incorporation of the algorithm in



(a) Image-based visual servoing for landing on a designed target. Image from [107].

(b) Landing site selection based on a stereo range map of the terrain. Image from [184].

Figure 2.7: Multi-camera pose estimation. Approaches based on a VICON system. Pose is estimated by detecting infrared marks at 100 Hz with millimeter accuracy.

the vehicle's control loop.

On the other hand, in [98] an autonomous landing based on the information of the GPS, a LIDAR sensor, and an onboard camera was achieved without using predefined targets. That work focused on the identification and avoidance of hazardous features, and on the estimation of the velocity and position relative to a landing site, using the information of an onboard camera. The vehicle moves towards the GPS coordinates of the identified safe landing area. Once the helicopter is in a predetermined distance above the ground (determined by the LIDAR), the helicopter performs a smooth landing on the estimated safe site. Other works in the same area are presented in [184] and [42].

For fixed-wing UAVs, in [19] a bio-inspired landing and take-off strategy was implemented in a small fixed-wing UAV. The strategy is based on measuring the optical flow from optic flow microsensors placed on-board the UAV. The strategy directly uses the optic flow information as feedback to the flight controller. Several automatic take-off and landing tests were successfully conducted. In [130], a vision algorithm for automatic detection of candidate landing sites for forced landing problem is described. The landing area is selected based on the size, shape, slope, and type of surface of the extracted information by image segmentation techniques and by edge extraction algorithms. A back propagation neural network is in charge of classifying this information.

Taking into account previous proposals, in this thesis we attempt to perform the landing task using both and on-board monocular system and a ground (external) multi-camera system. In the case of the on-board system, we use a direct method instead of a feature-based approach, unlike most of the algorithms presented in the literature [161, 141, 132, 39].

In the literature, approaches based on ground cameras for estimating the pose of UAVs

have also been proposed. In [35], a ground camera is used to control a *dirigible* based on the information of artificial landmarks. The recovery of the 3D position and of the orientation is based on the knowledge of the landmarks' geometrical properties, and the control system uses this data to keep the dirigible on a programmed trajectory. In [3], the pose estimation problem of a quadrotor vehicle using simultaneously ground and onboard cameras that see each other is presented. In that work, a blob tracking algorithm is used to determine the position and area of the colored blobs in the image planes. The six degrees of freedom (DOF) signals of the quadrotor were recovered and then sent to the quadrotor to stabilize it. In [151], a stereo vision system is located on the ground facing upwards, and is in charge of detecting the position and orientation of a small quadrotor. Color landmarks located on-board are detected and tracked based on the CAMSHIFT algorithm [25].

A recent work in this area (not published yet) is the one found in [104]. As described in that paper an infrared stereo system fixed on the ground is used to track UAVs during landing tasks (see Figure 2.8). A good feature of this system is that it moves using a pan-tilt unit that permits to augment the FOV of the cameras. Different tests (during the day and during the night) have been conducted using a quadrotor and a fixed-wing UAV, as can be seen in Figure 2.8.

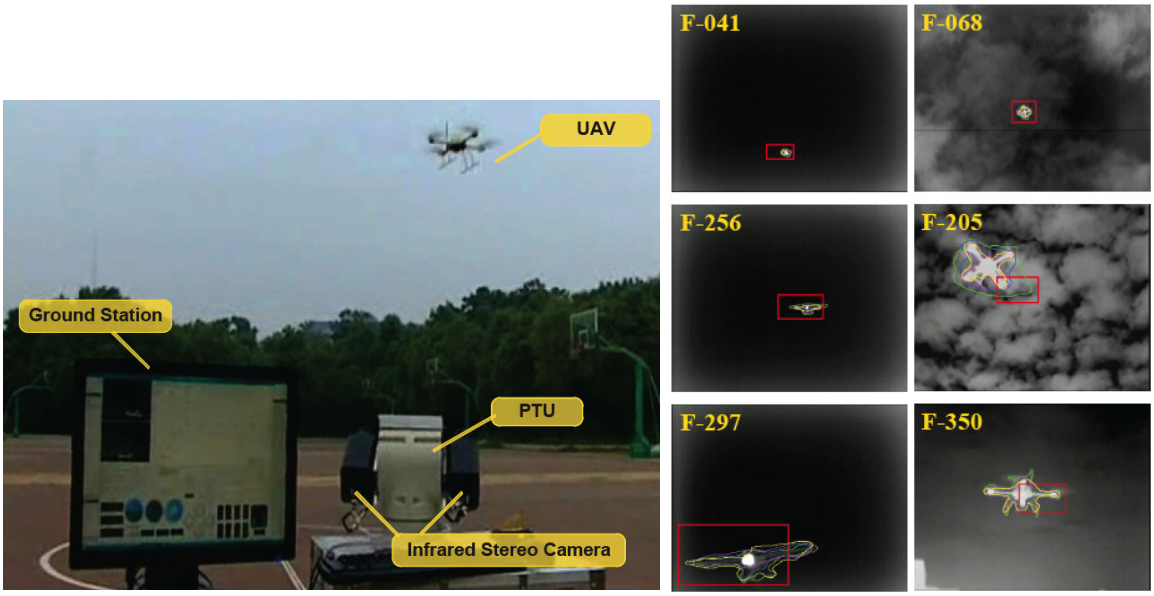


Figure 2.8: Ground-based infrared stereo system. The stereo system moves using a pan-tilt unit that permits to augment the FOV of the cameras (see left image). Different tests (during the day and during the night) have been conducted using a quadrotor and a fixed-wing UAVs (see images on the right). Images from [104].

In this thesis, among other things, we propose the use of a landing platform based on an external multi-camera system (a trinocular system). That system is in charge of recovering the position and heading of UAVs based on the detection and tracking of color key features on-board the UAV. One of the advantages of this system is that the installation of additional on-board sensors for conducting basic tasks

such as landing is not required, and therefore the limited capacity on-board can be well-exploited for other sensors needed for the mission. Additionally, with this system robust 3D information at real-time frame rates is obtained. The proposed system is suitable to complement or replace the GPS-based position estimation in situations where GPS information is unavailable or where its information is inaccurate, allowing the vehicle to perform tasks at low heights, for instance.

Chapter 3

Visual Tracking

3.1. Introduction

The visual tracking task consists in finding the location of an object of interest in each frame of an image sequence. This location can be found by comparing features between frames (feature-based methods), or by comparing the intensity values of each frame with the ones of a reference frame that contains the appearance information of the object we want to track (direct methods). Tracking algorithms are used in this thesis for estimating the position of objects in the image, not only for visual inspection tasks, but also for more complex tasks, such as pose estimation or landing. This is why robust and fast tracking strategies are required.

We focus on general image sequences of 3D scenes in which the objects and the camera may be moving. In this chapter, we concentrate on 2D visual tracking strategies, some of which, are used in other chapters of this thesis for tracking rigid planar objects (or that can be assumed to be planar) in different UAV applications. We do not intend to go into details of a particular algorithm. Instead, this chapter provides a general overview of some state of the art approaches. This chapter reviews two visual tracking techniques, classified as Kernel tracking techniques in [207], that are used in this thesis for tracking rigid objects. The chapter is organized as follows: Section 3.2 reviews a color-based tracking technique; and Section 3.3 formulates the visual tracking task based on image registration and introduces the tracking-by-registration algorithm (as it is called in [50]), which is based on direct methods.

The first technique uses a histogram-based appearance representation of the object to

track, using the color information as the visual feature that will facilitate distinguishing objects. In this thesis, this algorithm is used by the ground-based multi-camera system to detect and track color landmarks on-board the UAV, which are used to estimate the UAV's pose (Chapter 6) for vision-based landing tasks (Section 7.3.2). The second technique that is explained in this chapter uses a rectangular template to represent the object to track and uses the intensity values of the template as visual features to estimate the motion of the object in the image plane. This estimation is based on an image registration technique based on direct methods. In this thesis, this algorithm is used on-board the UAV to track different planar templates for visual inspection and landing tasks (Chapter 7), and also to track the drogue in Autonomous Air-to-Air Refuelling tasks (Section 5.6).

3.2. Color-Based Tracking

The way the object of interest is represented plays an important role in visual tracking. Different features provide different cues for tracking, where their main objective is always to reduce the amount of information present in an image, keeping thus only the relevant information in a particular situation. Features as edges, corners, points, etc, can be used to represent the objects, although they require a high processing time to be extracted from an image.

Nonetheless, the use of color information to represent the object to track, apart from being an intuitive way to represent the information, leads to an efficient algorithm (real-time) for recognizing objects from a variety of viewpoints and resolutions. Although color is affected by illumination conditions caused by weather, time in a day, lights, and shades, it has been used in the literature as an important clue to find an object in a scene in applications as: robotics [163, 121], road sings identification [62, 145], medical images [192, 177], human-computer interaction [203, 197], etc; where its uniqueness has helped not only to identify the object, but also to solve the matching problem through multiple views.

This Section reviews the detection and tracking algorithms that are used by the ground-based multi-camera system to detect and track the UAV.

3.2.1. Detection

Detecting an object of interest in the scene can be done based on the knowledge of the different features of this object, e.g. its color. One of the most difficult problems when using color information outdoors is the chromatic variation of daylight [65]. The apparent color of the object varies as daylight changes (shadows, clouds, and sun, as can be seen in Figure 3.1). Additionally, the presence of objects with similar colors in the scene make the color segmentation a more difficult task. As can be seen in Figure 3.1, the illumination conditions of the left image (sun) can make the blue landmark and the color of the sky be similar. In this figure, it is clearly illustrated how color changes due to variations of the light under which these images were taken.

Different color spaces can be used for segmentation [66], such as the red, green, and blue (RGB) space; the normalized RGB space; the hue, saturation and value (HSV) space;

or the cyan, magenta, and yellow (CMY) space. Additionally, this color information can be modeled in different ways, such as Histograms [180], Gaussian distributions [201], or mixtures of Gaussians [156].

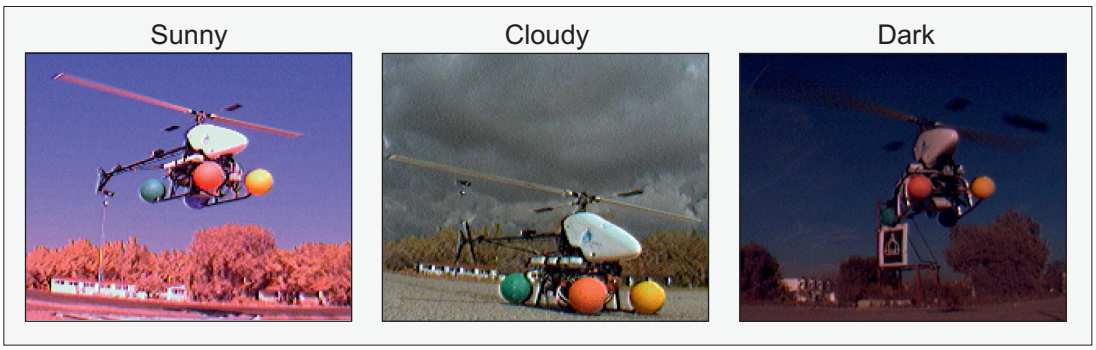


Figure 3.1: Color as object's feature. Illumination changes and the presence of similar colors in the scene make color segmentation when outdoors a difficult task.

In order to segment color information, a color image can be compared with a set of known color cues to find the best match. The RGB images are usually converted to another color space for analysis, in order to separate color from brightness information. In this thesis we use the HSV color space, shown in Figure 3.2.

The HSV color space [174] separates the color information (hue) from saturation (how concentrated the color is) and from brightness. As can be seen in Figure 3.2, hue (H) corresponds to the intuitive notion of color. It is invariant to the variations in light conditions and so it can be exploited for color segmentation under non-uniform illumination, because it is independent from the brightness. Saturation (S) is used to describe how pure a color is or how much white is added to a pure color (intensity of a color), and value (V) refers to the lightness or darkness of a color.

As can be seen in Figure 3.2, varying H corresponds to traversing the color circle. Decreasing S (desaturation) corresponds to increasing whiteness (it measures the departure of a hue from achromatic, i.e. from white or gray), and decreasing V (devaluation) corresponds to increasing blackness (it measures the departure of a hue from black, the color of zero energy). These terms were meant to capture the artistic ideas of hue, tint, shade, and tone [174].

The advantage of this color space is that the color information is managed separately in the channel H (i.e. provides independency between achromatic and chromatic components), making the selection and recognition of a specific color easy. In Figure 3.2, it is possible to observe the independence of the color information and how it is possible to achieve a color segmentation based on the H channel. Even though hue is a useful attribute, there are three problems in using the H channel for color segmentation [189]:

- H is meaningless when V is very low.
- H is unstable when S is very low.
- S is meaningless when V is very low.

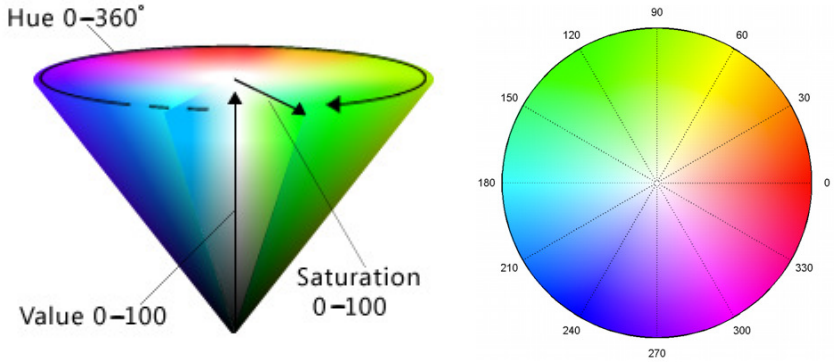


Figure 3.2: HSV color space. The color information is separated in the hue channel (H) from the information of saturation (S) (intensity of a color, how pure it is) and brightness or value (V) (lightness or darkness of a color).

In order to obtain robustness for color segmentation, the unstable areas of this color space should be taken into consideration in order to determine the effective ranges of hue and saturation for color image segmentation.

In this thesis, the objects to track are represented by their color histograms. Histograms are attractive because of their simplicity, speed and robustness to detect the object of interest; they are invariant to translation and rotation; and they change only slowly under changes of viewpoint, changes in scale, and occlusions [180]. Color histograms can be created by sampling the H channel only in the ranges it is stable in. Additionally, a threshold can be used in order to avoid the unstable areas of the HSV color space [189]. Therefore, the objects of interest can be identified by matching the color histogram of the image with model color histograms of the colors we want to segment, as can be seen in Figure 3.3.

The algorithm used to segment the color information is the backprojection algorithm proposed by [180], which is described in Figure 3.3. This method answers the question "Where in the image are the colors that belong to the object being looked for (i.e. the target)". This algorithm computes the likelihood that each image point belongs to a model (the model histogram). This relation is found calculating a ratio histogram **Rh**, as follows:

$$Rh(j) := min\left[\frac{Mh(j)}{Ih(j)}, 1\right] \tag{3.1}$$

Where $Mh(j)$ and $Ih(j)$ are the j^{th} bin of the model histogram (see upper part of Figure 3.3) and the histogram of the current image (left side of Figure 3.3), respectively; and $Rh(j)$ is the j^{th} bin of the ratio histogram created for the color we are looking for. This ratio histogram emphasizes the pixels whose color is highly present in the model and those pixels whose color is rarely present in the image, i.e. pixels that have the same probability distribution of the model are emphasized. On the other hand, locations whose colors are rarely in the model but appear very often in the image histogram are suppressed, see right side of Figure 3.3.

As can be seen in Figure 3.3, this histogram **Rh** is then backprojected onto the image, replacing the values of the image with the values of **Rh** that they index. The resulting image (**B**) of the backprojected algorithm (see Figure 3.3 images on the right) is a grayscale image, where pixels' values represent the probability that the pixel belongs to the color of interest. The backprojected image is created assigning a likelihood to each pixel of the image, as follows:

$$B(x, y) := R(j) \tag{3.2}$$

In Equation 3.2, for every pixel $I(x, y)$ the pixel in the backprojected image **B** will be defined as $B(x, y) = R(j)$, if $I(x, y)$ falls in bin j. The backprojected image can be then convolved by a mask in order to reduce noise.

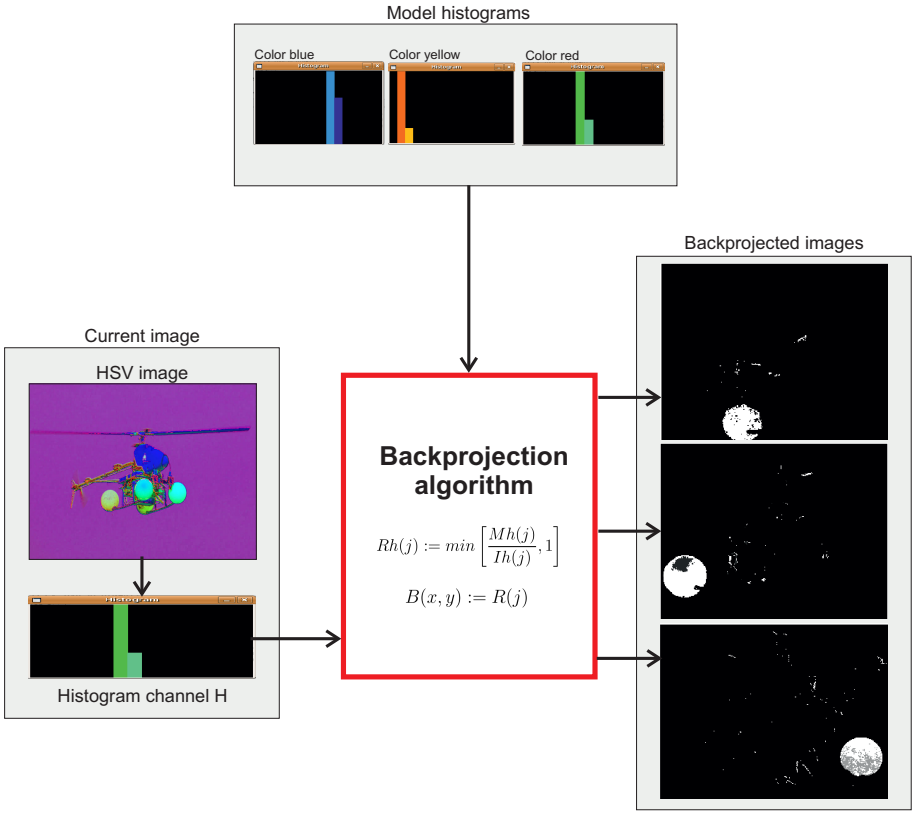


Figure 3.3: Backprojection algorithm. It is used to segment colors of interest. The RGB image is converted to the HSV color space (left). The backprojection algorithm compares model histograms of the colors we are looking for (above) with the histogram of the hue channel of the image that is being analyzed (left). The pixels' values of the backprojected images (right) represent the probability that the pixel belongs to the color of interest.

Figure 3.3 shows an example of the results obtained when the backprojection algorithm is applied using model histograms of different colors. It can be seen that this backprojection algorithm permits the segmentation of colors of interest. The resulting images (the ones on the right) contain the probability of the model colors being present in the image on the left. The model histograms of each color can be seen in the upper position

(blue, yellow, and red). As can be seen, the algorithm emphasizes the pixels whose color is present in the model histogram and suppresses those rarely present.

3.2.2. The CAMSHIFT Tracker

The Continuously Adaptive Mean-Shift (CAMSHIFT) algorithm [25] is based on the Mean-Shift algorithm originally introduced in [67]. In this section, the general idea of these algorithms is introduced. Mean-Shift analysis is a general nonparametric clustering technique based on density estimation for the analysis of complex feature spaces. It does not require prior knowledge of the number of clusters and it does not constraint the shape of the clusters. The algorithm consists in an iterative procedure that shifts each of the feature points to the nearest stationary point along the gradient directions of the estimated density function.

The following steps give an intuitive description of the Mean-Shift algorithm (how to find the densest region) and Figure 3.4 provides a graphical idea of these steps [25, 175]:

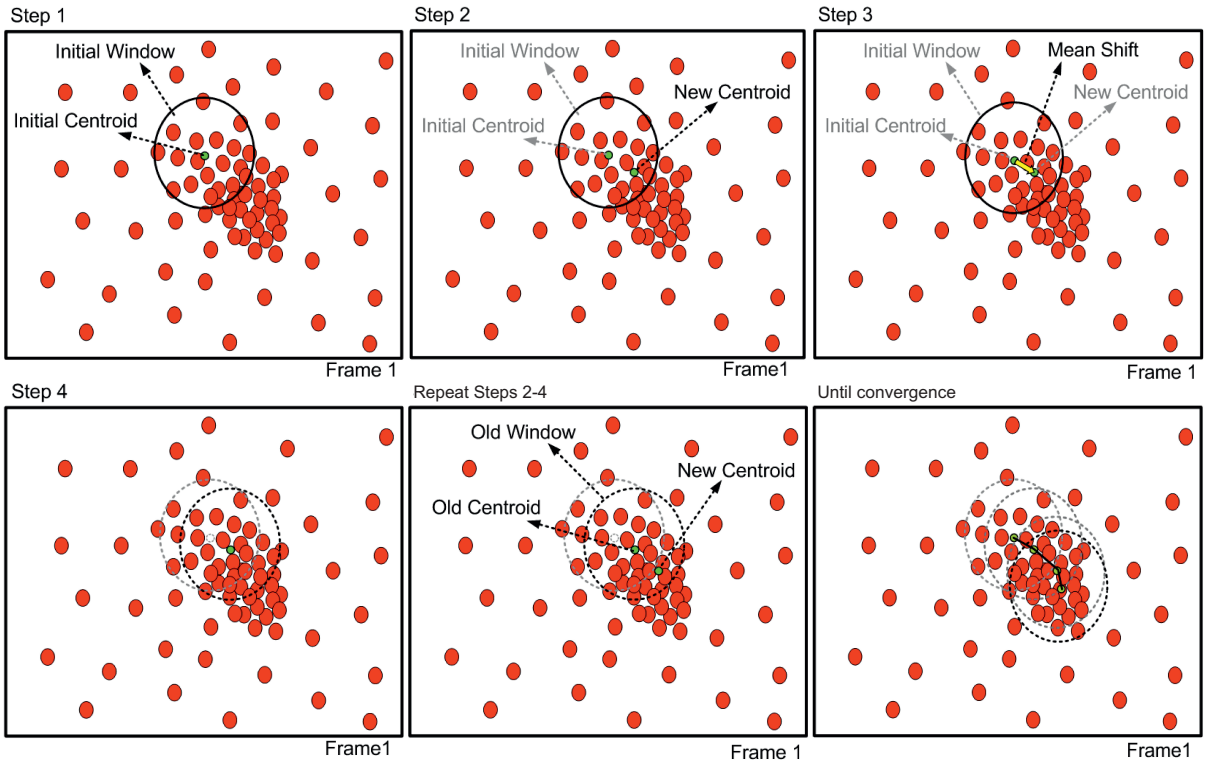


Figure 3.4: Mean-Shift procedure. The densest region is found through an iterative process. **Step 1**, initialize size and position (center) of the search window. **Step 2**, calculate the centroid. **Step 3**, calculate the Mean-Shift vector. **Step 4**, Move the search window to the location of the new centroid. Repeat Steps 2-4 until convergence. Images from [175].

1. Initialize the size and position (center) of the search window (see step 1, Figure 3.4).

2. Calculate the centroid of the search window (see step 2, Figure 3.4).

3. Calculate the distance vector (Mean-Shift vector) between the center of the search window (from step 1) and the calculated centroid (from step 2). See step 3, Figure 3.4.

4. Shift the search window equal to the distance vector (see step 4, Figure 3.4).

5. Repeat from step 2-4 until convergence occurs (or until the centroid location moves less than a preset threshold).

For discrete 2D image probability distributions, once the initial region of interest is selected in the current frame, its centroid (x_c, y_c) is found using the zeroth (m_{00}) and first (m_{10}, m_{01}) order moments [25], as follows:

$$\begin{gathered} m_{00} = \sum_x \sum_y I(x,y) \\ m_{10} = \sum_x \sum_y x\, I(x,y);\ \ m_{01} = \sum_x \sum_y y\, I(x,y) \\ x_{\mathrm{c}} = \frac{m_{10}}{m_{00}};\ \ y_{\mathrm{c}} = \frac{m_{01}}{m_{00}} \end{gathered} \tag{3.3}$$

where $I(x,y)$ is the probability value at position (x,y) in the image.

Then, the region is moved to the position of the calculated centroid. The vector that defines the change in position from an initial location, i.e. the position of the centroid in the previous frame $\mathbf{x}_{\mathbf{c}(F-1)}$, to a new position $\mathbf{x}_{\mathbf{c}(F)}$, is called the Mean-Shift vector ($\mathbf{\Delta x} = (\Delta x, \Delta y)$).

$$\mathbf{x}_{\mathbf{c}(F)} = \mathbf{x}_{\mathbf{c}(F-1)} + \mathbf{\Delta x} \tag{3.4}$$

The window is moved according to this vector and the process is repeated until convergence occurs ($\mathbf{\Delta x} = 0$, or equal to a specific threshold). The final position corresponds to the local density maximum, or local mode of the probability density function [175].

It is important to highlight the robustness of this method. In Figure 3.4, it can be seen that the algorithm is not distracted by points outside the search windows. It is also important to notice that the size of the search window has not been recalculated. That is why the selection of this parameter is crucial for the Mean-Shift process. If the window size is too large, it will contain too many background pixels. On the other hand, if it is too small, the algorithm can"roam" around the object, resulting in a poor object localization. A detailed description of the Mean-Shift algorithm can be found in [175].

On the other hand, the CAMSHIFT algorithm [25] was the first proposed algorithm that used the Mean-Shift algorithm to track moving objects. Since then, the Mean-Shift algorithm has been extended and used for different tracking applications [47, 2, 82]. The previously described Mean-Shift algorithm does not take into account that the distribution can change over time (moving towards or away from the camera, or rotating). That is why the CAMSHIFT algorithm allows the Mean-Shift algorithm to adapt dynamically to those changes by adjusting the search window size based on the zeroth moment information of

the search window [25]. Figure 3.5 shows the flow diagram of the CAMSHIFT algorithm for tracking color objects, and the following steps summarize the algorithm [24]:

1. Choose the initial position and size of the 2D Mean-Shift search window. This position corresponds to the target to be tracked. If that position is not known, the initial position could be the entire image.

2. Calculate the color probability distribution image in a region centered at the search window location but with a larger area than the Mean-Shift search window:

   - Calculate the color histogram.
   - Apply the backprojection algorithm.

3. Run the Mean-Shift algorithm to find the densest region containing the color that we are looking for (one or many iterations). Once it is found, store the zeroth moment and its centroid location.

   - Compute the centroid position in the search window.
   - Center the search window at the centroid position.
   - Repeat until convergence.

4. Set the search window size equal to a function of the zeroth moment found in the previous step.

5. Repeat 3-4 until convergence (the centroid location moves less than a preset threshold).

6. For the next frame, center the search window at the position stored in Step 4 and set the search window size equal to a function of the zeroth moment found in Step 4.

7. Go to step 2.

For each frame of the sequence, the CAMSHIFT algorithm tracks the center of the moving color object in a computationally efficient way, taking into account that the color probability distribution in each frame is not calculated over the whole image but over a region surrounding the current CAMSHIFT window. In this thesis, the CAMSHIFT algorithm is used to track color landmarks on-board the UAV for the ground-based multi-camera system presented in Chapter 6.

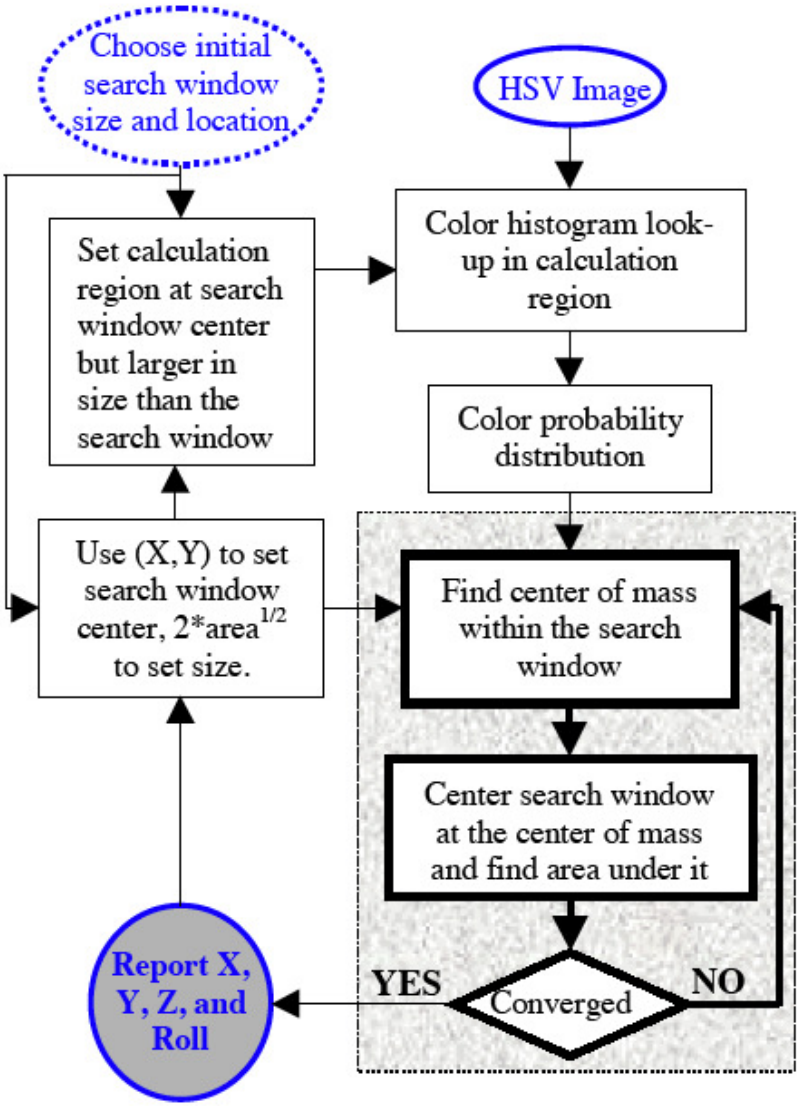


Figure 3.5: Block diagram of the CAMSHIFT algorithm. Image from [25].

3.3. Tracking Based on Image Registration

Image registration or image alignment, as depicted in Figure 3.6, is the process of estimating the transformation that aligns two images of the same scene (i.e., transforming two images into a common coordinate system): a reference image (image A, Figure 3.6) with another image (image B); being these images taken at different times, at different viewpoints, or by different sensors [27]. Image registration has been applied in situations where the integration of useful data obtained from separate images is desired for a better analysis of the situation. This is the case, for instance, of medical image analysis [63], image mosaicing [164], or super-resolution, among others. A good review of image registration algorithms can be found in [216] and [27].

Figure 3.6 shows an example of image registration. Two images A and B are misaligned by a shift due to the change in position of the camera. A reference image is selected (e.g.

image A), and the image registration algorithm is in charge of searching the parameters of the transformation (or motion model) required to match the images, i.e. the motion model that allows to transform image B to the reference frame of image A.

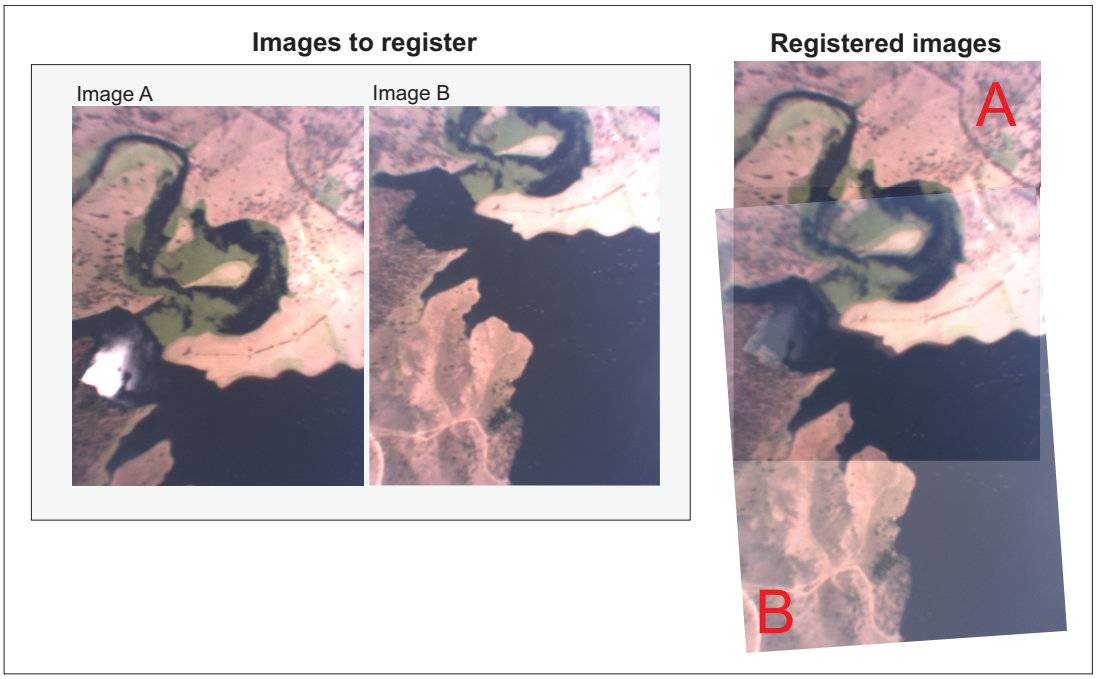


Figure 3.6: Image registration. It consists in recovering the transformation that aligns images A and B, i.e. to transform them into a common coordinate system.

On the other hand, tracking, as shown in Figure 3.7, consists in finding the location of an object of interest in each frame of an image sequence. Tracking can provide not only the position of the object, but also the region in the image that is occupied by the object and/or the state of the object. It can be conducted in 2D (in the image plane) or in 3D space [207].

Figure 3.7 shows an example of a 2D visual tracking task. Changes in the position of an object of interest can be either by the camera movements or by the object's movements. As can be seen in Figure 3.7, once the object of interest has been identified (initialization stage), the tracking algorithm will be in charge of estimating the changes in position of the object, i.e. the amount of motion (the motion model); in order to identify the position of the object in the following frames.

Figures 3.6 and 3.7 show examples of registration and tracking. Although registration and tracking are usually seen as two different strategies used for different applications, they can be used simultaneously in order to achieve a tracking-by-registration strategy [50]; where the object of interest is tracked by iteratively registering pairs of consecutive images. A tracking-by-registration approach requires the definition of the alignment (registration) strategy to be used (i.e. to use either direct or feature-based approaches), and the definition of the appropriate transformation that relates the pixel coordinates in one image with the pixel coordinates in the other one.

The process of image registration can be summarized in four stages [216]: feature

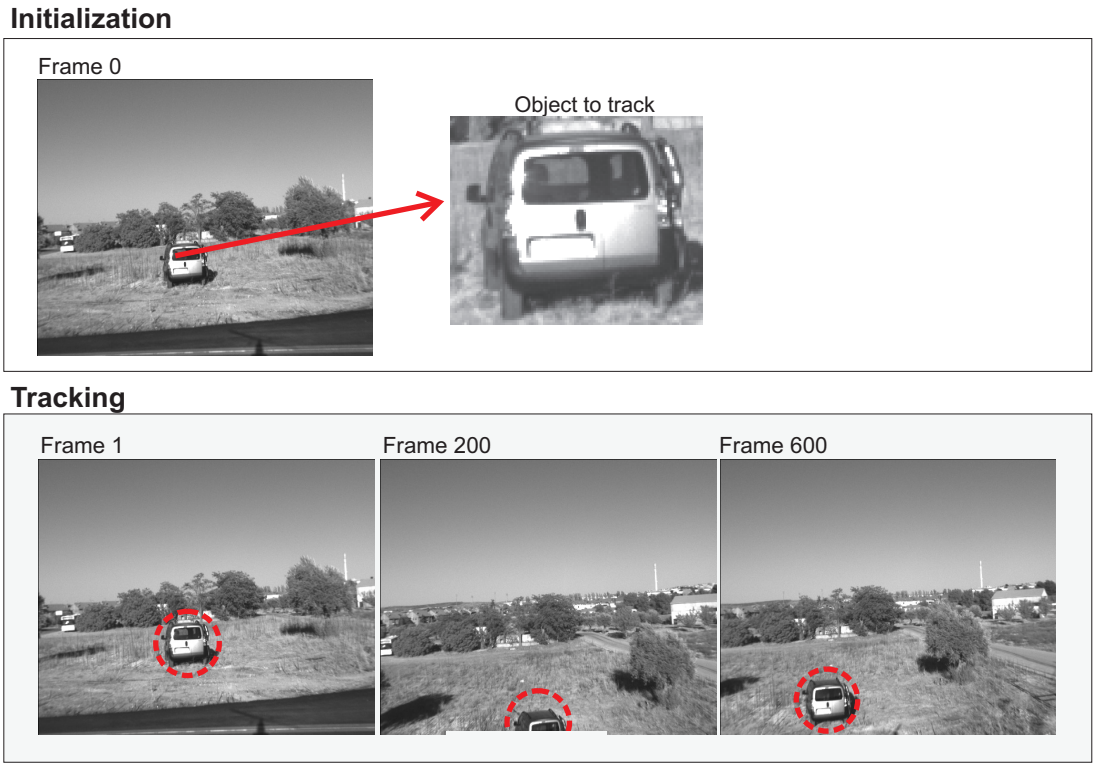


Figure 3.7: 2D Visual tracking. It consists in finding the location of an object of interest in each frame of an image sequence. Tracking can provide not only the position of the object, but also the region in the image that is occupied by the object and/or the state of the object.

detection (i.e. extraction of distinctive features), feature matching (i.e. establishing correspondences between features from two images), motion model estimation (i.e. estimation of the transformation that aligns the images), and image transformation (i.e. applying the found transformation in order to align the images). In the literature, these four stages are carried out in different ways. However, the image registration algorithms can be classified into two categories according to the kind of information used: they can be based on direct methods or based on features.

The approaches based on direct methods estimate the transformation using the intensity values of the images. They merge the feature detection and matching steps. The basic principle is to maximize or minimize the similarity of the images, based on the image brightness, e.g. the sum of square differences (SSD) of image brightness [94, 115]. In this method, either a predefined region of interest (ROI) or the entire image is used.

By contrast feature-based approaches [186] base image registration on the extraction and matching of distinctive features in the images, such as lines or edges [110, 199, 143, 215] or points [80, 212, 114, 16], among others features. These features are expected to be stable over time. Feature-based registration extracts features in both images and then finds the correspondences of the features (e.g. using RANSAC [64, 179]) in order to determine the transformation that aligns the images. The error measure is based on distance. Feature based approaches have the advantage of being more robust against scene movement, being able to align widely separated views. Additionally, feature-based

approaches are considered robust to scale changes, rotations and translations, illumination variations, and are also considered faster since they do not evaluate every pixel in the image but only a few features. Although these characteristics are points in favor, the detection and extraction of distinctive invariant features (robust to different motions) at real-time is a known difficult problem in computer vision.

On the other hand, one of the advantages of direct methods is that the motion and the matching of the pixels are found without intermediate steps, using the intensity information of all the pixels in the image or in the object to track (i.e. they merge the detection and matching steps). Additionally, direct methods make an optimal use of the information that is available in the image since they measure the contribution of every pixel in the image [181]. However, the drawbacks of direct methods are that they rely on some constraints [94] (e.g small frame-to-frame motions), and that their speed is highly dependent on the number of pixels in the template and the registration method used, being it sometimes difficult to achieve real-time frame rates. That is why in most situations feature-based methods are preferable to direct methods.

Therefore, one question that arises at this point is **what kind of method is the most suitable for our application of tracking on-board UAVs: feature-based registration or direct registration?**. This question of which technique is the best one has been asked many times in the computer vision community [186, 94, 216, 150, 181, 50], and in the following lines the main advantage and disadvantages of each technique are summarized:

- **Direct methods**

  Advantages

  – Recover motion with subpixel precision (accuracy).

  – Capable of recovering misalignments of up to $10\% - 15\%$ of the image size. For larger misalignments, an initial estimate is required [94].

  – Work matching sequential images [181].

  – Can be used after a feature-based alignment in order to refine the alignment to improve the results[181].

  – Are more generally applicable to a wide range of scene. They do not distinguish between discontinuous and smoothly varying regions of the image [40].

  – Do not require high-frequency textured surfaces (e.g. corners) to operate, and have optimal performance with smooth gray level transitions [50].

  – Are preferable when the images do not have many prominent details and the distinctive information is provided by gray levels/colors rather than by local shapes and structure.

  – Avoid the difficulties of the feature extraction and matching stages.

  Disadvantages

  – Computational cost depend on the number of pixels evaluated. They are typically more computationally demanding. However, carefully implemented direct methods have proved to be as successful in real-time applications as their

features-based alternatives [40]. To speed up the searching, direct methods often employ pyramidal image representation and optimization algorithms.

- Have a limited range of convergence [181].
- Fail too often for matching partially overlapping images in photo-based panoramas [181].
- Using every pixels means introducing noisy data, i.e. more outliers, which may cause incorrect convergence of the minimization [186].
- Images must have similar intensity functions. They are very sensitive to the intensity changes [216].

- **Feature-based methods**

  Advantages

  - Can be used for widely separated views with images at different scales and orientations [181].
  - Can be advantageous when complex distortions are present [27].
  - Low computational cost: the search space is reduced and irrelevant information is removed [27].
  - Have a wide range of photometric and geometric invariances; they are more robust to partial occlusions; and they are able to cope with severe viewing and photometric distortion [186].
  - Recommended if the images contain enough distinctive and easily detectable objects [216].
  - **Disadvantages**
  - Features must be distributed unevenly over the images [181].
  - A drawback is that the type and scale of features should be chosen depending upon the type of scene in question (e.g. smooth objects, scenes with low contrast, little edge information) [40].
  - Computational cost depends on feature detection, feature matching (depending on the number of features), and robust optimization. The dominant cost is that of determining feature correspondence. For small motions, the search region is small and correspondence fast. In wide-baseline the number of putative matches that must be considered for each feature grows rapidly and does the complexity of geometrically invariant metrics, which must be used to score their similarity [40].
  - Recommended if the images contain enough distinctive and easily detectable objects.
  - Applied if the local structural information is more significant than the information carried by the image intensities.
  - Features might be hard to detect and/or unstable over time [216].

– Feature based methods are sensitive to feature detection and cannot be applied to complex images that do not contain special sets of features to track [120].

As can be seen, the answer to the question is highly dependent on the application and the kind of scene imaged. Nonetheless, direct methods appear as a more general algorithm. On the other hand, recently registration methods have started to use simultaneously both feature-based and direct registration algorithms in order to compensate the drawbacks of both methods [84]. Some suggest that the best option is to combine both methods [150, 181]. So the image registration can be done by first warping using features to find an initial guess and then using direct methods to find a more accurate correspondence.

3.3.1. Direct Image Registration

Image registration consists in aligning two images, by finding the transformation that best aligns them. Using direct methods is one of the options that can be used to align these images. They estimate the parameters of the transformation by shifting or warping the images relative to each other and analyzing how much the intensity values of the pixels agree [181] (i.e. minimizing an error measure based on image brightness).

Direct registration is posed as a non-linear optimization problem taking into account that there is no direct relation between the motion of the pixels and the change in their intensity values. There are many techniques to find the relationship between the measured error (that measures how well the images match, based on image brightness) and the parameters variation. However, this relation is normally found iteratively, by minimizing the sum of squared differences (SSD) between a reference image $\mathbf{T}$, also known as image template, and the current image from an image sequence $\mathbf{I}$. The minimization is based on the brightness constancy constraint of direct methods [94]. Images $\mathbf{I}$ and $\mathbf{T}$ can also correspond to a sub-region of an image. Therefore, using the same notation used in [15], the minimization is written as:

$$\begin{aligned} \mathbf{E}_{\mathrm{SSD}}(\mathbf{p}) &= \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2 \\ \mathbf{p}^* &= \arg\min_{p} \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2 \end{aligned} \tag{3.5}$$

Where $\mathbf{T}$ is the template image, $\mathbf{I}$ the current image, $\mathbf{x} = (x, y)^T$ represents the pixel coordinates, $\mathbf{W}(\mathbf{x};\mathbf{p})$ is the motion model parameterized by the vector of parameters $\mathbf{p} = (p_1, p_2.., p_i)^T$ that describes the transformation, and $E_{\mathrm{SSD}}(\mathbf{p})$ is the residual error. Other robust error metrics can also be used, such as SAD (sum of absolute differences), the wighted SSD function, correlation, or normalized cross correlation mutual information [181, 50].

The most common approach to minimize the Equation (3.5) is to do iterative gradient descent on the SSD, using a Taylor series expansion of the function. This approach has been used in different fields: pose estimation [30], tracking [17], motion segmentation [95], and mosaics [171]; being based, sometimes, on a first-order or on a second-order Taylor approximation of the cost function [14, 26, 115, 17]. The general idea of an iterative gradient descent optimization method applied to minimize Equation (3.5) can be summarized

in three steps, as described in Algorithm 1 and as shown in Figure 3.8. As mentioned previously, minimizing Equation (3.5) is a non-linear problem. However, assuming that a current estimate of the parameters is known ($\mathbf{p}_{\text{init}}$), the algorithm iteratively computes: the error function (step 1 Algorithm 1), the increment of the parameters $\mathbf{\Delta p}$ (step 2 Algorithm 1); and also updates the parameters $\mathbf{p}$ (step 3 Algorithm 1); until some stopping criteria are met, such as when the maximum number of iterations (n) have been reached, the increment of the parameters is below a threshold, etc. Figure 3.8 shows the block diagram of the algorithm.

input : initialize iteration $n = 0$; $\mathbf{p}_{(n)}$=$\mathbf{p}_{(\text{init})}$

repeat

1. **Compute** error function $\mathbf{E}_{\text{SSD}}(\mathbf{p})$
2. **Compute** increment of the parameters $\mathbf{\Delta p}$
3. **Update** the parameters $\mathbf{p}_{(n+1)}$ based on $\mathbf{p}_{(n)}$; and $n = n + 1$

until convergence criteria are reached;

Algorithm 1: Outline of an iterative gradient descent optimization for image registration.

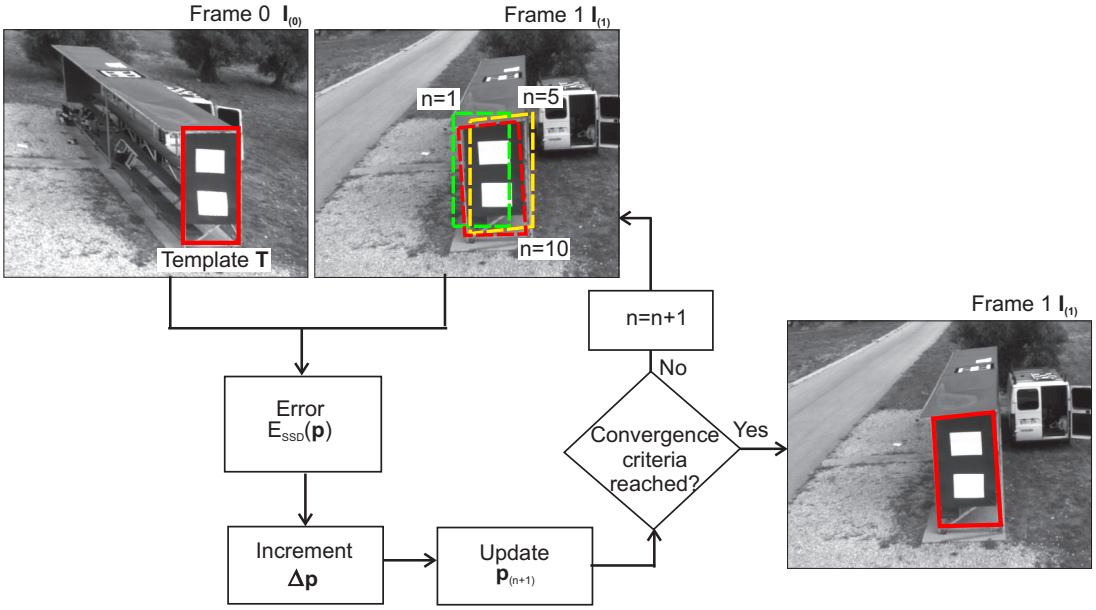


Figure 3.8: Iterative gradient descent optimization for image registration. The parameters are initialized, the error function is calculated between the image and the template, the increment of the parameters is calculated, and the parameters are updated. If convergence conditions are not reached, the process is repeated. If they are reached, the alignment is obtained.

Step 3 of Algorithm 1 can be conducted either additively, by adding the parameters: $\mathbf{p}_{(n+1)} \leftarrow \mathbf{p}_{(n)} + \mathbf{\Delta p}$; or compositionally, by composing the motion model: $\mathbf{W}(\mathbf{x};\mathbf{p})_{(n+1)} \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p})_{(n)} \circ \mathbf{W}(\mathbf{x};\mathbf{\Delta p})$, where n represents the iteration. In [15], gradient descent approaches were classified according to the update rule of the parameters of the motion

model as: forwards additive [115], forwards compositional [171], inverse additive [78], and inverse compositional [14].

3.3.1.1. Lucas-Kanade LK Algorithm

The first work on image alignment based on direct methods is the Lucas-Kanade (LK) algorithm [115]. This is a forwards additive algorithm that has been widely used for object tracking [22] and optical flow estimation [24]. In order to optimize Equation (3.5), the algorithm assumes that a current estimation of $\mathbf{p}$ is known ($\mathbf{p}_{\text{init}}$), and then the algorithm iteratively obtains increments to the parameters $\mathbf{\Delta p}$. The outline of the algorithm is shown in Algorithm 2. Therefore, the LK minimizes:

$$\sum_{\mathbf{x}}[I(\mathbf{W}(\mathbf{x};\mathbf{p}+\mathbf{\Delta p}))-T(\mathbf{x})]^2 \tag{3.6}$$

Where $\mathbf{T}$ is the template image, $\mathbf{I}$ the current image, $\mathbf{x}=(x,y)^T$ represents the pixel coordinates, and $\mathbf{W}(\mathbf{x};\mathbf{p})$ is the motion model, where $\mathbf{p}=(p_1,p_2..,p_i)^T$ is the vector of parameters that describes the transformation. After linearizing Equation (3.6) using a first-order Taylor expansion, the increment of the parameters, i.e. the minimum of Equation (3.6), is found as follows (Step 2 of Algorithm 2):

$$\mathbf{\Delta p}=\mathbf{H}^{-1}\sum_{\mathbf{x}}\left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}[T(\mathbf{x})-I(\mathbf{W}(\mathbf{x};\mathbf{p}))] \tag{3.7}$$

Where $\mathbf{H}$ is the approximation of the Hessian matrix, defined as:

$$\mathbf{H}=\sum_{\mathbf{x}}\left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}\left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right] \tag{3.8}$$

In Equations (3.7) and (3.8), $\nabla\mathbf{I}=\left(\frac{\partial\mathbf{I}}{\partial x},\frac{\partial\mathbf{I}}{\partial y}\right)$ is the gradient of $\mathbf{I}$ that is evaluated at $\mathbf{W}(\mathbf{x},\mathbf{p})$, and $\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$ is the Jacobian of the transformation (i.e. of the motion model).

Then, the parameters are updated (additive update) in Step 3 of Algorithm 2, as follows:

$$\mathbf{p}_{(n+1)}\leftarrow\mathbf{p}_{(n)}+\mathbf{\Delta p} \tag{3.9}$$

Steps 1-3 in Algorithm 2, i.e. Equations (3.7) and (3.9), are iteratively applied until stopping criteria are reached, denoting the best local alignment solution.

As can be seen in Equation (3.7), the gradient $\nabla\mathbf{I}$ and the Jacobian $\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$ depend on the parameters $\mathbf{p}$, and $\mathbf{p}$ varies in every iteration. Therefore, all the components of Equation (3.7) must be computed in every iteration, which depending on the situation, can be computationally expensive.

A detailed derivation of the LK algorithm can be found in [15].

input : initialize iteration $n = 0$; $\mathbf{p}_{(n)}$=$\mathbf{p}_{(\text{init})}$

repeat

1. **Compute** error image $\mathbf{T}(\mathbf{x}) - \mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))$
2. **Compute** increment of the parameters, i.e. calculate $\mathbf{H}$, $\mathbf{H}^{-1}$, $\nabla\mathbf{I}$, $\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$

$$\mathbf{\Delta p} = \mathbf{H}^{-1}\sum_{\mathbf{x}}\left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

3. **Update** the parameters $\mathbf{p}_{(n+1)} \leftarrow \mathbf{p}_{(n)} + \mathbf{\Delta p}$; and $n = n + 1$

until convergence criteria are reached;

Algorithm 2: Outline of the iterative Lucas-Kanade algorithm.

3.3.1.2. Inverse Compositional Image Alignment ICIA

Now we focus on the Inverse Compositional Image Alignment (ICIA) algorithm proposed by [14]. It is considered an efficient algorithm for image registration (or image alignment) that permits an efficient estimation of the parameters that define the motion of the objects in the scene.

The key points of the ICIA algorithm, when compared with the Lucas and Kanade (LK) approach, are the changes of roles of images $\mathbf{I}$ and $\mathbf{T}$ of Equation (3.6), and the way the motion model is updated in Equation (3.9). In the LK approach, the parameters of the Hessian matrix and its inversion must be carried out in every iteration. This implies a high computational cost. However, in the ICIA algorithm, the change of roles between the images $\mathbf{I}$ and $\mathbf{T}$ makes the Hessian matrix be constant. It is calculated at the beginning of the tracking task, thus making the ICIA algorithm a more efficient approach than the LK. The goal of the ICIA consists in finding the vector of parameters $\mathbf{p}$ of a motion model. Algorithm 3 summarizes the different steps carried out to find these parameters. The ICIA algorithm minimizes:

$$\sum_{\mathbf{x}}[T(\mathbf{W}(\mathbf{x};\mathbf{\Delta p})) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]^2 \tag{3.10}$$

Comparing Equations (3.6) and (3.10), it can be seen that the roles of $\mathbf{I}$ and $\mathbf{T}$ have now changed. After linearizing Equation (3.10), performing a first-order Taylor expansion, the increment of the parameters is found, as follows:

$$\mathbf{\Delta p} = \mathbf{H^{-1}}\sum_{\mathbf{x}}\left[\nabla\mathbf{T}\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})] \tag{3.11}$$

Where $\mathbf{H}$ is the approximation of the Hessian matrix.

$$\mathbf{H} = \sum_{\mathbf{x}}\left[\nabla\mathbf{T}\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}\left[\nabla\mathbf{T}\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right] = \sum_{\mathbf{x}}\left[\mathbf{SDI}(\mathbf{x})^{\mathrm{T}}\,\mathbf{SDI}(\mathbf{x})\right] \tag{3.12}$$

Where $\mathbf{SDI}(\mathbf{x}) = \nabla\mathbf{T}(\mathbf{x})\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$ is the steepest descent image [15].

The difference between Equation (3.12) and Equation (3.8) is that now the gradient of the template $\nabla\mathbf{T} = \left(\frac{\partial\mathbf{T}}{\partial x}, \frac{\partial\mathbf{T}}{\partial y}\right)$ and the Jacobian of the transformation $\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$ are evaluated

when the template is defined, i.e. at $\mathbf{W}(\mathbf{x}, \mathbf{0})$.

As can be seen in Equations (3.11) and (3.12), thanks to the change of roles of the images $\mathbf{T}$ and $\mathbf{I}$ in Equation (3.10), the Hessian matrix does not depend on the current parameters $\mathbf{p}$. As a consequence of this, as can be seen in Algorithm 3, most of the terms in Equation (3.11) can be pre-computed: the gradient, the Jacobian, and the Hessian with its inverse; and therefore a faster alignment can be achieved.

The motion model is updated as follows (inverse compositional update):

$$\mathbf{W}(\mathbf{x};\mathbf{p})_{(n+1)} \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p})_{(n)} \circ \mathbf{W}(\mathbf{x};\mathbf{\Delta p})^{-1} \tag{3.13}$$

The steps of the ICIA algorithm are summarized in Algorithm 3. Steps 5-7 in Algorithm 3 are repeated until stopping criteria are reached denoting the best alignment solution.

input : initialize iteration $n = 0$; $\mathbf{p}_{(n)}$=$\mathbf{p}_{(\text{init})}$

Pre-compute

1. **Evaluate** the gradient of the template image: $\nabla\mathbf{T}$
2. **Evaluate** the Jacobian: $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
3. **Compute** the steepest descent image: $\mathbf{SDI}(\mathbf{x}) = \nabla\mathbf{T}(\mathbf{x})\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
4. **Compute** the Hessian matrix and its inverse: $\mathbf{H} = \sum_{\mathbf{x}} \left[\mathbf{SDI}(\mathbf{x})^{\mathrm{T}}\,\mathbf{SDI}(\mathbf{x})\right]$, and $\mathbf{H}^{-1}$

repeat

5. **Compute** error image $\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \mathbf{T}(\mathbf{x})$
6. **Compute** increment of the parameters using the pre-computed values: $\mathbf{\Delta p} = \mathbf{H^{-1}} \sum_{\mathbf{x}} \left[\nabla\mathbf{T}\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]$
7. **Update** the parameters $\mathbf{W}(\mathbf{x};\mathbf{p})_{(n)} \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p})_{(n)} \circ \mathbf{W}(\mathbf{x};\mathbf{\Delta p})^{-1}$; and $n = n + 1$

until convergence criteria are reached;

Algorithm 3: Outline of the Inverse Compositional Image Alignment algorithm.

3.3.2. 2D Tracking-by-Registration Based on Direct Methods

The 2D visual tracking task consists in determining the position of an object in the image plane, assuming that the 3D displacements of the object can be modeled by a 2D transformation (translation, affine, homography [81]). In the literature, the vast majority of approaches make use of feature-based methods to track objects [208, 141, 132]. Feature-based approaches have the advantage of geometric and photometric invariance [186]. However, the selection of appropriate features, and the matching process among them, make the performance of feature-based methods, in some cases, slow and sometimes difficult to conduct, depending on the characteristics of the scene.

The methods described in Section 3.3.1 for direct image registration can be used for tracking. In this case, the tracking task is formulated as an incremental image registration task where the intensity values of the pixels are used to estimate the motion of the object to track. We refer to this kind of tracking approaches as in [50]: tracking-by-registration

approaches based on direct methods (see Figure 3.9). The inputs of registration are the two images to be registered: template image $\mathbf{T}$, and the current image $\mathbf{I}$. These images must overlap; and the output is a geometrical transformation, which transforms points in one image to points in the other one.

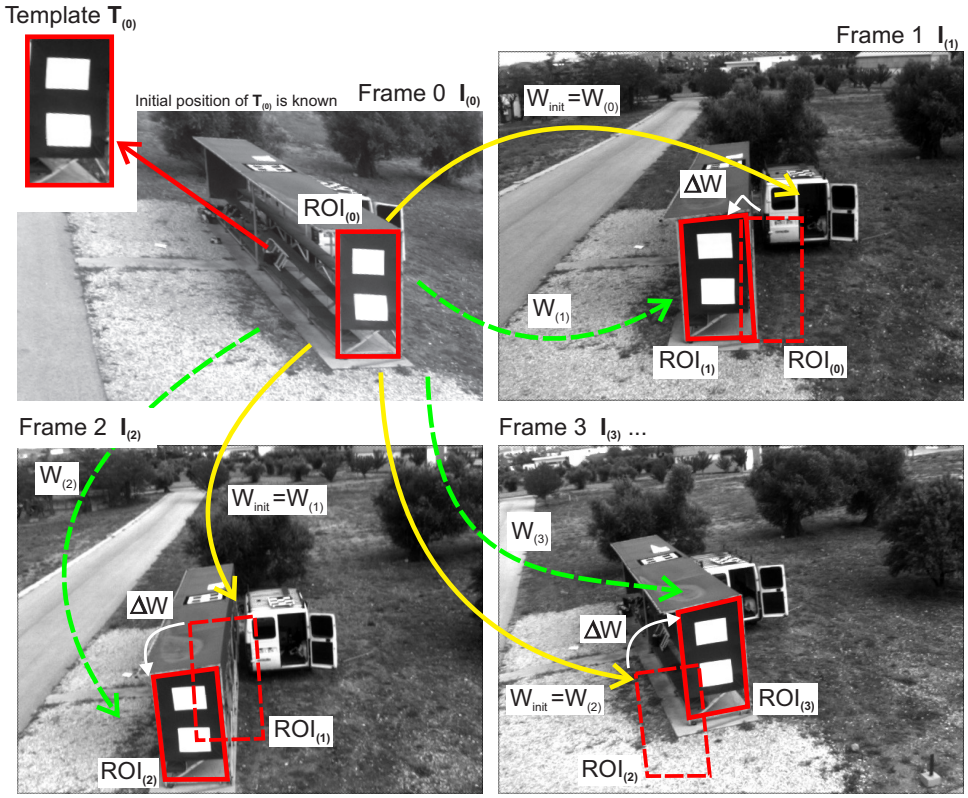


Figure 3.9: 2D tracking-by-registration strategy based on direct methods. The tracking task is formulated as an incremental image registration task. The object to track is defined in the first frame $\mathbf{T}_{(0)}$. When a new frame is analyzed, e.g. $\mathbf{I}_{(1)}$ (upper right image), the motion $\mathbf{W}_{(1)}$ between the reference and the current images (green arrow) is found by an image registration technique, assuming that an initial estimation of the motion $\mathbf{W}_{\text{init}}$ is known (yellow arrow).

In the case of tracking-by-registration, the inputs are also two images, with the slight difference that in this case the reference image $\mathbf{T}$ is an image or a sub-region of an image that contains the object we want to track, and the current image corresponds to an image of a sequence. Therefore, the registration is conducted between the template image (e.g. the first image) and the current image of the sequence, using as initial estimation of the motion model the one estimated in the previous frame (i.e. the one that permitted to locate the object in the previous frame). This initial estimation is valid, assuming that the frame-to-frame motion is small, and therefore the images will be close enough to be registered. The output, in the case of tracking, for every analyzed frame, is also a transformation that relates the reference image and the image of the sequence; but in this case, that transformation is used to identify the position of the template image in each frame of the sequence.

The following paragraphs explain the steps of a tracking-by-registration task based on

direct methods, which are summarized in Algorithm 4 and are graphically represented in Figure 3.9.

The goal of a tracking-by-registration based on direct methods is to find the 2D position of the object to track in the image plane in each frame of the image sequence, using as template a region that represents the object to track, and using the intensity values of the template as visual features to estimate the motion of the object in the image plane. Therefore, assuming that an initial position of the object is known, that the motion between frames is small, that the pixels that belong to the object move similarly, and that the appearance of the object does not change over time (the direct methods' constraints [94]), the tracking task can be formulated as an incremental image registration task, as shown in Figure 3.9.

As shown in Figure 3.9, a reference image (the object to track) is defined in the first frame (template $\mathbf{T}_{(0)}$, Figure 3.9, Frame 0, upper left image). This reference image corresponds to a sub-image or $\text{ROI}_{(0)}$ (Region of Interest), called image template, defined in the first frame $\mathbf{I}_{(0)}$ (the subscript represents the number of the frame), and is found either manually or automatically by detection algorithms.

When a new frame is analyzed, e.g. $\mathbf{I}_{(1)}$ (Figure 3.9, Frame 1, upper right image), the motion $\mathbf{W}_{(1)}$ between the reference and the current images (Figure 3.9, Frame 1, green arrow) is found by an image registration technique, assuming that an initial estimation of the motion $\mathbf{W}_{\text{init}}$ is known (Figure 3.9, Frame 1, yellow arrow). When an initial estimation is not known (because this is the first frame analyzed), this initial estimation can be assumed as the identity matrix, assuming that the frame-to-frame motion is small (i.e. the ROI of the object to track in the previous frame and the ROI of the object to track are overlapped).

Therefore, the image registration algorithm will be in charge of estimating the incremental motion model $\mathbf{\Delta W}$ in every iteration, as explained in Section 3.3.1. Thus, the motion $\mathbf{W}_{(1)}$ is estimated, and as a consequence of this, the $\text{ROI}_{(1)}$ is found, i.e. the position of the object to track in the current frame is found. Then, the estimated motion $\mathbf{W}_{(1)}$ (Figure 3.9, Frame 1, green arrow) is propagated to the next frame, as an initial estimation of the motion $\mathbf{W}_{\text{init}} = \mathbf{W}_{(1)}$ (yellow arrow, Fig. 3.9, Frame 2, bottom left image). This process is repeated with each image of the sequence, as described in Algorithm 4.

input : Initial frame $\mathbf{I}_{(0)}$

Initialization stage

1. **Detect** $\mathbf{T}_{(0)}$ (or $\text{ROI}_{(0)}$) in $\mathbf{I}_{(0)}$
2. **Initialize** $\mathbf{W}_{\text{init}} = \mathbf{W}_{(0)}$, where $\mathbf{W}_{(0)}$ can be either a known transformation or the identity matrix

Tracking stage

foreach *image* $\mathbf{I}_{(F)}$ *in the sequence* **do**

1. **Estimate** $\mathbf{W}_{(F)}$ applying an image registration algorithm, using $\mathbf{W}_{\text{init}}$ as initial guess
2. **Find** $\text{ROI}_{(F)}$ using $\mathbf{W}_{(F)}$, i.e. find the position of the object to track $\mathbf{T}_{(F)}$ in the current image
3. **Propagate** the motion model $\mathbf{W}_{(F)}$ as initial guess to the next frame: $\mathbf{W}_{\text{init}} = \mathbf{W}_{(F)}$

end

Algorithm 4: Outline of a tracking-by-registration algorithm.

3.3.2.1. Motion Models W

In the algorithms based on image registration described in this Chapter, we have referred to $\mathbf{W}$ as a transformation or motion model that is defined by a vector of parameters. Depending on the parameters, $\mathbf{W}$ can represent different transformations which are reviewed in this section.

In the case of object tracking, the transformation or motion model $\mathbf{W}$ represents the trajectory of the object in the image plane while it moves around the scene, whereas in the case of registration, this motion model represents the mathematical relationship between pixels' coordinates of the reference image and the current image. In this section, we focus on 2D motion models that use 2D parametric transformations to define the displacement of the pixels. However, other transformations can be adopted: 3D transformations, cylindrical and spherical coordinate, or lens distortion, [181, 94].

The motion model $\mathbf{W}$ is represented as a 3×3 matrix (3.14) parameterized by the vector of parameters $\mathbf{p} = (p_1, ...p_i)^{\mathrm{T}}$ in such a way that $\mathbf{W}$ is the identity matrix when the parameters are equal to zero.

$$\mathbf{x}' = \mathbf{W}\,\mathbf{x} = \mathbf{W}(\mathbf{x}; \mathbf{p})$$

$$\mathbf{W} = \begin{bmatrix} 1+p_1 & p_2 & p_3 \\ p_4 & 1+p_5 & p_6 \\ p_7 & p_8 & 1 \end{bmatrix} \tag{3.14}$$

Therefore, $\mathbf{W}$ is the motion model that will transform the 2D pixel coordinates $\mathbf{x}$ (where $\mathbf{x} = (x, y, 1)^{\mathrm{T}}$) of the reference image $\mathbf{T}$, into the 2D coordinates $\mathbf{x}' = (kx', ky', k)^{\mathrm{T}}$ in the current image $\mathbf{I}$. If $\mathbf{W}$ represents the homography, then $k = xp_7 + yp_8 + 1$. Otherwise $k = 1$.

$\mathbf{W}$ models different 2D transformations, as follows:

- **Translation**: has two degrees of freedom (DOF) p_3 and p_6. They represent the position in the x and y axes in the image plane, and p_1, p_2, p_4, p_5, p_7 and p_8 are equal to zero.
- **Rotation+Translation**: is also known as 2D rigid body transformation, and has three DOF: the 2D position in the image plane, i.e. the translation (p_3, p_6); and rotation (p_2=-p_4, assuming small rotations). p_1, p_5, p_7 and p_8 are equal to zero. The invariants of this motion model are the Euclidean distance, the angle between two lines, and the area.
- **Translation+Scale**: has three DOF: the 2D position in the image plane, i.e. the translation (p_3, p_6); and scale (p_1=p_5). p_2, p_4, p_7 and p_8 are equal to zero.
- **Rotation+Translation+Scale**: also known as similarity transformation. This transformation has four DOF: the 2D position in the image plane, i.e. the translation (p_3, p_6); rotation (p_2=-p_4, assuming small rotations); and scale (p_1=p_5). p_7 and p_8 are equal to zero. This transformation preserves shape, and its invariants are: ratio of length and angle between lines.
- **Affine**: this model has 6 DOF: p_1...p_6 (p_7 and p_8 are equal to zero). This transformation preserves parallelism, ratio of length, and ratio of areas.
- **Homography**: also known as perspective transformation. It has eight DOF (defined up to scale value), and will be parameterized by 8 parameters as shown in Equation (3.14), although other parameterizations can be used, as presented in [13, 18].

The previously mentioned transformations belong to the hierarchy of the 2D parametric transformations [81], where each transformation is a specialization of a more complex one. The goal of the image registration technique consists, therefore, in finding the vector of parameters $\mathbf{p}$.

3.4. Summary

In this chapter, tracking strategies used in the development of this thesis were introduced. The concepts of the color-based tracking strategy were presented. This strategy is the one used in our ground-based multi-camera system. Additionally, in this chapter, the concepts of image registration have been reviewed, and two well known strategies for registration: the LK algorithm and the efficient ICIA algorithm have been described. Finally, the tracking-by-registration strategy based on direct methods was introduced. This strategy and the ICIA algorithm are used in this thesis for tracking tasks on-board UAVs.

Chapter 4

Hierarchical Multi-Parametric and Multi-Resolution Tracking Strategy (HMPMR)

4.1. Introduction

This chapter[1], presents a hierarchical tracking-by-registration strategy based on direct methods, which is proposed for performing tracking tasks with cameras on-board aerial vehicles. This strategy is proposed to overcome problems posed by challenging conditions of the task, such as constant vibrations, fast 3D changes, partial occlusions, real-time requirements, and the limited processing capacity on-board UAVs.

The proposed strategy is a hierarchical tracking strategy in terms of image resolution and number of parameters estimated in each resolution, that we have called: the Hierarchical Multi-Parametric and Multi-Resolution strategy (HMPMR). In this strategy, different numbers of parameters (i.e. different motion models) are integrated in the estimation of the motion. We propose to use the ICIA [14] as an image registration algorithm to deal with the efficiency problem of direct methods (real-time operation); and we also propose to extend it with the HMPMR strategy, to deal with the small motion constraint

[1]Publications of the author related to this chapter:
- Towards Autonomous Air-To-Air Refuelling for UAVs Using Visual Information [125]
- A Hierarchical Tracking Strategy for Vision-Based Applications On-Board UAVs [123]

of these kinds of methods.

In the literature, as was shown in Section 2.1, the vast majority of approaches make use of feature-based methods to track objects. In this chapter it is shown that although some of these feature-based solutions are faster, direct methods can be more robust under fast 3D motions (fast changes in position), constant vibrations (without requiring any specific hardware or software for video stabilization), partial occlusions, and situations where part of the object to track is out the field of view of the camera. That robustness can be achieved without compromising the real-time operation of the algorithm.

In this chapter, the proposed tracking strategy is introduced and its performance is analyzed comparing it with state of the art algorithms based both on direct methods and feature-based methods; using simulated data, aerial images, and publicly available datasets of planar templates; and also using different evaluation mechanisms that can be used to analyze the performance of the HMPMR-ICIA algorithm, such as those related to manually generated ground truth data. Additionally, in this chapter, we present results obtained with the application of the proposed tracking strategy in two different scenarios: tracking objects on-board UAVs, and tracking the drogue in Autonomous Air-to-Air Refuelling tasks.

4.2. Strategy Overview

As was shown in the previous chapter, the goal of image registration (or image alignment) techniques consists in finding the vector of parameters of the motion model that aligns the images. As mentioned in Section 3.3.2, an incremental image registration algorithm can be used to track objects. If the iterative ICIA algorithm is used, an efficient tracking can be achieved using direct methods.

This iterative algorithm (the ICIA) relies on the assumption that a previous estimation of the parameters of the motion model is known, and that after a linearization of the cost function (Equation (3.10)) the algorithm iteratively estimates the increments of the parameters of the motion model until stopping criteria are reached. Nevertheless, this linearization is valid only when the range of motion between frames is small. Concerning tracking with cameras on-board aerial vehicles, as well as regarding other applications, this constraint can not always be ensured (because of fast 3D motions and limited capacity on-board, etc).

Multi-resolution (MR) approaches were proposed to help dealing with the small frame-to-frame motion constraint of direct methods [18]. Images are downsampled creating an MR pyramid, and in each resolution level the same motion model is estimated. However, in practice, as pointed out in [181], "it is hard to use more than two or three levels of a pyramid before important details start to be blurred away". The MR strategy suggests that at low resolutions the vector of motion is smaller and long displacements can be better approximated, as was pointed out in [18]. The higher the frame-to-frame motion is, the bigger the number of levels the MR strategy requires to be able to cope with the large displacement. Nevertheless, there must be a compromise between the number of levels required to overcome the large inter-frame motion and the amount of information required to estimate this motion.

In Figure 4.1, it can be seen that the maximum motion a MR strategy can estimate (i.e. the maximum number of levels the MR structure should have) is closely related to the image information in each level. In the example shown in Figure 4.1, a MR pyramid with 5 levels is used to estimate the homography motion model (8 parameters). An intuitive analysis of Figure 4.1 suggests that the information at low resolutions, e.g. at levels 4 and 5, may not be good enough in terms of quality and quantity for estimating the 8 parameters of the homography (in a MR strategy, the same motion model is estimated in each level of the pyramid). As was said in [182]: sometimes "the homography contains more free parameters than necessary, in those cases, the algorithm suffers from slow convergence and sometimes gets stuck in local minima".

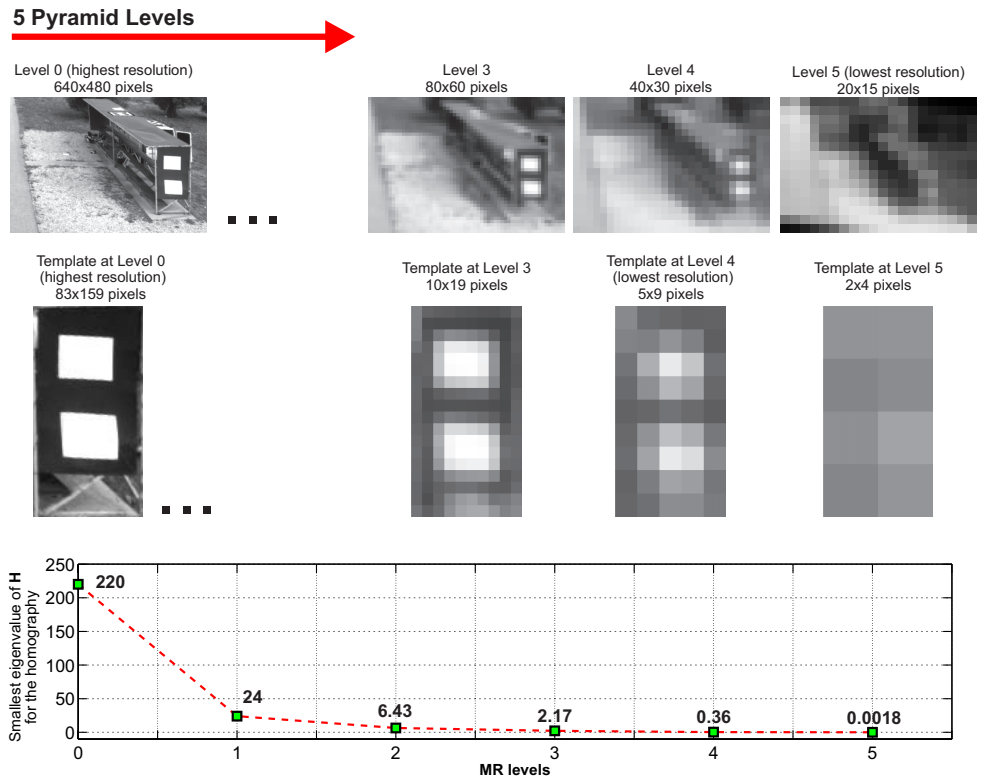


Figure 4.1: General idea of an MR strategy. At low resolutions, the information may not be good enough to estimate motion models with a large number of parameters (e.g. a homography, 8 parameters). In the plot, it can be seen that there is a relation between the image information and the number of parameters that can be estimated with that information, which can be analyzed with the smallest eigenvalue criterion (**H** is close to be singular if its smallest eigenvalue is close to 0). See levels 4 and 5.

In [170], a threshold in the smallest eigenvalue as a measurement of feature quality was proposed. In this thesis, we propose to use the smallest eigenvalue of the Hessian (**H**) as a confidence measurement of the tracking algorithm for estimating specific numbers of parameters with the information available in a given resolution. In the plot shown in the lower part of Figure 4.1, it can be seen that if the MR-ICIA algorithm is used, the smallest eigenvalues of **H** in the fourth and the fifth levels are close to zero, i.e. the Hessian matrix **H** is ill-conditioned. In those levels, the quality of the available visual information is

not going to be enough to correctly estimate the 8 parameters of the homography (the tracking algorithm requires a well-conditioned **H**).

Therefore, from Figure 4.1 it can be inferred that when only a MR strategy is used, the low resolution information could be not sufficient to find a robust estimation of a motion model with a large number of parameters. Nonetheless, if less pyramid levels are considered in order to avoid the loss of information produced by the low resolution, the reduction of levels in the pyramid will cause a reduction in the range of motion the algorithm can tolerate. For this reason, for many applications MR approaches are sometimes not good enough to solve the tracking problem in the presence of large frame-to-frame motions, presenting an unstable behavior when estimating complex motion models with a large number of parameters (e.g. a homography), especially when they are estimated in the low resolution levels.

The **aforementioned problem motivated the idea of including the multi-parametric** (MP) estimation of the motion model inside the MR pyramid. We have called this strategy the Multi-Parametric and Multi-Resolution strategy: the HMPMR. The HMPMR approach allows to continue taking advantage of low resolution information to find large ranges of motion, even when motion models with a large number of parameters are required to be estimated in the highest level of resolution. Therefore, **the main objective of the HMPMR structure is to cope with the deficiencies of the MR structure**. It is focused on robustly estimating complex motion models under large frame-to-frame motions.

The proposed strategy for tracking using cameras on-board UAVs can be seen in Figure 4.2. It makes use of two hierarchical structures: the multi-resolution (MR) structure, created by downsampling the images (Gaussian [6] or Laplacian [32]); and the multi-parametric (MP) one, where for each MR level, as shown in Figure 4.2, a specific motion model is recovered (different motion models with different numbers of parameters are estimated in each level). The general idea behind this strategy, as shown in Figure 4.2, is that by estimating only a small number of parameters at the lowest resolution levels and smoothly increasing the complexity of the motion model (i.e. the number of estimated parameters increases) through the MR pyramid (i.e. with the resolution of the image), it is possible to obtain a robust estimation of the motion model by increasing the range of motion the algorithm can tolerate. By doing this, we ensure to have a well-conditioned Hessian in the lowest resolution levels. The values of the smallest eigenvalue of **H** increase, in comparison with the ones obtained when the same motion model is estimated in each level (i.e. when the MR approach is used).

The previous considerations can be confirmed using the example shown in Figure 4.1 and the smallest eigenvalue criterion (i.e. **H** is close to be singular if its smallest eigenvalue is close to zero). In Figure 4.3, for different HMPMR-ICIA configurations, the values of the smallest eigenvalue of **H** in each level, are plotted (in each level, the upper number represents the number of parameters estimated in that resolution, and the lower number corresponds to the value of the smallest eigenvalue). In the figure, it can be seen that when a few parameters (e.g. 2 parameters) were estimated in the low resolution levels (e.g. levels 4 and 5), the values of the smallest eigenvalues of **H** increased in those levels compared to the values obtained when the 8 parameters were estimated. Therefore, it can be seen that there is indeed a relation between the image information and the number of

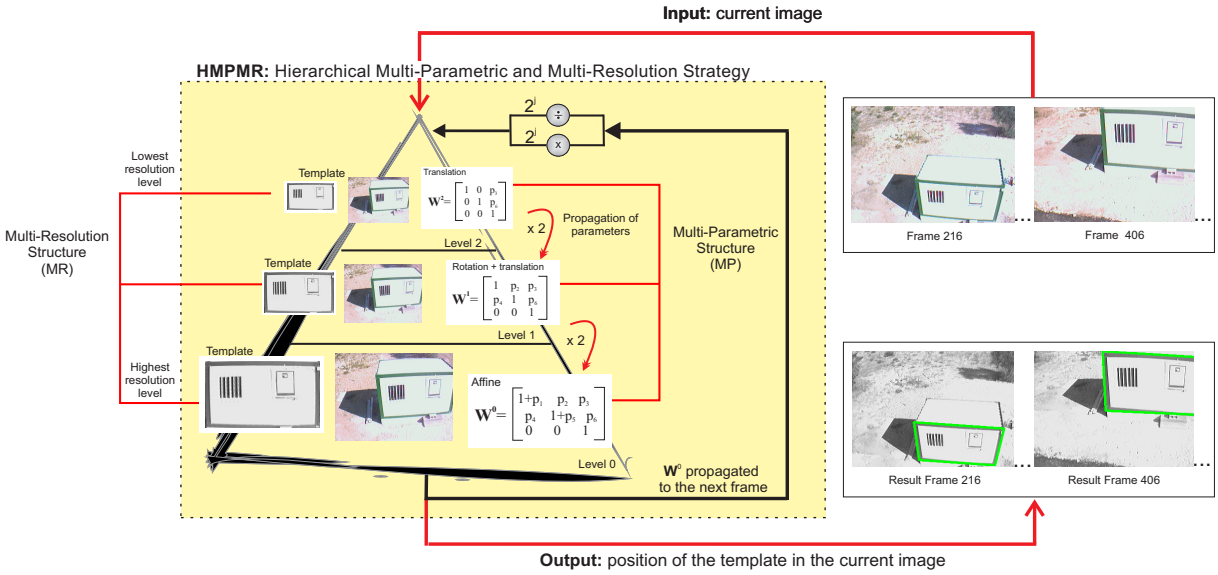


Figure 4.2: Hierarchical tracking strategy. Multi-parametric (MP) and multi-resolution (MR) strategies are used to solve the tracking problem on-board UAVs, especially when the range of motion between frames is large. Different motion models are estimated in each level. The motion model estimated in level 0 permits to find the position of the template image in the current image.

parameters that can be estimated with that information, which can be analyzed with the smallest eigenvalue criterion.

Additionally, in Figure 4.3 it can be seen that when the MR approach is used (blue/dark-solid bars, the first bars in each level), only the visual information of levels $0-3$ should be used to estimate the homography motion model. In those levels, the smallest eigenvalues are not close to 0. Conversely, when different configurations of the HMPMR-ICIA strategy are used (red/dark-dashed bars and cyan/light-dashed bars), it can be seen that the smaller eigenvalues of $\mathbf{H}$ are larger, which means that with the available information, the parameters selected for each level can be estimated.

Therefore, by using the HMPMR strategy in the example shown in Figure 4.1, the range of motion that the tracking algorithm can handle is increased compared with the one handled when only a MR approach is used. This is due to the fact that with the HMPMR structure more multi-resolution levels can be used.

Thus, by integrating the MP and the MR strategies, the tracking algorithm is robust to changes in position, by estimating only a few parameters in the lowest resolution level (e.g. translation), and refining them in the subsequent levels of the pyramid, using more image information, and more complex motion models. Additionally, this integration permits to obtain a robust estimation of motion models with a large number of parameters (e.g. affine, 6 parameters; homography, 8 parameters) with a low computational cost.

The HMPMR strategy requires the definition of different variables, such as the number of levels (pL) of the MR structure, and the motion models in the MP structure. All these parameters are defined in the following paragraphs. Additionally, the advantages of the HMPMR strategy are discussed and tested in Section 4.4. In that section, an analysis

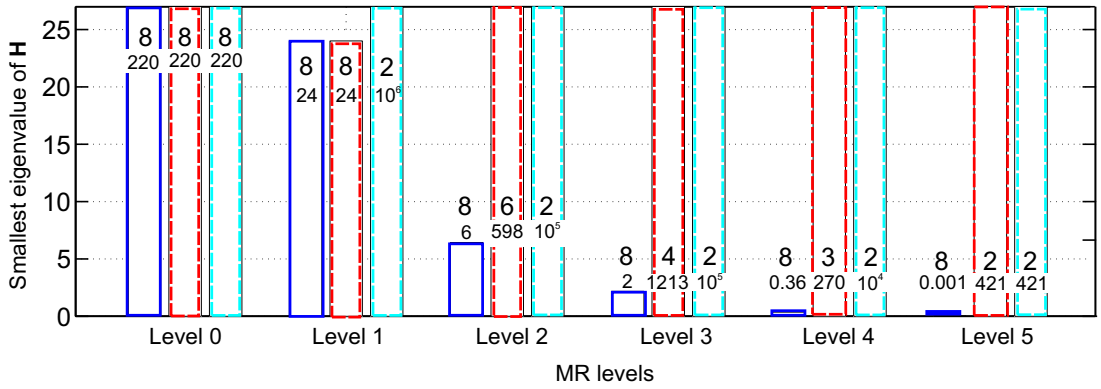


Figure 4.3: Comparison of the smallest eigenvalues of **H** found for the images of Figure 4.1

of how the configuration of the MP structure affects the performance of the tracking algorithm is presented.

4.2.1. Definition of Pyramid Levels

The different levels (pL) of the MR structure shown in Figure 4.2 are defined as a function of the size of the ROI that defines the template image **T** (i.e. the one that contains the object to track), in such a way that in the lowest resolution level (i.e. the $j_{\max}$ level), an image with no less than a defined number of pixels ($minPixels$) will be used. Therefore, pL is defined as follows, taking into account that the images are downsampled by a factor of 2:

$$2^{pL} = \frac{lowS}{minPixels} \tag{4.1}$$

where $lowS$ represents the lowest size between the width and height values of image **T** (the ROI that contains the object of interest). On the other hand, $minPixels$ is defined as the minimum size the template must have in the lowest resolution image, in order to have enough information to find the motion model of that level (e.g. 5 pixels could be enough to estimate the translation). Equation (4.1) permits to define the different levels of the MR structure automatically, and could be modified on-line according to the size of the template in the current image.

4.2.2. Definition of Motion Models

The MP structure suggests an increase in the complexity of the motion model as the resolution of the image increases. It is defined according to the motion model selected at the lowest level of the pyramid —the highest resolution level— $\mathbf{W}^0$ (where the superscript represents the level of the pyramid). In this level, $\mathbf{W}^0$ must be chosen as the best transformation that represents the motion of the object in the image plane. On the other hand, in order to ensure the detection of large frame-to-frame motions, the translation motion model is chosen for the highest level of the pyramid (the level that has the lowest resolution image): $\mathbf{W}^{j_{\max}}$ (where j represents the level of the pyramid, and is initialized as $j = j_{\max}$, where $j_{\max} = pL - 1$). This ensures a well-conditioned Hessian in the lowest

resolution level. Finally, the criterion to define the intermediate levels is to select them in such a way that a smooth transition of the parameters from the highest to the lowest levels of the pyramid is obtained (e.g. as shown in Figure 4.2).

If, for example, the homography is chosen in the highest resolution level $\mathbf{W}^0$ to represent the motion of an object in the image plane, then, assuming a hierarchical structure of four levels, the following combination of parameters could be used in the lower resolution levels:

$$\begin{array}{rcrcll} \mathbf{W}^{j_{\max}} & = & \mathbf{W}^3 & = & 2 \text{ parameters} & \rightarrow \text{Lowest resolution} \\ \mathbf{W}^{(j_{\max}-1)} & = & \mathbf{W}^2 & = & 3 \text{ parameters} & \\ \mathbf{W}^{(j_{\max}-2)} & = & \mathbf{W}^1 & = & 4 \text{ parameters} & \\ \mathbf{W}^{(j_{\max}-3)} & = & \mathbf{W}^0 & = & 8 \text{ parameters} & \rightarrow \text{Highest resolution} \end{array}$$

The combination of motion models of the MP structure is represented by the different numbers of parameters the motion model has. Therefore, in the previous example the MP configuration is in the form: 8-4-3-2, where the first number corresponds to the motion model $\mathbf{W}^0$ that will be estimated in the lowest pyramidal level (highest resolution image), which in this example is the homography (8 parameters). The last number corresponds to the motion model $\mathbf{W}^{j_{\max}}$ that will be estimated in the highest pyramidal level (lowest resolution image), which in this example is the translation (2 parameters); and the other numbers correspond to the motion models estimated in the intermediate levels, being them in the example the similarity (4 parameters) and the translation+rotation motion model (3 parameters).

Therefore, $\mathbf{W}^j$ models different 2D transformations (see Section 3.3.2.1). In our application, the assumption of 2D motion models is sufficient, considering that the tracking algorithm will be used for tracking planar surfaces (building inspection, helipad for landing) or non planar surfaces that can be assumed to be planar: e.g. the ground, when flying at high altitudes, as shown in [122]; or the drogue in Autonomous Air-to-Air Refuelling tasks [126].

An analysis of how the selection of the different parameters in the MP structure affects the tracking task is presented in Section 4.4.

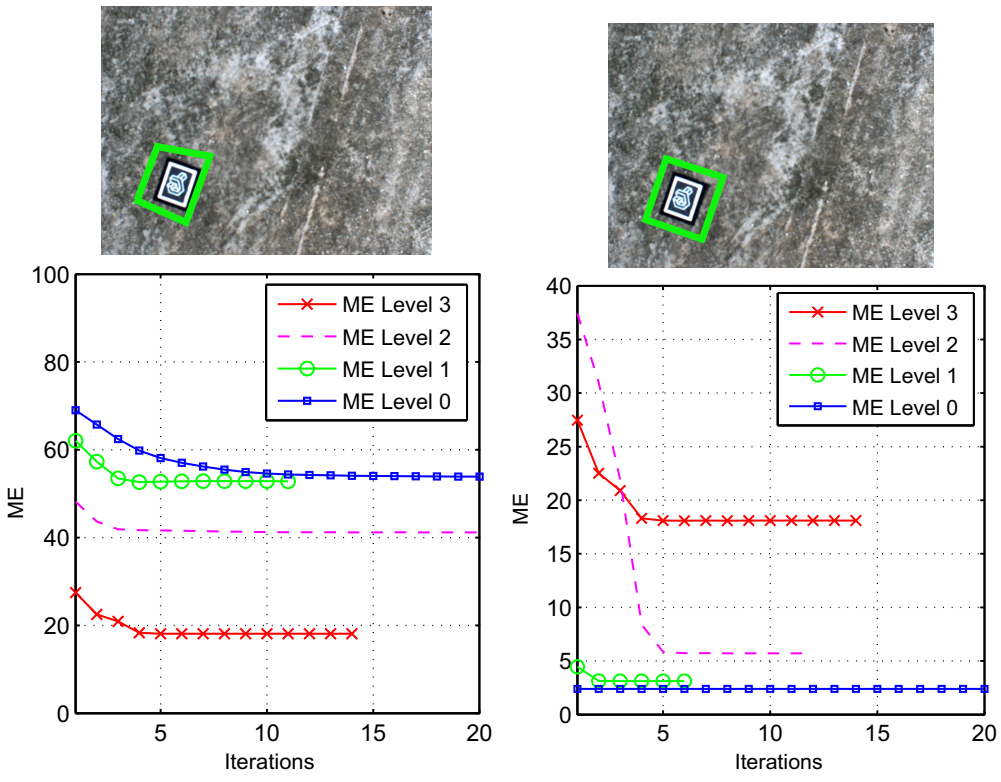
4.2.3. Propagation of Parameters

An important part of the HMPMR structure is the propagation of parameters inside the MR structure and among frames. As shown in Figure 4.2, the parameters that are estimated in each level by the image registration algorithm are used as an initial estimation of the motion for the following levels, taking into account that the images have been scaled by a factor of two, as follows:

$$\begin{array}{ll} p_i^{j-1} = p_i^j & for \quad i = \{1, 2, 4, 5\} \\ p_i^{j-1} = 2p_i^j & for \quad i = \{3, 6\} \\ p_i^{j-1} = \dfrac{p_i^j}{2} & for \quad i = \{7, 8\} \end{array} \tag{4.2}$$

Where the subscript i represents the parameter shown in Equation (3.14), and j represents the level of the pyramid, where $j = \{j_{\max}, j_{\max} - 1, \ldots, 0\} = \{pL - 1, pL - 2, \ldots, 0\}$.

Figure 4.4 shows the advantages of the propagation of parameters inside the MR structure of the HMPMR one. In Figure 4.4(b), it can be seen that when the parameters are propagated from level-to-level, the differences between the template and the current image (mean error ME) decrease in each level, and the template is correctly found in the current image (see thumbnail image). On the contrary, In Figure 4.4(a), when the parameters are not propagated, the ICIA estimates the motion model independently in each level. In Figure 4.4(a), it can be seen that the ME in the lowest level of the pyramid is the same found in Figure 4.4(b). However, as the resolution and complexity of the motion model increase, the ICIA without a hierarchy can not properly estimate the motion (see thumbnail image Figure 4.4(b)), due to the large frame-to-frame motion of this sequence. For that reason, the MEs that are found are higher.



(a) Without propagation of parameters. (b) With propagation of parameters.

Figure 4.4: Propagation of parameters in the HMPMR structure. The parameters that are estimated in each level by the image registration algorithm are used as an initial estimation of the motion for the following levels.

On the other hand, in level 0, another kind of propagation occurs from the lowest level of the pyramid (level 0) of the previous image $\mathbf{I}_{(F-1)}$, to the highest level of the pyramid ($j = j_{max}$) of a new image $\mathbf{I}_{(F)}$. This propagation is the basis of the tracking-by-registration algorithm, explained in Section 3.3.2.

The propagation is conducted as follows:

$$\begin{aligned} p^{j_{\max}}_{i_{(F)}} &= p^{0}_{i_{(F-1)}} \quad for \quad i = \{1,2,4,5\} \\ p^{j_{\max}}_{i_{(F)}} &= \frac{p^{0}_{i_{(F-1)}}}{s} \quad for \quad i = \{3,6\} \\ p^{j_{\max}}_{i_{(F)}} &= s p^{0}_{i_{(F-1)}} \quad for \quad i = \{7,8\} \end{aligned} \tag{4.3}$$

Where $s = 2^{j_{\max}}$ and f represents the number of the frame. This propagation of the parameters allows to validate the linearization of Equation (3.10) done by the image registration algorithm. Therefore, when a new image is analyzed by using the motion model $\mathbf{W}(\mathbf{x};\mathbf{p})$ estimated in the previous frame, the images $\mathbf{T}$ and $\mathbf{I}$ are close enough to find a minimum in the alignment of the images.

4.3. The HMPMR-ICIA Algorithm

Figure 4.5 shows the diagram of the HMPMR structure for tracking using the ICIA algorithm described in Section 3.3.1; and Algorithm 5 describes in more detail the different steps of the algorithm. We have selected the ICIA algorithm because of its efficiency, being this aspect very important for applications with UAVs. We propose to use the ICIA algorithm to deal with the efficiency problem of direct methods and the HMPMR strategy to improve the tracking task in situations where MR approaches are insufficient to cope with large frame-to-frame motion. Therefore, with the HMPMR-ICIA robust motion estimations are achieved, allowing to track objects during long periods of time at real-time frame rates.

As input, the algorithm requires the information of $\mathbf{I}_{(0)}$ (where the subscript represents the number of frames), and the $\mathbf{x}$ coordinates in $\mathbf{I}_{(0)}$ (i.e. in the initial frame) of the object to track. These coordinates can be found manually or automatically, e.g using template matching approaches [24]. Additionally, the algorithm requires the definition of the levels of the MR structure (pL), which is defined using Equation (4.1); and the definition of the different motion models in the MP structure $\mathbf{W}^{j}$. As explained previously, the definition of the motion models depends on the complexity of the task.

In general terms, the HMPMR-ICIA algorithm is based on two stages: the initialization and the tracking stages.

In the initialization stage (steps 1–6, Algorithm 5), the initial frame, e.g. $\mathbf{I}_{(0)}$, is downsampled according to the different levels (pL) of the MR structure. This creates, the template image $\mathbf{T}^{j}_{(0)}$ for each level, as shown in Figure 4.5. Additionally, in this initialization stage, for each level of the pyramid, the Hessian matrix (Equation (3.12)) $\mathbf{H}^{j}_{(0)}$ and its inverse are calculated. These steps, i.e. steps 1–6 of Algorithm 5, are carried out at the beginning of the tracking task and every time the template image is updated.

One of the advantages of the ICIA is that $\mathbf{H}$ is calculated in the initialization stage and not in every frame. Hence, before starting it is possible to know if $\mathbf{H}$ is well-conditioned for the kind of image to track and for the parameters chosen for each level. This advantage could be especially useful in an automatic update of the template, so that both the MP

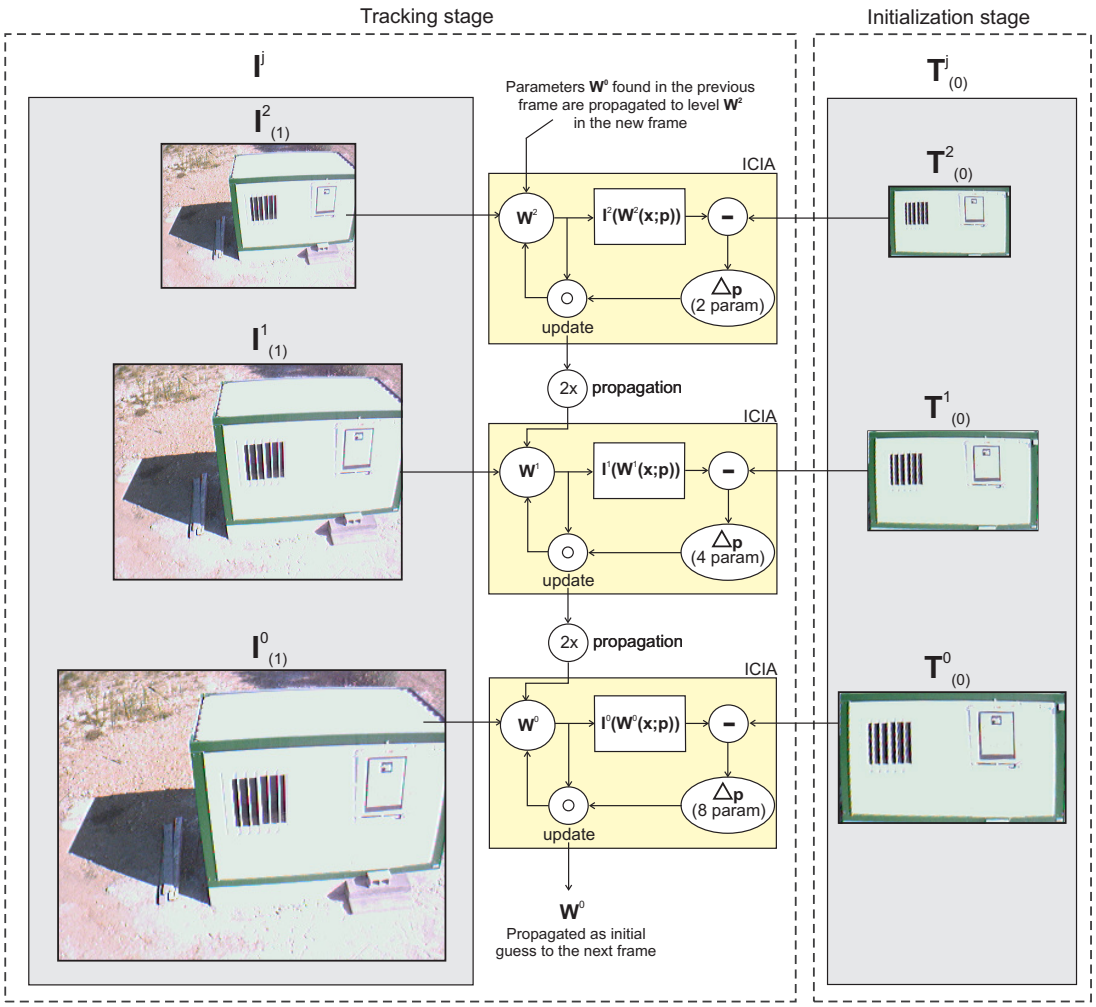


Figure 4.5: HMPMR-ICIA framework. Images $\mathbf{I}$ and $\mathbf{T}$ are downsampled to create the MR structure. In each level, the $\mathbf{x}$ coordinates in $\mathbf{T}$ are transformed by $\mathbf{W}^j$, and the errors between $\mathbf{I}^j(\mathbf{W}(\mathbf{x};\mathbf{p}))$ and $\mathbf{T}^j$ are calculated. Then, $\Delta\mathbf{p}$ is calculated, and the parameters are updated. The minimization is done with respect to the parameters defined in each level. When the stopping conditions have been reached, the parameters are propagated to the next level. At level $j = 0$, $\mathbf{W}^0$corresponds to the transformation that allows to find $\mathbf{T}$ in image $\mathbf{I}$.

and the MR structures of the tracker can change dynamically during the tracking task based on the new information of $\mathbf{H}$.

During the tracking stage, when a new image is analyzed (e.g. $\mathbf{I}_{(1)}$), it is first downsampled to create the MR structure, as shown in Figure 4.5 (tracking stage, left-dashed box). The motion model at the highest level of the pyramid $\mathbf{W}_{(1)}^{j_{\max}}$ (lowest resolution) is initialized using Equation (4.3), as shown in step 8 of Algorithm 5. Because this is the first frame analyzed in the tracking stage, $\mathbf{W}_{(1)}^{j_{\max}}$ is the identity matrix.

Then, for each level of the pyramid, as illustrated in Figure 4.5, the HMPMR-ICIA algorithm is applied as follows:

1. The coordinates $\mathbf{x}$ in $\mathbf{T}^j$ are warped using $\mathbf{W}^j$; and $\forall\mathbf{x}$, the error between $T^j(x)$ and $I^j(\mathbf{W}^j(\mathbf{x};\mathbf{p}))$, is calculated (steps 9-11, Algorithm 5).

2. The increment of the parameters is found using Equation (3.11) step 12, Algorithm 5.

3. The motion model is updated using Equation (3.13), i.e. step 3, Algorithm 5. In each level of the pyramid, the minimization is conducted only with respect to the parameters of the motion model defined for that level.

4. When the defined stopping conditions have been reached, the parameters are propagated to the next level of the pyramid using Equation (4.2), taking into account that the images have been scaled by a factor of two.

For each level of the pyramid j, as illustrated in Figure 4.5, the HMPMR-ICIA algorithm is applied. The process is repeated in the different levels of the pyramid. The complexity of the motion model increases as the resolution of the image increses, as can be seen in Figure 4.5. Therefore, at the lowest level of the pyramid (i.e the one that has the image with the highest resolution), the most complex motion model is estimated. This motion model $\mathbf{W}^0_{(F)}$ will contain the parameters that minimize the differences between the template and the current images, being this motion model the best approximation to the motion of the object in the image plane. With this information, it is possible to determine the position of $\mathbf{T}_{(F)}$ (i.e. the object to track) in the current image $\mathbf{I}_{(F)}$ (steps 15-16 Algorithm 5).

Finally, the motion model found in this frame (e.g. $\mathbf{I}_{(1)}$) is propagated as initial guess to the highest level of the pyramid $j_{\max}$ of the next frame (e.g. $\mathbf{I}_{(2)}$), using Equation (4.3). This propagation of the parameters from the lowest level of the pyramid in the previous frame to the highest level of the pyramid in the new frame permits to validate the linearization of Equation (3.10) (in the case of the ICIA algorithm), done by the image registration algorithm. This permits that when a new frame is analyzed by using the estimation of $\mathbf{W}_{(F-1)}$ in the previous frame as initial estimation of the motion in the current frame, images $\mathbf{T}_{(0)}$ and $\mathbf{I}_{(F)}$ are close enough to each other to find a minimum.

The pseudocode of the HMPMR algorithm using the ICIA alignment is presented in Algorithm 5.

input : $\mathbf{I}_{(\mathbf{0})}$,$\mathbf{W}^j$, pL, $\mathbf{I}_{(0)}$, $\mathbf{x}$
output: Transformation that tells where $\mathbf{T}_{(0)}$ is located in $\mathbf{I}_{(f)}$
Initialization stage
1. Downsample $\mathbf{I}_{(0)}$ according to pL and create $\mathbf{T}^j_{(0)}$
2. Initialize $j_{\max} = pL - 1$
for $j \leftarrow j_{\max}$ **to** 0 **do**
  3. Evaluate the gradient $\nabla\mathbf{T}^j_{(0)} = \left(\frac{\partial \mathbf{T}^{\mathbf{j}}{}_{(0)}}{\partial x}, \frac{\partial \mathbf{T}^j_{(0)}}{\partial y}, \mathbf{1}\right)$
  4. Evaluate the Jacobian $\mathbf{J}^j = \frac{\partial \mathbf{W}^j}{\partial \mathbf{p}}$ at $(\mathbf{x}, \mathbf{0})$
  5. Compute the steepest descent images $\mathbf{SDI}^j(\mathbf{x}) = \nabla\mathbf{T}(\mathbf{x})^j_{(0)} \frac{\partial \mathbf{W}^j}{\partial \mathbf{p}}$
  6. Compute the Hessian matrix and its inverse $\mathbf{H}^j = \sum_{\mathbf{x}} \left[\mathbf{SDI}(\mathbf{x})^{j^{\mathrm{T}}} \mathbf{SDI}(\mathbf{x})^j\right]$, and $\mathbf{H}^{j^{-1}}$
end
Tracking stage
foreach *new image* $\mathbf{I}_{(F)}$ **do**
  7. Downsample $\mathbf{I}_{(F)}$ according to pL to create $\mathbf{I}^j_{(F)}$
  8. Initialize $\mathbf{W}^{j_{\max}}_{(F)}$ according to Equation (4.3). If it is the first image, $\mathbf{W}^{j_{\max}}_{(F)}$ is the identity matrix
  for $j \leftarrow j_{\max}$ **to** 0 **do**
    repeat
      foreach $\mathbf{x}$ *in* $\mathbf{T}_{(0)}$ **do**
        9. Warp $\mathbf{W}^j_{(F)}(\mathbf{x};\mathbf{p})$ to find $I^j(\mathbf{W}^j_{(F)}(\mathbf{x};\mathbf{p}))$
        10. Compute $E^j = [I^j_{(F)}(\mathbf{W}^j_{(F)}(\mathbf{x};\mathbf{p})) - T^j_{(F)}(\mathbf{x})]$
        11. Compute $\mathbf{b}^j = \mathbf{b}^j + \mathbf{SDI}(\mathbf{x})^{j^{\mathrm{T}}} E^j$
      end
      12. Compute $\Delta\mathbf{p}^j = \mathbf{H}^{j^{-1}}\mathbf{b}^j$
      13. Update the warp $\mathbf{W}^j_{(F)}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}^j_{(F)}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}^j_{(F)}(\mathbf{x};\Delta\mathbf{p}^j)^{-1}$
    until $||\Delta\mathbf{p}^j|| \leq \varepsilon$;
    14. Propagate the parameters in $\mathbf{W}^j(\mathbf{x};\mathbf{p})$ to the next level using Equation (4.2)
  end
  15. Use $\mathbf{W}^0_{(F)}$ to find the position of $\mathbf{T}_{(0)}$ in $\mathbf{I}_{(F)}$
  16. Draw results
end

Algorithm 5: HMPMR-ICIA tracking algorithm

4.4. Evaluation of the HMPMR Tracking Strategy

Different tests have been conducted in order to evaluate the performance of the HMPMR-ICIA algorithm. In these tests, the advantages of extending the ICIA algorithm with a HMPMR framework are shown, especially under the presence of large frame-to-frame motions. Qualitative and quantitative results are presented.

In the first test, the MP structure of the HMPMR-ICIA algorithm and its influence in the tracking results are analyzed. In this test, the HMPMR-ICIA is compared with different configurations of the ICIA algorithm (MR-ICIA and without hierarchies). In a second test, the behavior of the algorithm under perspective changes is analyzed. In the second test, different configurations of the MP structure are analyzed, and the HMPMR-ICIA algorithm is compared with state of the art feature-based approaches: the SIFT [114] and the pyramidal Lucas Kanade [22] (KLT) algorithms. The KLT feature-based algorithm is one of the most widely used approaches for tracking objects in aerial images, mainly because of its efficiency (low computational cost). Concerning this, see [39, 69].

In the tests, different algorithms were used to compare the performance of the HMPMR strategy. Some of those algorithms are based on direct methods (test 1 and 2), and others are based on features (test 2). The different configurations of the ICIA algorithm (HMPMR-ICIA, MR-ICIA, ICIA) have been implemented in C + +; and the OpenCV libraries [24] have been used for managing image data. On the other hand, The KLT [22] feature-based tracking algorithm that was used is based on the implementation of the tracker included in the OpenCV libraries [24]. Additional functions were included in order to allow the estimation of different motion models. The maximum number of features was defined as 200, a window size of 5 was used, and different pyramid levels were used in the multi-resolution structure of the algorithm. Those levels were defined based on Equation (4.1). On the other hand, the SIFT algorithm is based on the implementation developed by Rob Hess [85, 86]. The values of the different parameters that the algorithm requires correspond to the standard values that come with the implementation of the algorithm. Additional experiments of the HMPMR-ICIA tracking algorithm are shown in Chapters 5 and 7.

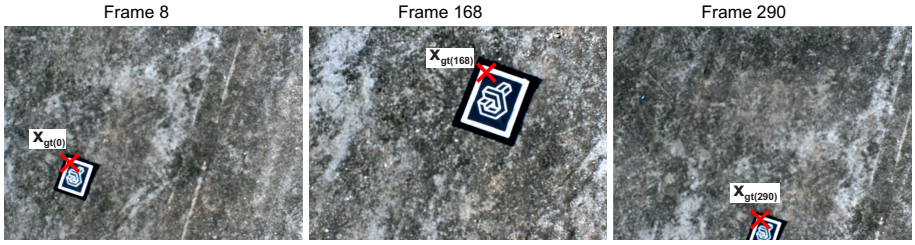
4.4.1. Test 1: Analysis of the MP Hierarchy

In this test, the selection of the MP structure is analyzed. Additionally, the performance of the HMPMR strategy during a tracking task recovering simple and complex motion models is evaluated. Different configurations of the ICIA algorithm are compared: the proposed HMPMR-ICIA, the ICIA algorithm without hierarchies, and the MR-ICIA (all of them are based on direct methods).

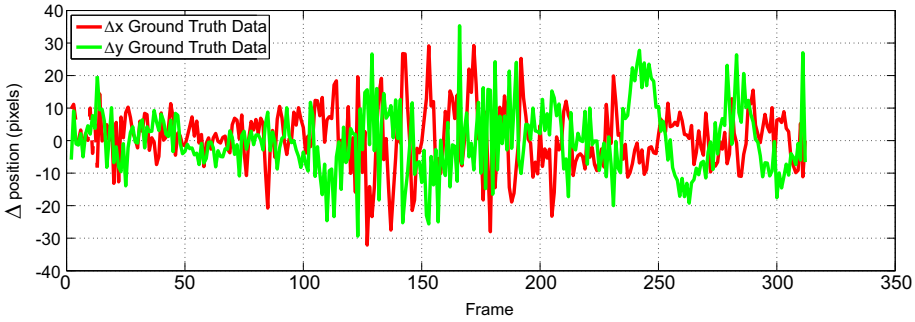
In the image sequence used in the test, the object to track is a flat symbol located on the ground. The size of the images is 640×480 pixels; and the size of the template is 115×125 pixels, so that according to Equation (4.1) $pL = 4$, considering $minPixels = 5$. It is important to mention that the selected image sequence contains large frame-to-frame motions (5, 10, and sometimes > 20 pixels), as can be seen in Figure 4.6(b), where the frame-to-frame change in position of the generated ground truth data is plotted. This characteristic makes this sequence challenging from the visual tracking point of

view, especially if it is considered that the linearization of Equation (3.10) is a good approximation when the frame-to-frame motion is < 1 pixel [94].

The algorithms are evaluated based on manually generated ground truth data (GT) in all the frames of the sequence, as shown in Figure 4.6(a) (the red cross in the upper-left corner).



(a) Manually Generated Ground Truth Data



(b) Δ in Position

Figure 4.6: Ground Truth Data (GT). The upper-left corner of the template is selected manually in all the frames of the sequence, see Figure 4.6(a). In Figure 4.6(b), the frame-to-frame changes in position of the generated GT data are plotted. This sequence contains large frame-to-frame motion, which sometimes is higher than 20 pixels, as can be seen in Figure 4.6(b).

The GT data in each image ($\mathbf{x_{gt}}_{(F)}$) is transformed into the coordinate system of the first frame (i.e. it is back-warped), using the transformation found by the tracking algorithm $\mathbf{x_{bw}}_{(F)} = \mathbf{W}_{(F)}^{-1}\mathbf{x_{gt}}_{(F)} = \mathbf{W}_{(F)}^{-1}(\mathbf{x_{gt}}_{(F)}; \mathbf{p})$. The accuracy of the motion model found with the different tracking algorithms is evaluated by measuring the Mean Absolute Error (MAE) of the upper left corner, as follows:

$$\text{MAE} = \frac{1}{n}\sum_{F=1}^{n} ME_{(F)} = \frac{1}{n}\sum_{F=1}^{n} \frac{|x_{bw_{(F)}} - x_{gt_{(0)}}| + |y_{bw_{(F)}} - y_{gt_{(0)}}|}{2} \tag{4.4}$$

Where ME is the mean error, n is the total number of frames of the image sequence, $x_{bw(F)}$ and $y_{bw(F)}$ are the back-warped coordinates of the GT point of each frame (F), and $x_{gt_{(0)}}$ and $y_{gt_{(0)}}$ are the coordinates of the GT point in the first image (in the frame where the tracking algorithm was initialized).

Based on the analysis of the ME of the corner's coordinates (see Equation (4.4)), the percentage of frames tracked by the different algorithms is examined (TF). If ME(F) > 2

pixels, the tracker is considered lost in that frame. This threshold was defined assuming that the error by "clicking" the GT data is ±2 pixels.

In Figure(s) 4.7, 4.8, 4.9, and 4.10, a collection of images that show the performance of the tested algorithms is presented. Figure 4.7 presents the results of the tracking task using the ICIA algorithm without any hierarchy, recovering different motion models with different numbers of parameters: 8 parameters (the homography, first row in the figure), 6 parameters (the affine transformation, second row), and 4 parameters (the similarity transformation, third row). The green/light box indicates the result of the tracking task. As can be seen in Figure 4.7, the large frame-to-frame motion of this sequence violates one of the main constraints of direct methods (small motion), and so the ICIA (without hierarchies) was not able to track the template in more than three frames in any of the tested configurations.

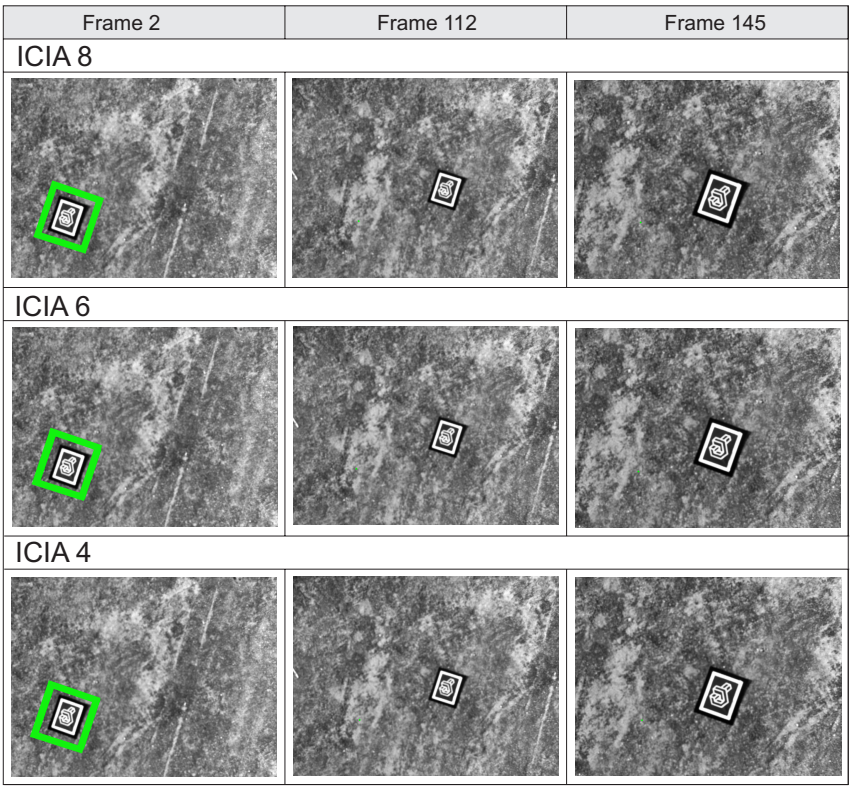


Figure 4.7: ICIA results. The green/light box indicates the result of the tracking task when recovering motion models with 8 (first row), 6 (second row), and 4 (third row) parameters. Without using any hierarchy, none of the tested configurations of the ICIA was able to track the template because of the large motion of the sequence (> 5 pixels).

Figure 4.8 presents the results of the MR-ICIA. In this configuration, the same number of parameters is found in the four levels of the hierarchical structure. The first row presents the results recovering the homography in the four levels of the pyramid (8 parameters), and the second and third rows present the results recovering the affine (6 parameters) and the similarity transformation (4 parameters), respectively. None of the tested configurations tracked the template throughout the sequence.

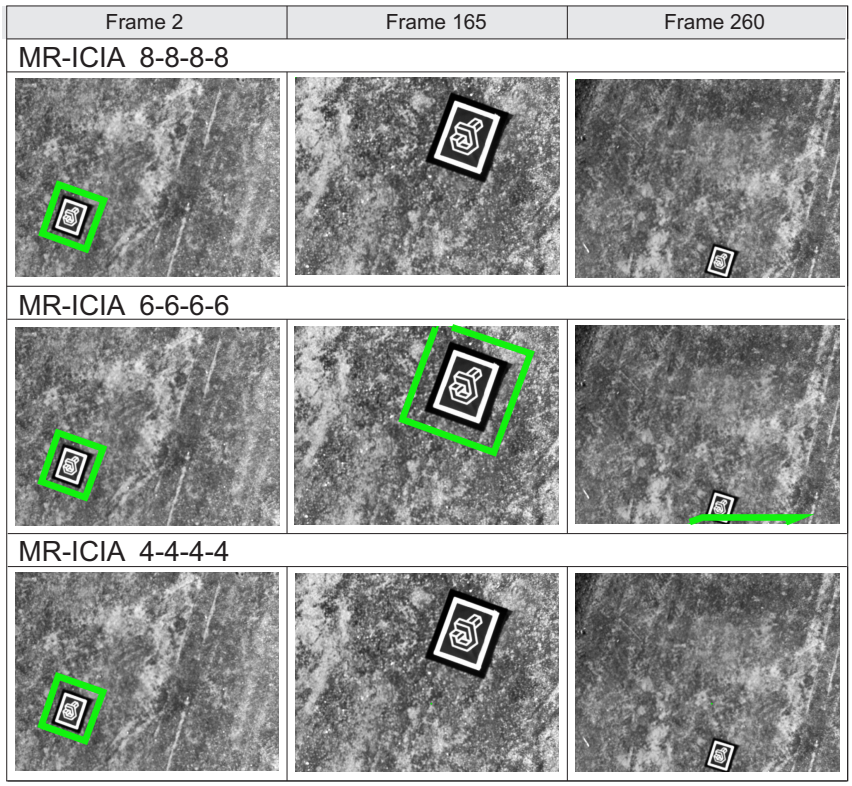


Figure 4.8: MR-ICIA results. The green/light box indicates the result of the tracking task when recovering 8 (first row), 6 (second row), and 4 (third row) parameters. As can be seen, none of the tested configurations of the MR-ICIA was able to track the template in all the sequence.

In Figure(s) 4.9 and 4.10, the results of the tracking task using the HMPMR-ICIA algorithm are presented. Different configurations of parameters were tested with this algorithm in order to analyze the behavior of the MP structure, and to define a criterion to select the motion models, especially under the presence of large frame-to-frame motions. The configurations that were tested allow to obtain different motion models in the highest resolution level, such as the homography, the affine, and the similarity transformations.

Figure 4.9 shows the different configurations of parameters that were able to track the template throughout the sequence. The first row shows the configurations that recovered the homography (8 parameters) in the highest resolution level; the second row, the ones that recovered the affine motion model (6 parameters); and the third row, the ones that recovered the similarity transformation (4 parameters). The images correspond to the configuration shown in bold letters (HMPMR-ICIA 8-6-4-2, HMPMR-ICIA 6-4-3-2 and HMPMR-ICIA 4-3-2-2), although the other configurations also obtained similar results when tracking the template (i.e. the template was tracked in all the frames of the sequence).

Conversely, Figure 4.10 shows the different configurations of parameters that were *not* able to track the template in the sequence. The first row shows the configurations that recovered the homography (8 parameters) in the highest resolution level; the second row, the ones that recovered the affine motion model (6 parameters); and the third row, the ones that recovered the similarity transformation (4 parameters). None of these configurations

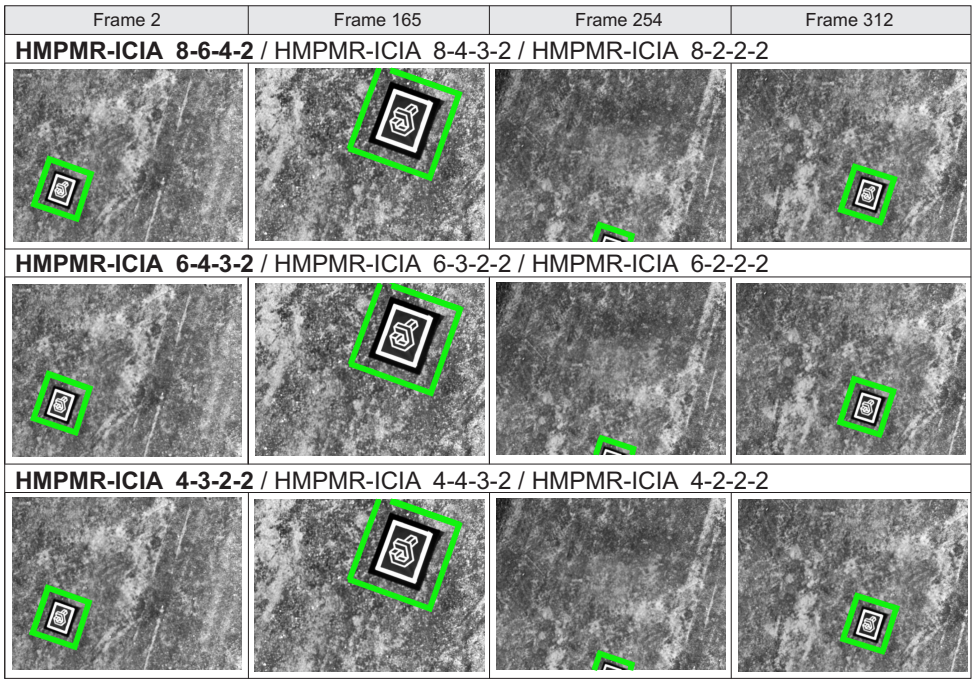


Figure 4.9: HMPMR-ICIA configurations that tracked the template. The green/light box indicates the results.

tracked the template throughout the sequence.

On the other hand, comparing the smallest eigenvalues of $\mathbf{H}^j$ in some of the configurations of the ICIA algorithm that were tested (the ICIA, cyan/dashed-light bars; the MR-ICIA 8-8-8-8, blue/solid bars; and the HMPMR-ICIA 8-4-3-2, red/dashed-dark bars), as those seen in Figure 4.11. It can be seen that although the smallest eigenvalues of $\mathbf{H}^j$ were > 1 in all the tested configurations, when the eight parameters of the homography are estimated in the smallest level of the hierarchical structure with the MR-ICIA 8-8-8-8 (red/dashed-dark bars), the smallest eigenvalue of the Hessian is much smaller than the one obtained when the translation motion model is estimated in this level with the HMPMR-ICIA 8-4-3-2 (cyan/dashed-light bars).

The analysis of the smallest eigenvalue of $\mathbf{H}^j$ can give us an idea (i.e. a confidence measurement), about the quality of the image information in an specific level for estimating a defined number of parameters. However, it cannot predict the behavior of the tracking algorithm. The performance of the tracking algorithm is conditioned by other factors, such as the amount of inter-frame motion, the maximum number of iterations defined in each level, the accumulation of errors frame by frame, or the application (e.g. is real time required?), among others. Unfortunately, some of these factors can not be predetermined, as for example happens with the amount of inter-frame motion; and other factors have been defined depending on the application, e.g. the maximum number of iterations is not too high to ensure real-time frame rates.

This is why when analyzing the smallest eigenvalues, the results found in this test show that although the Hessian was well-conditioned (smallest eigenvalue > 1) for estimating large numbers of parameters in the smallest levels of the MR structure, other factors such

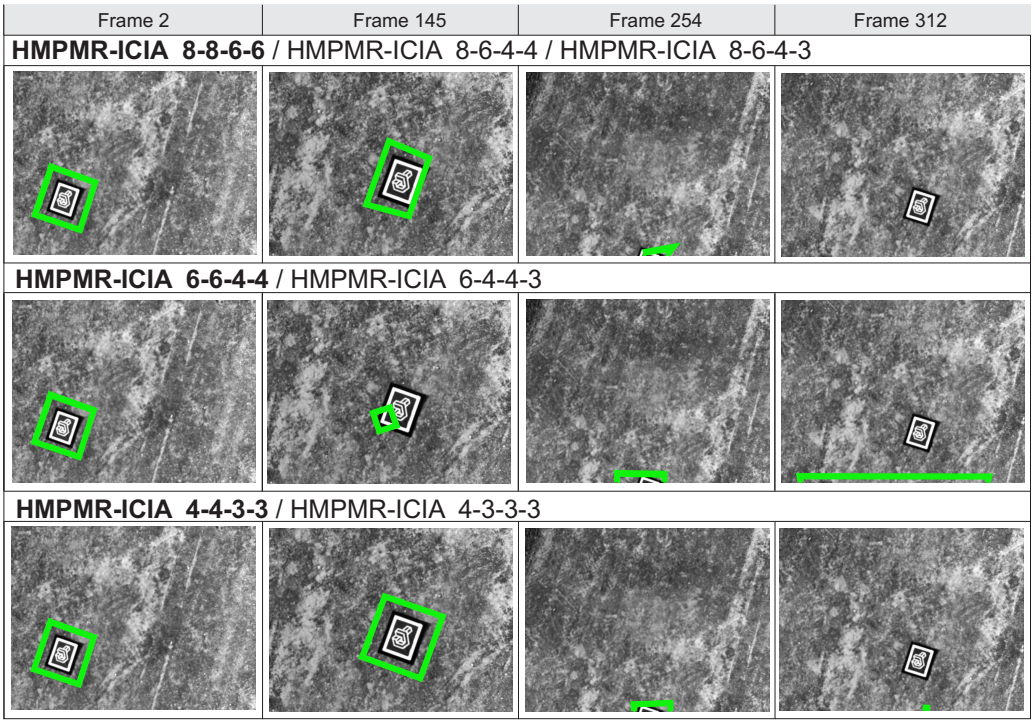


Figure 4.10: HMPMR-ICIA configurations that did *not* track the template. The green/light box indicates the results. As can be seen, the definition of the MP structure affects the behavior of the tracking algorithm. Some of the tested configurations are not able to track the template in all the sequence.

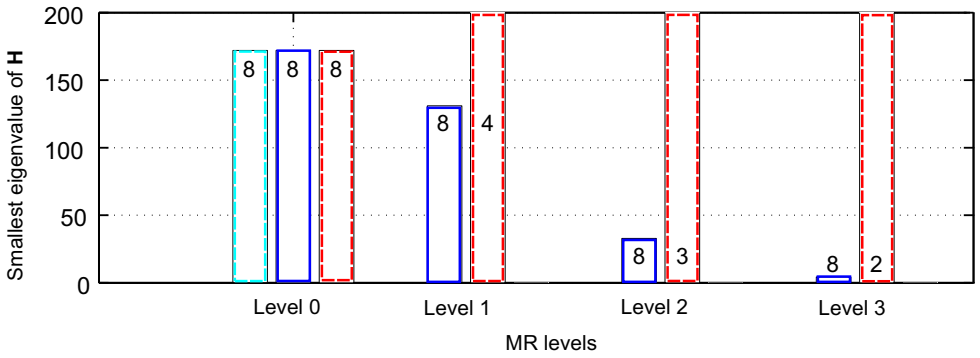


Figure 4.11: Comparison of the smallest eigenvalues of **H** with different configurations of the ICIA: ICIA without hierarchies (cyan/light-dashed), MR-ICIA (blue/dark-solid), HMPMR-ICIA (red/dark-dashed).

as the defined number of iterations per level and the large frame-to-frame motion of this sequence constrained the performance of the tracking algorithms. For this reason, the MR-ICIA 4-4-4-4, that estimated fewer parameters, was not able to track the template either.

Conversely, in Figure 4.11 it can be seen that when the HMPMR-ICIA algorithm estimated the translation motion model in the lowest level of the pyramid, the smallest eigenvalue of the Hessian was much larger than when other parameters were estimated.

Therefore, with these configurations, the Hessian was well-conditioned, the large frame-to-frame motion was overcome, and the template was tracked in the image sequence (see Figure 4.9). Additionally, in Figure 4.11 it can be seen that although the ICIA (without hierarchies) had a well-conditioned hessian for estimating the 8 parameters of the homography (see Figure 4.11), the large frame-to-frame motion made the tracker algorithm fail in the first two frames of the sequence (see Figure 4.7).

Figure 4.12 compares the optimization function of the algorithms in the frame where the ICIA 8 (i.e. without hierarchies) failed. It can be seen that because of the large motion of the sequence, the ICIA 8 (left plot) did not converge (in Frame 2 the frame-to-frame motion was ≈ 5 pixels). Conversely, the hierarchical structures of the MR-ICIA 8-8-8-8 (center plot) and the HMPMR-ICIA 8-4-3-2 (right plot) make it possible to overcome this large motion and find a minimum. Therefore, these algorithms tracked the template in that frame.

Figure 4.12 also permits to analyze the hierarchical structures of the MR-ICIA (center plot) and the HMPMR-ICIA (right plot). Taking into account that the same maximum number of iterations is defined in all the algorithms, in the plot of the center it can be seen that for the lowest resolution level (Level 3, blue/dark solid-squares line) the MR-ICIA obtained an ME that was smaller than the one obtained with the HMPMR-ICIA (right plot, Level 3, blue/dark solid-squares line). This is because the MR-ICIA estimated 8 parameters in that level, modeling the motion of the object in a better way than when only two parameters were estimated with the HMPMR-ICIA. Nonetheless, as can be seen in the plot on the right, as soon as the HMPMR-ICIA increases the complexity of the motion model with the resolution of the image, the obtained MEs are similar to the ones obtained with the MR-ICIA (see right plot, Level 2, black/dark solid line), with the additional advantage that less parameters have been estimated and therefore the possibilities of being trapped in local minimum are reduced.

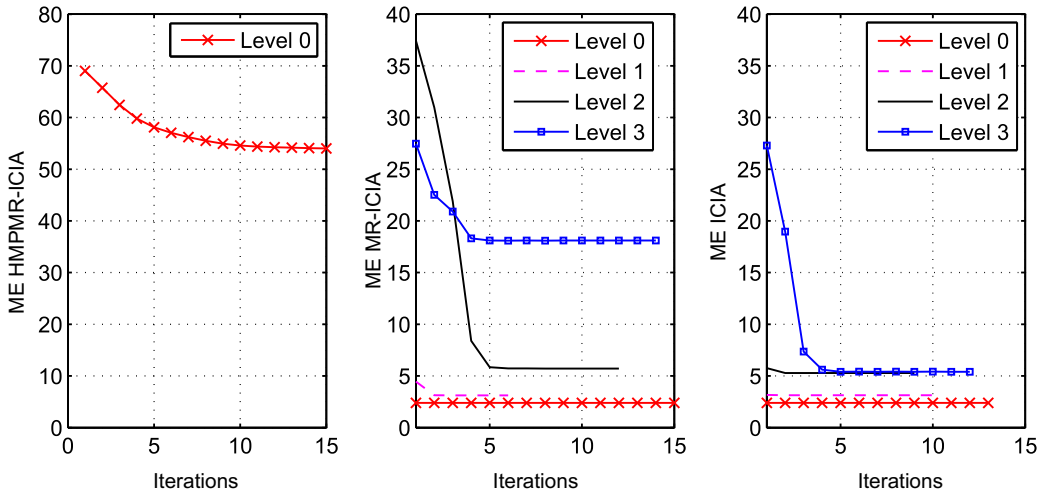


Figure 4.12: Comparison optimization function from Frames 2 to 3. Because of the large motion of the sequence, the ICIA 8 (left plot) fell to a local minimum. Conversely, the hierarchical structures of the MR-ICIA (center plot) and the HMPMR-ICIA (right plot) make it possible to overcome this large motion and find a minimum in those frames.

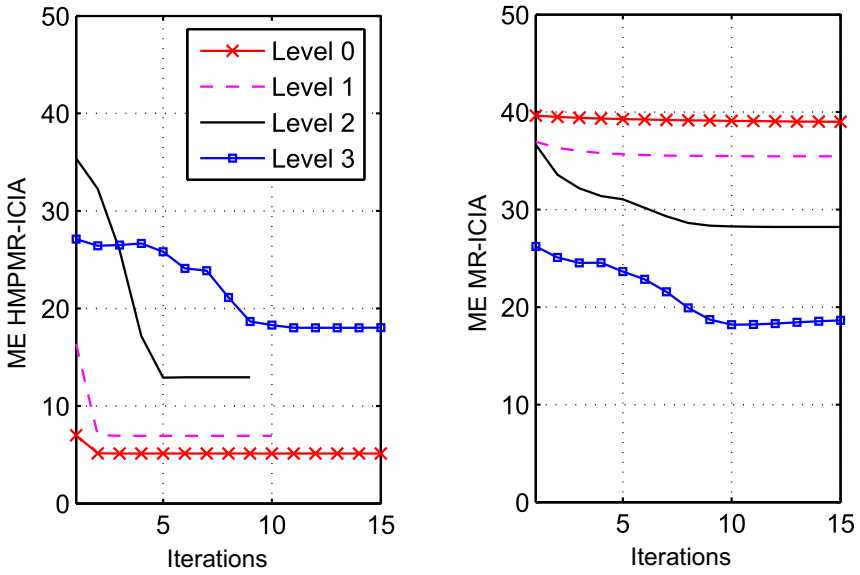


Figure 4.13: Comparison optimization function from Frames 109 to 110. Because of the large inter-frame motion $\approx$ 20 pixels, the MR-ICIA (right plot) fell to a local minimum. Conversely, HMPMR-ICIA (left plot) found a minimum in those frames.

On the other hand, analyzing the optimization function of the algorithms in the frame where the MR-ICIA 8-8-8-8 failed (Frame 110), in Figure 4.13 it can be seen that the MR-ICIA 8-8-8-8 (right plot) was not able to overcome the large frame-to-frame motion of the sequence (in Frame 110 the frame-to-frame motion was $\approx$ 20 pixels). Therefore, the minimum found in the lowest resolution level (Level 3), estimating 8 parameters, was similar to the one obtained when only 2 parameters were estimated with the HMPMR-ICIA 8-4-3-2 (see Level 3 in the plot on the left).

However, when the motion model found in Level 3 is propagated to the next level (i.e. Level 2), the MR-ICIA 8-8-8-8 was not able to improve the estimation of the parameters, falling in a local minimum, as can be seen in the plot on the right side of Figure 4.13. On the contrary, as can be seen in the plot on the left side of the figure, the HMPMR-ICIA 8-4-3-2 found a minimum thanks to the propagation of the 2 parameters from Level 3 to Level 2, that were well-estimated, and also thanks to the fact that in Levels 3 and 2 only a few parameters were estimated. This reduces the possibilities of being trapped in local minima.

As was pointed out in [206], the ICIA gives good results when the initial estimate is close to the global minimum. In the first part of the sequence, a MR approach was sufficient to satisfy this constraint (see Figure 4.7). However, when the frame-to-frame motion of the sequence increased, only the HMPMR hierarchy was able to deal with the large motion with configurations that estimated the translation (2 parameters) in the lowest resolution level, as can be seen in Figure 4.9.

Table 4.1 shows the results of the analysis of accuracy conducted to the three tested algorithms (HMPMR-ICIA, MR-ICIA, and ICIA without hierarchies), using the ground truth (GT) data shown in Figure 4.6. The GT data is used to analyze the recovered transformation (motion model). In Table 4.1, the mean absolute error MAE of the upper

left corner of the template is shown. Additionally, the percentage of tracked frames (TF) is presented.

Algorithm	MAE GT data	Tracked Frames TF [%]
ICIA 8	$1.03 * 10^5$	0.96(3 frames)
ICIA 6	$6.1 * 10^8$	0.32(1 frame)
ICIA 4	$1.9 * 8^8$	0.32(1 frame)
MR-ICIA 8-8-8-8	4.11	34.7
MR-ICIA 6-6-6-6	39.79	89.7
MR-ICIA 4-4-4-4	$7.6 * 10^4$	44.6
HMPMR-ICIA 8-4-3-2	0.7454	100
HMPMR-ICIA 8-6-4-2	0.7455	100
HMPMR-ICIA 8-2-2-2	0.7501	100
HMPMR-ICIA 8-6-4-3	62.034	80.385
HMPMR-ICIA 8-8-6-6	35.046	80.385
HMPMR-ICIA 8-6-4-4	$2.08 * 10^3$	44.373
HMPMR-ICIA 6-2-2-2	0.7576	100
HMPMR-ICIA 6-4-3-2	0.7577	100
HMPMR-ICIA 6-3-2-2	0.7572	100
HMPMR-ICIA 6-6-4-4	96.80	44.69
HMPMR-ICIA 6-4-4-3	126.62	80.38
HMPMR-ICIA 4-3-2-2	0.8314	100
HMPMR-ICIA 4-4-3-2	0.8317	100
HMPMR-ICIA 4-2-2-2	0.8317	100
HMPMR-ICIA 4-4-3-3	101.50	44.69
HMPMR-ICIA 4-3-3-3	61.87	80.38

Table 4.1: Analysis of accuracy conducted on the HMPMR-ICIA, the MR-ICIA, and the ICIA without hierarchies, using the ground truth (GT) data of Figure 4.6.

In Table 4.1, it can be seen that none of the configurations of the ICIA without hierarchies tracked the template in more than 3 frames. Additionally, these configurations obtained the highest MAE. On the other hand, with the MR-ICIA algorithm, it can be seen that the MR-ICIA 8-8-8-8 configuration was the one that tracked less frames (34%) but also the one with the smallest error in the estimation of the motion model (more degrees of freedom in the motion model allow to have a better representation of the motion, but the algorithm is more sensible to be trapped in a local minima). However, with the different evaluation criteria, it can be seen that none of the configurations of the MR-ICIA tracked the template in the sequence. This shows that a multi-resolution hierarchy is not always sufficient to solve the tracking problem when large frame-to-frame motions are presented.

The last section of Table 4.1 shows the results obtained with the different configurations of the HMPMR-ICIA. The shadowed configurations correspond to the ones that tracked the template throughout the sequence, overcoming the large frame-to-frame motion of the sequence (TF = 100%). It can be seen that a common characteristic of these configurations is that the translation motion model was estimated in the lowest resolution

level. Additionally, the tests with the different configurations reveal that the ones that recovered motion models with higher degrees of freedom in the highest resolution level (e.g. the homography, 8 parameters; and the affine, 6 parameters) were the ones that obtained the smallest MAE (e.g. HMPMR-ICIA 8-6-4-2 obtained MAE= 0.7455). Therefore, the results of this test indicate that by including the MP structure, it was possible to track the template throughout the sequence and to recover complex motion models, which was not possible to achieve using only an MR structure.

4.4.2. Test 2: Comparison with Feature-Based Algorithms

This test compares different configurations of the HMPMR-ICIA, recovering the homography motion model in the highest resolution level: 8-6-4-2, 8-4-3-2, and 8-2-2-2. Additionally, the HMPMR-ICIA algorithm is compared with three other algorithms: the MR-ICIA 8-8-8-8, based on direct methods; and the KLT and SIFT feature-based algorithms. In the image sequence used in the test, the front of a "house" is used as template image $\mathbf{T}$ (i.e. the object to track). The size of the images is 320×240 pixels, and the size of the template is 213×123 pixels, so that according to Equation (4.1) four pyramid levels are used. In this sequence, the images contain constant changes in positions because of UAV vibrations, changes in the appearance of the object to track (due to 3D movements), and loss of information when the object goes out the field of view (FOV) of the camera.

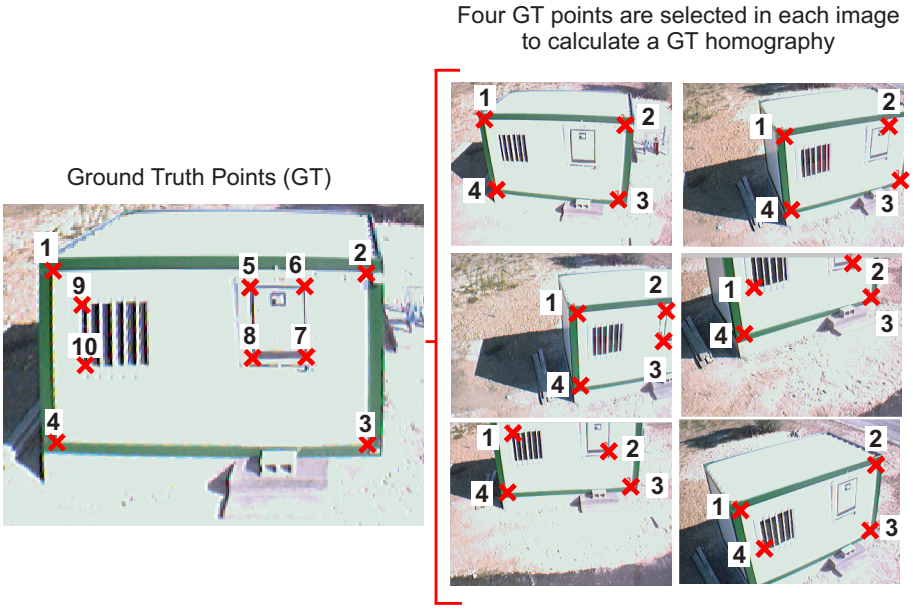


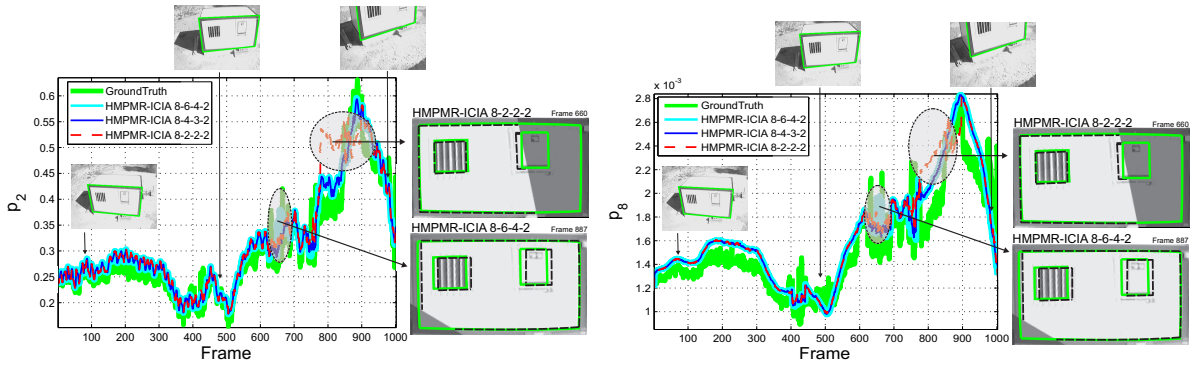
Figure 4.14: Ground Truth Data. Four points (right images) of the possible 10 points (left image) are selected to calculate a ground truth homography that relates points in the first frame to points in each frame of the sequence.

Ground truth (GT) data is used to evaluate the performance of the algorithms which is based on an analysis of the parameters of the homography estimated by the tracking algorithms. Figure 4.14 shows the GT data. As can be seen in the image located on the left, 10 different manually selected GT points can be used. Nonetheless, taking into

account that due to the movements of the UAV the front of the "house" goes out of the FOV of the camera, only 4 GT points (Figure 4.14, right images), well distributed over the template, are manually selected in each frame in order to calculate a ground truth homography ($\mathbf{x}^i_{(F)} = \mathbf{H_{GT}}\mathbf{x}^i_{(0)}$, for $i = \{1, 2, \ldots, 10\}$). This GT homography relates points in the first frame (Frame 0 where the template image was initialized) to points in each frame of the sequence.

In Figure 4.15, parameters p_2 and p_8 (see Equation (3.14)) of the GT homography (green/solid line) are compared with the ones obtained with three different tested configurations of the HMPMR strategy: the HMPMR-ICIA 8-6-4-2 (cyan/light solid line), the HMPMR-ICIA 8-2-2-2 (red/dashed line), and the HMPMR-ICIA 8-4-3-2 (blue/solid line). In these figures, it can be seen that around Frame 660 and Frame 887, the homographies recovered by the HMPMR-ICIA 8-6-4-2 (cyan/light solid line) and the HMPMR-ICIA 8-2-2-2 (red/dashed line) configurations present a small deviation from the ones estimated by the HMPMR-ICIA 8-4-3-2 (blue/solid line) and the GT data (green/solid line).

When a visual examination of the results was conducted, this small deviation was very difficult to perceive. Nonetheless, the images of the template shown in the right side of Figure 4.15 clearly show the small variation of the results. In these four images, the template image that was selected in the first frame (green/light lines) is overlapped with the back-warped template in Frames 660 and 887 (black/dashed lines). The back-warped template is found by transforming Frames 660 and 887 into the coordinate system of the first frame, using the homography calculated by the different configurations of the HMPMR-ICIA algorithm. If the estimated homography is good, the black/dark-dashed lines and the green/light-solid lines should coincide. However, analyzing these overlapped images, it can be seen that there is a small deviation in the homography that causes a slight difference between the templates.



(a) Parameter 2 (p_2) of the transformation

(b) Parameter 8 (p_8) of the transformation

Figure 4.15: Comparison of the estimated homographies with the GT data. The parameters p_2 and p_8 of the homography estimated with different configurations of the HMPMR strategy are compared with the ones obtained with the GT homography (green/light-solid line). Small errors, that are not perceived when analyzing the tracking results, are present in the HMPMR-ICIA 8-6-4-2 and HMPMR-ICIA 8-2-2-2 configurations.

This can also be seen analyzing the optimization function in those frames. Figure 4.16 shows the mean error, from Frame 886 to Frame 887, of the highest resolution level of the

configurations of the HMPMR structure that were tested. In the plot, it can be seen that because of the small number of parameters estimated in the intermediate levels (there was not a smooth transition of the parameters), the HMPMR-ICIA 8-2-2-2 configuration reached a minimum higher than the one obtained with the other two configurations. That is why in this frame the parameters of the HMPMR-ICIA 8-2-2-2 slightly differ from the GT ones. This causes the slight difference between the template and the back-warped image shown in the right side of Figure 4.16.

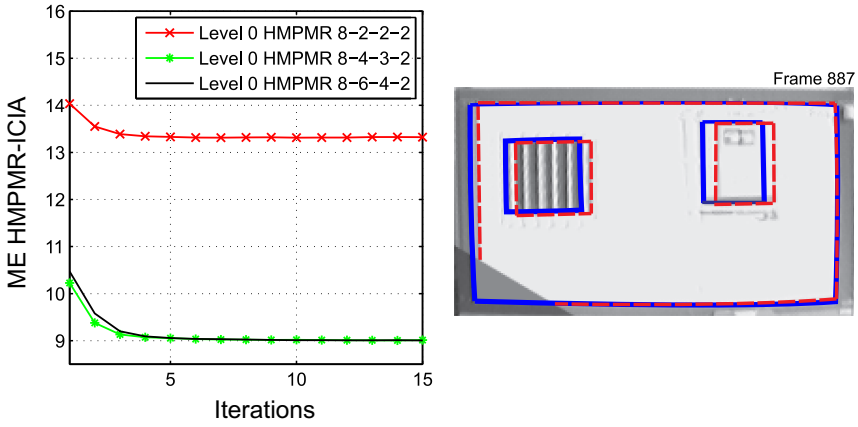


Figure 4.16: Comparison optimization function from Frame 886 to Frame 887. Because of the small number of parameters estimated in the intermediate levels (there was not a smooth transition of the parameters), the HMPMR-ICIA 8-2-2-2 configuration reached a minimum higher than the one obtained with the other two configurations.

These results permit to affirm that, although visually all the tested configurations of the HMPMR were able to track the template, the best results were achieved with the configuration HMPMR-ICIA 8-4-3-2 (blue/dark-solid line in Figure 4.15), which estimates at the lowest resolution level the translation motion model (2 parameters), at the highest resolution the homography (8 parameters), and that in the intermediate levels estimates a progressive distribution of the number of parameters (4 and 3 parameters). Therefore, we consider that the best configuration is the one that estimates the minimum required number of parameters. Although previous analysis revealed that the HMPMR-ICIA 8-6-4-2 and the HMPMR-ICIA 8-4-3-2 obtained similar results, the HMPMR-ICIA 8-4-3-2 configuration has less parameters. It means that this configuration can converge faster (as can be seen in Figure 4.16 green/light solid-dot line), avoiding the risk of being trapped in local minima.

On the other hand, in following figures the performance of the HMPMR-ICIA 8-4-3-2 algorithm is compared with the performance of the MR-ICIA 8-8-8-8 algorithm, which is also based on direct methods; and with the KLT and SIFT algorithms, based on features. Figure 4.17 shows a collection of images illustrating the performance of algorithms.

In Figure 4.17, it can be seen that the feature-based methods: SIFT (first row) and KLT (second row), failed to track the template, e.g. as seen in Frames 426, 669, and 1000. The direct method MR-ICIA (third row) failed in some frames (e.g. in Frames 426 and 669); whereas the HMPMR-ICIA tracked the template in all the frames of the sequence (fourth row).

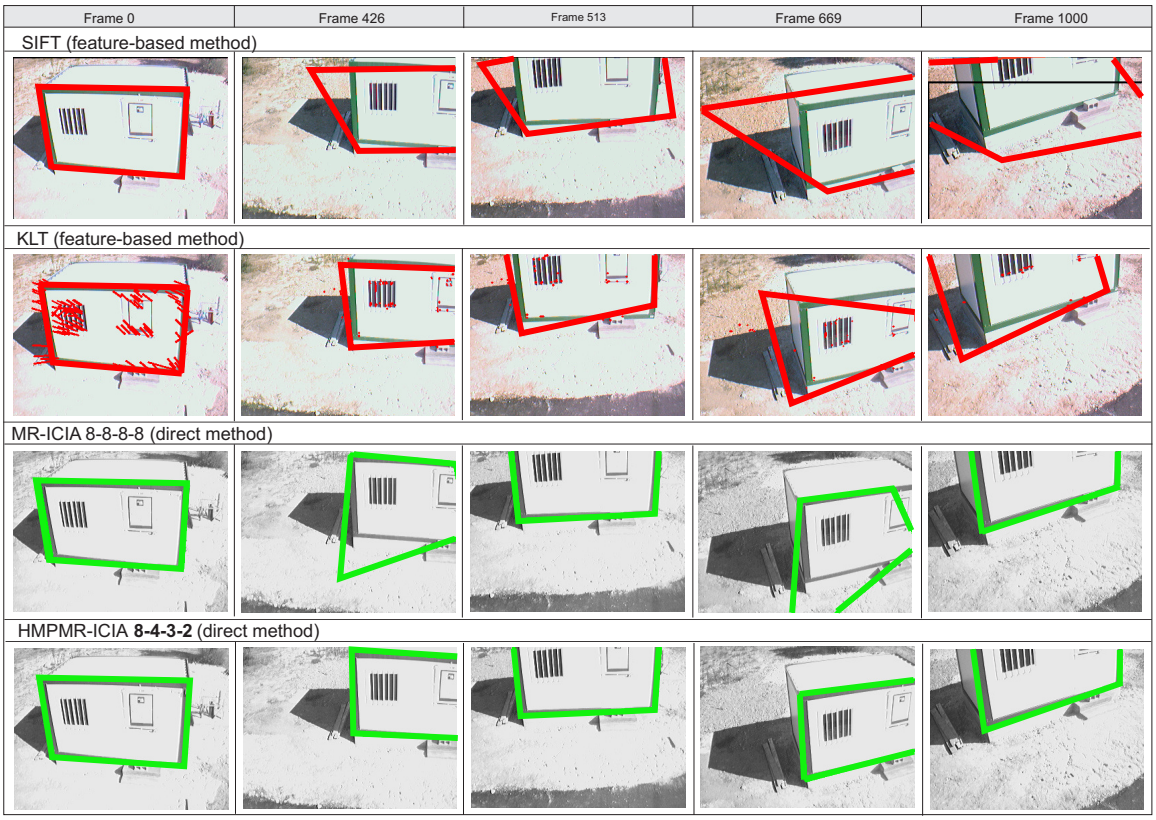


Figure 4.17: Comparison of tracking results. The polygons indicate the template found by the algorithms.

Additionally, when comparing some of the parameters estimated by each algorithm with the ones of the GT homography, the same results are found. Figure 4.18 shows the comparison among some of the parameters found by the HMPMR-ICIA (blue/dark-solid line), the MR-ICIA (black/dark-dashed-dot line), the KLT (red/dark-dashed line), the SIFT (cyan/light-dashed line), and the $\mathbf{H_{GT}}$ (green/light-solid line).

In Figure 4.18, it can be seen that, most of the time, the KLT (red/dark-dashed line) and the SIFT (cyan/light dashed line) failed to detect a correct transformation, in spite of the different features they found (the KLT found an average of 85 features and the SIFT found an average of 100 features). The parameters recovered by the tested feature-based algorithms differ from the ones of the $\mathbf{H_{GT}}$ (green/light solid line).

With respect to the algorithms based on direct methods, in Figure 4.18 it can be seen that the MR-ICIA (black/dark-dashed-dot line) fails in some parts of the sequence (e.g. Frames 426 and 669), but because the position of the template in the following frames coincided with the wrong position estimated by the MR-ICIA, the tracker recovered the template after Frame 775. On the other hand, it can be seen that the values of the parameters estimated by the HMPMR-ICIA algorithm (blue/dark solid line) do have behavior and values that are similar to the ones of the ground truth data (green/light solid line).

Table 4.2 shows the average speed of the four algorithms. As expected, the KLT obtained the fastest speed, an average speed of 27 frames per second (FPS) matching

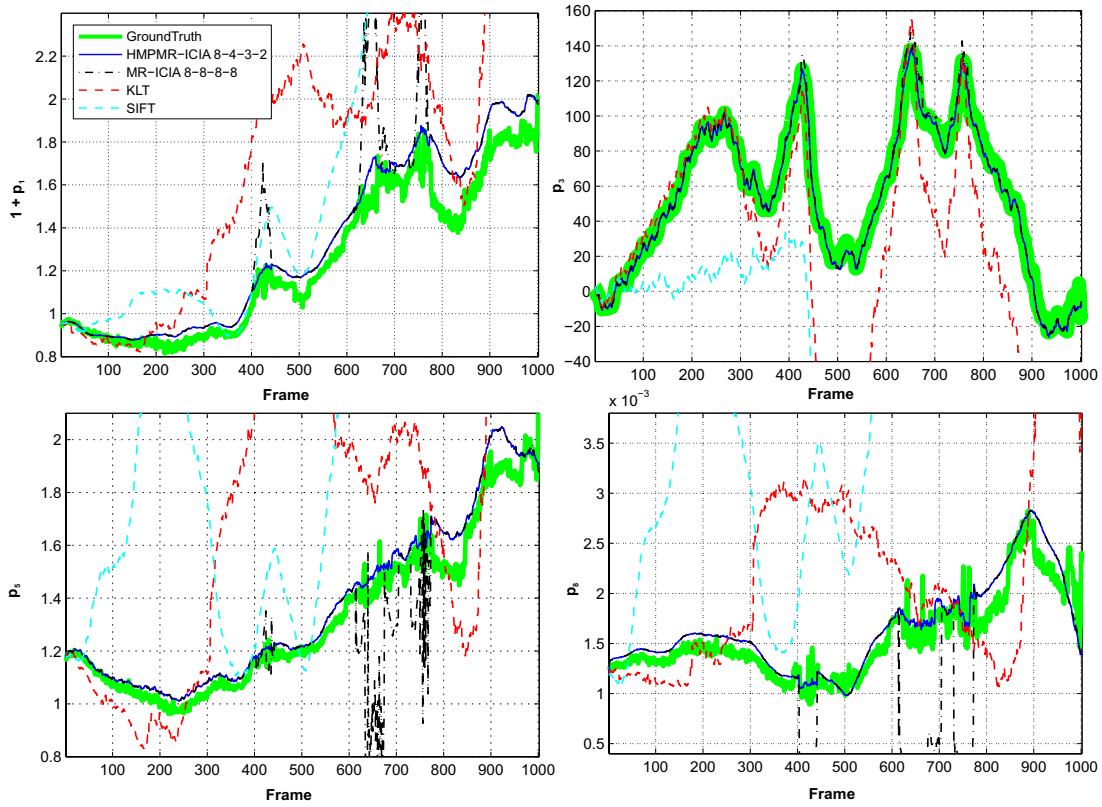


Figure 4.18: Comparison with $\mathbf{H}_{\mathrm{GT}}$ data. The parameters: p_1, p_3, p_5, and p_8 of the homography estimated with the tested algorithms are compared with the ones obtained with the GT homography (green/light-solid line).

≈ 85 features per frame. That is why the KLT algorithm is widely used in aerial images. The SIFT algorithm obtained an average speed of 3 FPS, obtaining the slowest speed. This is due to the high computational overheads in the different steps of the algorithm: e.g the calculation of the descriptor for each point, the matching of points, etc.

On the other hand, with the tested configurations of direct methods, we can see that by adding the MP structure, the HMPMR-ICIA algorithm tracks the template faster (16 FPS) than the MR-ICIA strategy (8 FPS), making it possible to reach real-time frame rates. It is important to consider that the direct method analyzes each pixel of the template in each level of the pyramid (around 26000 pixels must be analyzed only in the highest resolution level). Despite the amount of information the algorithm analyzes, we can see that by using the MP and MR strategies at the same time, a robust real-time tracking algorithm can be obtained.

4.4.3. Discussion

In the tests, images from real-flight tests were used to evaluate the performance of the HMPMR-ICIA algorithm. The main characteristic of these sequences was the large frame-to-frame motion. Different feature-based algorithms and different configurations of the ICIA algorithm to estimate different motion models were tested.

Algorithm	MP Structure	Frame-rate (FPS)
KLT	8-8-8-8	27
SIFT		3
MR-ICIA	8-8-8-8	8
HMPMR-ICIA	8-4-3-2	16

Table 4.2: Speed comparison feature-based methods and direct methods.

With the ICIA without hierarchies, the MR-ICIA, and the feature-based algorithms, the template was not correctly tracked in all the tested sequences. Conversely, with the different configurations of the HMPMR-ICIA, only those configurations that estimated the translation motion model in the lowest level of the hierarchical structure (e.g. Table 4.1 TF= 100%) were the ones that tracked the template in all the frames of the sequences, when estimating either simple or complex motion models. From the tests, it was possible to see that, by estimating a few parameters in the lower resolution levels, the algorithm converges faster to a solution close to the global minimum. In this regard, from the tests we could see that the MR approach is not enough to overcome frame-to-frame motions that are > 5 pixels, whereas a well configured HMPMR strategy can deal with larger frame-to-frame motions > 5 pixels, that can sometimes reach 30 pixels, as was shown in Figure 4.6(b).

An important aspect of the proposed strategy is the selection of the MP hierarchy. In the tests, an analysis of the distribution of parameters in the MR structure and its influence in the tracking results have been presented. It has been found that the most critical level is the one with the lowest resolution image (the highest level). We have proved that in order to have a well-conditioned Hessian, and in order to overcome large motions taking advantage of low resolution information, the translation motion model should be estimated in this level. On the other hand, in the highest resolution level the selected motion model should be the one that describes the motion of the object with the smallest number of parameters. In the tests, it has been shown that the HMPMR works fine estimating both complex and simple motion models in the highest resolution level. Finally, the tests have also shown that in the intermediate levels a smooth transition of the parameters, i.e. increasing the number of parameters, gives good results (e.g. configurations 8-6-4-2 and 8-4-3-2).

Therefore, from our experience with the algorithm, we have seen that the configuration that uses fewer numbers of parameters in the intermediate levels, but that smoothly increases the complexity of the motion model from level-to-level, is the one that gives better results in most of the situations (e.g. configurations 8-4-3-2 or 8-4-3-2-2).

One of the advantages of the ICIA algorithm is that an important part of the algorithm is calculated at the beginning of the tracking task (when the template is selected). Taking advantage of this, the eigenvalue analysis of the Hessian matrix $\mathbf{H}$ can give an idea, in advance, on whether the selected motion model can be estimated with the available image information. In the tests, it has been shown that the use of the MP structure improves the conditioning of $\mathbf{H}$ in the lowest resolution levels (the smallest eigenvalues are higher than when only an MR structure was used). Additionally, it was shown that when the MR structure is used, the extra flexibility in the motion model (more parameters) in the

lower resolution levels sometimes leads to bad local image warps, especially under lager frame-to-frame motions.

From the tests, it has been seen that the inter-frame motion plays an important role in the performance of the tracking task. This information is not available in advance and is critical in order to determine, for example, the maximum number of levels of the MR structure. The selection of the number of levels depends strongly on the image quality and therefore the maximum motion that can be estimated is constrained by this relation. Nonetheless, we have shown that the adopted criteria to select the number of MR levels, explained in Section 4.2.1, and the use of the MP structure inside the MR one, permit to obtain a tracking algorithm that tolerates larger frame-to-frame motions than the ones tolerated by other algorithms, such as the KLT, the SIFT and the MR-ICA.

One limitation of the HMPMR strategy has to do with the range of motion the algorithm can tolerate, which depends on different factors. One of those is the size of the template. When it is small, the MR structure can not have many levels, and therefore, the amount of inter-frame motion that the algorithm can tolerate is reduced. Additionally, when the frame-to-frame motion is so excessively large that it does not permit the estimation of the 2 parameters in the lowest resolution level, then the HMPMR algorithm will be trapped in a local minimum. In these cases, the incorporation of additional information, e.g. information from other sensors, could help the algorithm to have initial good estimations that permit to reduce the misalignment of the images, and make it possible for the algorithm to converge to a minimum.

Additional tests with different image sequences and different configurations of the HMPMR-ICIA algorithm can be seen in Chapters 5 and 7, where the algorithm was applied to solve the tracking problem in aerial refuelling tasks.

4.5. Results: Public Dataset

In this section, the proposed strategy is tested with Zimermmann's database [214], which has available ground truth data for each frame ($\approx$ 12000 frames). The sequences contain images of three planar objects: a MOUSEPAD (M), a TOWEL (T), and a PHONE (P). Although there are several publicly available datasets with ground truth information, we have found that this data set is the one that is best suited to test our tracking algorithm: e.g. the kind of template used is planar. Additionally, the kind of motions presented in the sequences are appropriate for the objective of the experiments: to analyze the performance of the HMPMR-ICIA algorithm under fast scale, perspective, and rotation changes; and also under partial occlusions of the object; which are examples of the motions encountered in our application (aerial images).

The same evaluation criteria shown in [214] are used to analyze the performance of the algorithm: the average error in object corners, expressed in percentage and normalized by the actual size of the object in the upper edge; and the loss-of-locks, defined in [214] as the cases where the error was higher than 25 percent in at least one of the corners. In those frames, the tracker was reinitialized using the ground truth positions.

The HMPMR configurations used in these sequences were created according to Equation (4.1): HMPMR-ICIA 8-8-4-3-2-2 for the M sequence; HMPMR-ICIA 8-8-4-2-2 for

T; and P sequences. Taking into account that the template images are big in the three sequences ($\approx 500 \times 400$ pixels), and that the number of pyramid levels is also high, in order to achieve real-time frame rates not all the pixels of the template in levels 0 and 1 were considered in the estimation of the motion model.

Figure 4.19 shows results of the tracking task conducted by the HMPMR-ICIA algorithm. The first row shows the results of the MOUSEPAD (M) sequence, and the second and third rows show the results of the TOWEL (T) and the PHONE (P) sequences, respectively. As can be seen in Figure 4.19, these sequences contain strong motion blur (e.g. images in the first and third rows); partial occlusions of the template (e.g. first row images); template going partially out of sight (e.g. images in the second and third rows); scale, rotation, and perspective changes; and repetitive structures in the object (e.g. phone buttons), among other features.

Figure 4.19: Results of the HMPMR-ICIA applied to the Zimermman's database. The first row shows the results of the MOUSEPAD sequence; the second and third rows show the results of the TOWEL and the PHONE sequences, respectively. The sequences contain strong motion blur, partial occlusions of the template, and fast 3D changes (e.g. scale, rotation, perspective)

Table 4.3 compares the results obtained with the HMPMR-ICIA algorithm with the results of state of the art algorithms that were applied to these sequences, which were reported in [214], [59] and [87]: NoSLLiP [214] (traker formed by a Number of Sequences of Learned Linear Predictors), SIFT [114], Lucas-Kanade tracker [115], the ICIA [15] (called IC in [214]), LLiP LS [99] (Learned Linear Predictors learned by the Least Squares method), ML-ALPs [87] (Multilayer Adaptive Linear Predictors) , and ZSPs [59] (Ultra-fast tracking based on zero-shift points). Most of these algorithms have been tested with the MOUSEPAD sequence, which is the longest one.

[1]Results reported in [214]
[2]Results reported in [59]
[3]Results reported in [87]

Method	Object	Frame-rate [FPS]	Loss-of-locks [−/−]	Error [%]
HMPMR-ICIA	M	15.25	4/6849	1.7
NoSLLiP[1]	M	18.9	13/6935	1.5
SIFT[1]	M	0.5	281/6935	1.4
LK (ICIA)[1]	M	2.5(25)	398/6935	2.4
LLiP LS[1]	M	24.4	1083/6935	6.3
LLiPLS 1/2 [1]	M	24.2	93/6935	3.0
ZSPs [2]	M	-	92	-
ML-ALPs[3]	M	17.2	1/6945	2.1
HMPMR-ICIA	T	16.8	1/3203	1.1
NoSLLiP[1]	T	21.8	5/3229	2.1
ZSPs[2]	T	-	8	-
HMPMR-ICIA	P	17.3	4/2259	1.1
NoSLLiP[1]	P	16.8	20/1799	1.8
ALPs[3]	P	96.7	10/2299	1.2
ZSPs[2]	P	-	17	-

Table 4.3: Results Zimermman's database. Comparison of the results of the HMPMR-ICIA applied to Zimermman's database [214] with the results reported in [214] [59], and [87].

Comparing the results, in Table 4.3 it can be seen that the performance of the HMPMR-ICIA in the different sequences is comparable with that of state of the art algorithms. The obtained number of loss-of-locks and the errors are low in the three sequences. From this table, it can be seen that when the SIFT is applied to the MP sequence, the frame rate is slower and the number of loss-of-locks is higher than the HMPMR-ICIA. On the other hand, it can also be seen that using the HMPMR strategy the results of the ICIA are improved (they were the same results found when the SIFT and the ICIA algorithms were tested with aerial images in Section 4.4).

On the other hand, Figure 4.20 shows some of the images where the HMPMR-ICIA failed, i.e. where loss-of-locks were detected. In the M sequence, the four detected loss-of-locks correspond to situations with acute perspectives after fast chaotic motions (see Figure 4.20, first row). The number of loss-of-locks obtained by the HMPMR-ICIA in this sequence (M) are smaller than the ones obtained by most of the other algorithms. Additionally, Figure 4.20, second row, shows examples of the frames where the loss-of-locks count increased for the P (on the right) and T (on the left) sequences.

With the P sequence, the loss-of-locks were found when the object was inclined, in front of the camera producing acute perspective effects in the image plane. In these cases, the HMPMR-ICIA did not lose the template completely, but the motion model was not the correct one. On the other hand, the motions found in the T sequence are very similar to the ones experienced in the application of tracking using cameras on-board aerial vehicles. This sequence contains strong motion blur (handheld camera), as can be seen in Figure 4.20, second row. Concerning this, the HMPMR-ICIA has shown to be robust to this kind of perturbation, obtaining the lowest loss-of-locks count of the algorithms that have been tested with this sequence (see Table 4.3).

Figure 4.20: Loss-of-locks of the HMPMR-ICIA. The first row shows the results of the MOUSEPAD sequence; and the second and third rows show the results of the TOWEL and the PHONE sequences, respectively.

4.5.1. Discussion

In this test, the performance of the HMPMR-ICIA algorithm was tested with a publicly available dataset, which has been used in the literature as benchmark for testing different tracking algorithms. Results have shown that a well configured HMPMR-ICIA permits to track planar templates affected by fast motion, partial occlusions, or motion blur, among other factors.

Additionally, the tests have shown that the HMPMR-ICIA obtains results that are comparable to the ones obtained with state of the art algorithms. Table 4.3 presents the results obtained with other algorithms. Nonetheless, a direct comparison with these algorithms is not conducted because their nature is different (some incorporate a learning stage).

4.6. Results: Tracking On-Board UAVs

The different tests described in the previous sections have shown that the proposed HMPMR strategy is able to track objects under different conditions, recovering low and large numbers of parameters with a performance that is better than the one obtained when using only an MR approach, in the case of direct methods; or than the one obtained when using some feature-based methods. In all these situations it has been shown that the HMPMR strategy has been able to track the object throughout the sequences.

In this section, the HMPMR-ICIA algorithm is used for tracking planar templates on-board UAVs. Two tests are conducted. In the first test, an on-board camera in a

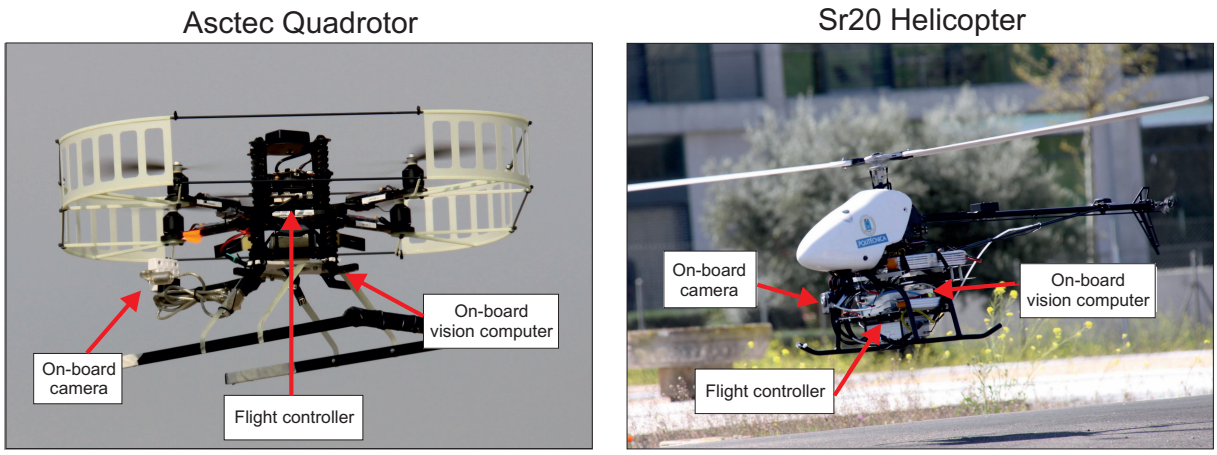


Figure 4.21: UAVs testbeds. The Rotomotion SR20 electric helicopter (image on the right) and the Asctec quadrotor (image on the left). On-board cameras placed in a downwards or a forwards looking configurations are in charge of collecting image data.

forwards-looking configuration is used to track a planar template. The performance of the HMPMR-ICIA is compared to the one obtained with different configurations of the ICIA algorithm: the ICIA without hierarchies, the MR-ICIA, and the HMPMR-ICIA (all of them based on direct methods). In a second test, an on-board camera in a downwards-looking configuration is used to analyze the performance of the algorithm under the visual conditions present in a vision-based landing task (large frame-to-frame motions and rapid changes in scale).

The data used in the tests correspond to images collected from different flights conducted with the Rotomotion SR20 electric helicopter (the Colibri III system) and the Asctec quadrotor (the Pelican system) testbeds [52], both shown in Figure 4.21. The systems carry on-board cameras placed in downwards or forwards looking configurations, as can be seen in Figure 4.21. Additional details of the systems are found in Appendix C.

4.6.1. Test 1: Behaviour Under Perspective Changes

In this test, the performance of the HMPMR-ICIA algorithm is evaluated when tracking part of a structure affected by the 3D motions of the UAV. A comparison of different configurations of the ICIA algorithm is conducted: the ICIA without hierarchies, the MR-ICIA, and the HMPMR-ICIA (all of them based on direct methods) are compared. In this test, the advantages of simultaneously using the MR and MP hierarchies during the tracking task are analyzed, especially when large frame-to-frame motions are presented.

The object to track corresponds to a flat section of a 3D structure. The UAV flies around the structure during the task. The selected image sequence contains large frame-to-frame motions (sometimes 5 and 10 pixels from frame-to-frame). This characteristic makes this sequence challenging from the visual tracking point of view. The size of the images is 640×480 pixels, and the size of the template is 84×170 pixels. Thus, according to Equation (4.1) the MR structure has 4 levels ($pL = 4$), assuming $minPixels = 5$. The camera on-board the UAV is in a forwards-looking configuration, and the homography (8 parameters) is chosen as the transformation that best describes the changes of the scene

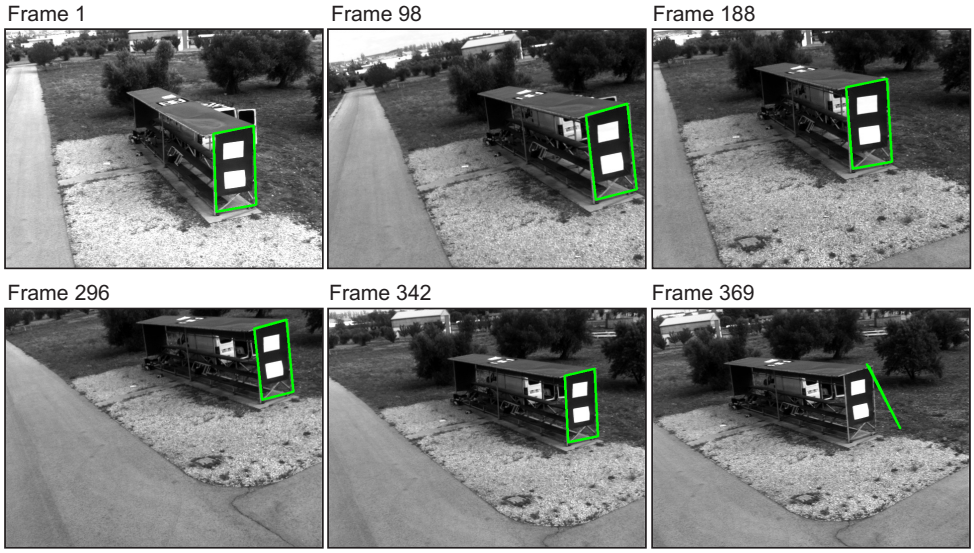


Figure 4.22: Tracking results of the ICIA. The green/light box indicates the result of the tracking task. Without using any hierarchy, the ICIA is not able to track the template when there are large motions in the sequence (> 20 pixels).

due to UAV movements.

Therefore, the ICIA estimates 8 parameters (the homography), i.e. no hierarchical structure is used; the MR-ICIA estimates the same number of parameters in the different levels of the MR structure (8-8-8-8); and the HMPMR-ICIA estimates different motion models in its MP structure (8-4-3-2): the homography in the lowest level of the pyramid (8 parameters), the translation in the highest level (2 parameters), and the similarity (4 parameters) and rotation+translation (3 parameters) motion models in the intermediate levels. The evaluation of the results obtained with the different algorithms that are tested is based on a visual examination of the tracking results, i.e. if the green/light box is covering the tracked area during the sequence.

Figure 4.22 presents the result of the tracking task using the ICIA algorithm without any hierarchy, which recovers 8 parameters (the homography). The green/light box indicates the result of the tracking task. As can be seen in Figure 4.22, the ICIA was not able to continue tracking the template after Frame 360. The large frame-to-frame motion in some parts of the sequence violates one of the main constraints of direct methods (small motion), and so the ICIA without hierarchies is not able to track the template in this sequence. This behavior was also found when the HMPMR structure was evaluated in Section 4.4.

Figure 4.23 presents a collection of images that shows the performance of the MR-ICIA during the tracking task. In this hierarchical configuration, 8 parameters are found in the four levels of the MR structure. The green/light box indicates the results in each frame. Analyzing Figure 4.23, we can see that the MR-ICIA 8-8-8-8 configuration fails after Frame 20. The MR-ICIA is not able to track the template in the image sequence. As mentioned in Section 4.2, an MR hierarchy is not always enough in our application to solve the tracking problem when large frame-to-frame motions are presented. Additionally, as was also mentioned in Section 4.2, one of the problems with MR approaches is that at

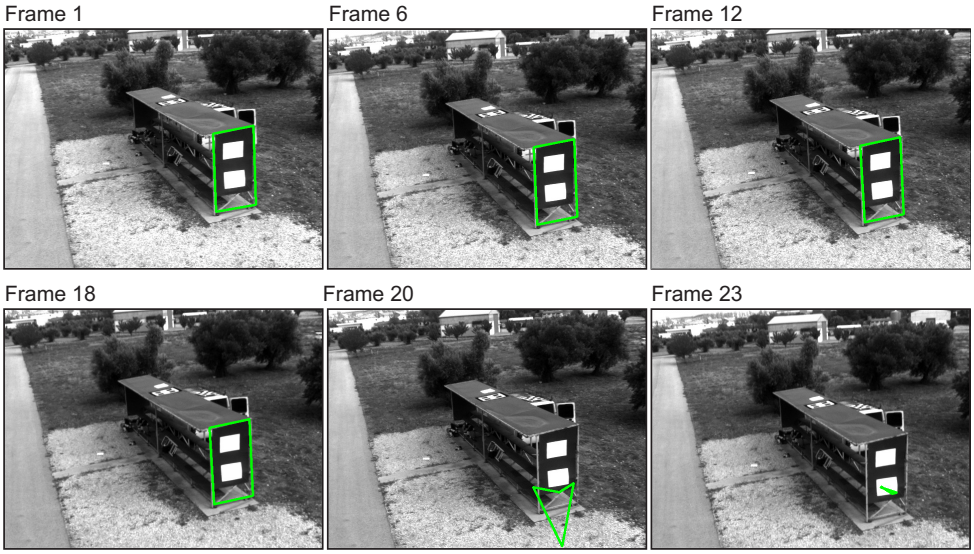


Figure 4.23: Tracking results of the MR-ICIA. The green/light box indicates the result of the tracking task. As can be seen, the MR-ICIA strategy can not track the template in all the sequence.

low resolutions the quality and quantity of the available information is not good enough to find a good estimation of motion models with a large number of parameters. For this reason, it can be seen that the MR-ICIA fails earlier than the ICIA algorithm without hierarchies.

Finally, in Figure 4.24 the results of the proposed HMPMR strategy using the ICIA algorithm are presented. Figure 4.24 presents a collection of images illustrating the performance of the tracking task using the HMPMR-ICIA algorithm. As can be seen in the images, the HMPMR strategy is able to track the template in all the frames in spite of the sudden motions the sequence has.

As a result of the different tested algorithms, it can be concluded that the MR approach is not enough to overcome frame-to-frame motions that are > 5 pixels, whereas a well configured HMPMR strategy can deal with large frame-to-frame motions > 5 pixels. Additionally, it can be said that the HMPMR is more robust than the MR approach recovering motion models with a large number of parameters. In this test, it has been shown that by configuring the direct method with MR and MP hierarchies, the results of the tracking task present a more robust behavior than when using only an MR hierarchy or none of the hierarchies. As a consequence of this, for our application using the ICIA algorithm with an HMPMR strategy is more robust than using only an MR approach.

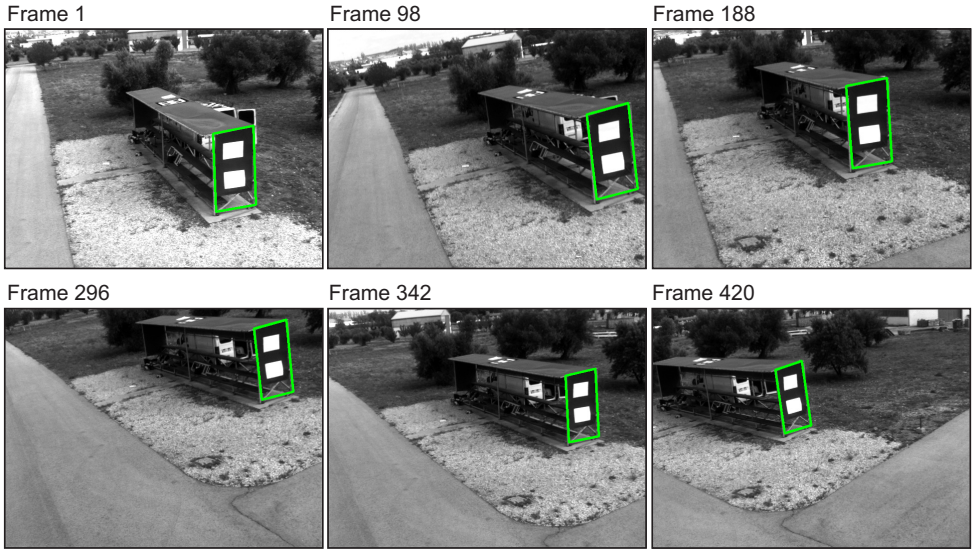


Figure 4.24: Tracking results of the HMPMR-ICIA. The green/light box indicates the result of the tracking task. The HMPMR-ICIA tracks the template in all the sequence.

4.6.2. Test 2: Performance During a Landing Task

4.6.2.1. Template Update Strategy

In tasks where the scale of the object changes, e.g. during take-off, landing, or during an aerial refuelling procedure, the appearance of the object to track can change notoriously throughout the task, as shown in Figure 4.25, especially when the object, from being far, gets closer (e.g. during landing). All these changes can affect the behavior of the algorithm by not satisfying one of the direct methods' constraints (appearance does not change over time).

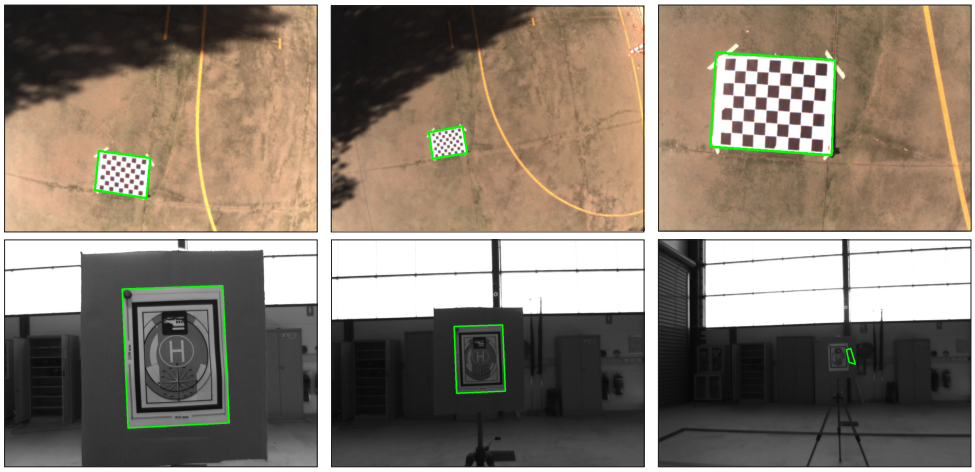


Figure 4.25: Changes in appearance due to scale changes. These changes can affect the behavior of the algorithms.

In order to deal with this problem during the landing task, we propose the use of a dynamic template. We call it dynamic because its intensity values will change according

to the size of the template in the current image. If its size (in the current image) is 2 times bigger or smaller than the size of the reference template, then the intensity values of the template are modified according to the values in the current template. Therefore, with this dynamic structure the template is modified automatically if it is so required, providing robustness to the tracking task under the previously mentioned circumstances.

An image sequence that contains different changes in scale of a planar template is used to analyze the advantages of updating the intensity values of the template. In this sequence, a FireWire camera is moved manually forwards and backwards. The camera captures images with a 1024×740 pixels size at a frame rate of 7.5 FPS. This small frequency creates a large frame-to-frame motion in the image data. In this test, the analysis of the results is conducted visually by analyzing the ROI, i.e. the green/light box found by the HMPMR-ICIA algorithm. On the other hand, the configuration of the HMPMR-ICIA that was used had the following combination of parameters: 8-8-4-3-2-2.

Figure 4.26 shows the results of the tracking task using the previously mentioned configuration. The first and second rows of Figure 4.26 present the results using the HMPMR-ICIA algorithm with a fixed template. The first row shows the template used, and the second row shows the results of the tracking task. On the other hand, the third and fourth rows of Figure 4.26 show the results obtained when the intensity values of the template are dynamically updated. The third row shows the different templates used throughout the sequence, and reveals that the intensity values of the template changed when the size of the template changed notoriously. The fourth row shows the results of the tracking task when the update template strategy is applied.

With the results obtained, in Figure 4.26 it can be seen that by dynamically updating the intensity values of the template (see third row), the tracking task is improved in situations when the changes in scale affect the visual information of the template (fourth row). In the second row of Figure 4.26, where the same template was used in all the sequence, the tracker failed after Frame 102. Nonetheless, in the fourth row of Figure 4.26 it can be seen that by updating the intensity values of the template the tracking task was accomplished throughout the image sequence.

4.6.2.2. Tracking Tests

In this test, the HMPMR-ICIA 8-4-3-2 algorithm is used to track a planar template located on the ground, that simulates the landing area. A UAV (a quadrotor) flies over the landing area, and an on-board camera (USB camera) placed in a downwards-looking configuration is used to capture the images (of 640×480 pixels size). This image sequence contains large frame-to-frame motions > 10 pixels and rapid changes in scale (e.g. Figure 4.27, Frame 696), commonly found during a landing task. In this test, the performance of the HMPMR-ICIA algorithm is compared with the KLT feature-based algorithm.

Figure(s) 4.27 and 4.28 show a collection of images that illustrate the performance of the tracking algorithms. Figure 4.27 shows the results of the KLT. As can be seen in the images, the template that is used allows the detection of a large number of features, which helps the KLT to track the template in almost all the sequence. However, in the final part of the task (see Figure 4.27, Frame 696), large frame-to-fame motions (> 20 pixels) make the KLT algorithm fail.

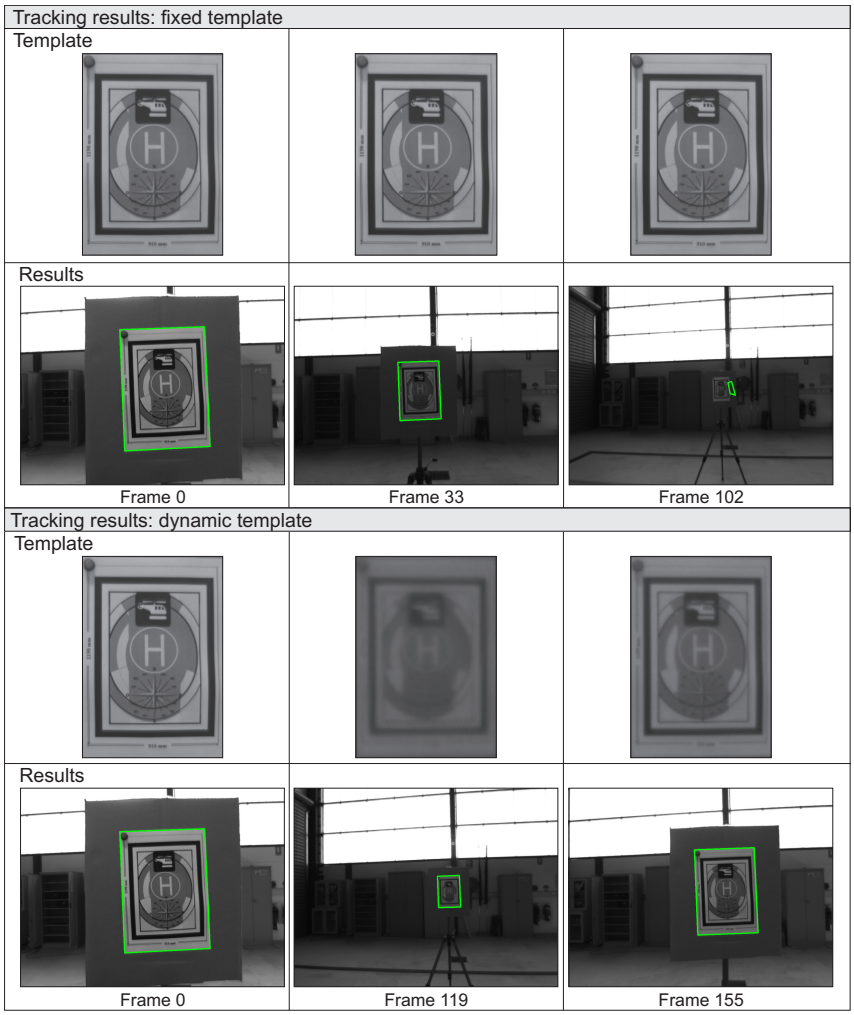


Figure 4.26: Template update results. The second row shows the results of the tracking task using the fixed template shown in the first row; and the fourth row shows the results when the templates shown in the third row are used. As can be seen in the fourth row, the tracking task is improved when a dynamic template is used.

On the other hand, Figure 4.28 shows the results of the HMPMR-ICIA algorithm. As can be seen, the HMPMR-ICIA algorithm was able to track the template in all the sequence, in spite of the large frame-to-frame motion of the sequence, vibrations (due to the UAV's movements), perspective changes (e.g. Figure 4.28, Frame 682), and scale changes (e.g. Figure 4.28, Frames 488 and 696), among other challenging factors.

In general terms, both algorithms performed well tracking the helipad during the landing task, except for the final part of the task where the KLT lost the template. The average frame rates reached during the task were: 22 FPS (KLT) and 17 FPS (HMPMR-ICIA). In the light of this test, it is possible to see that the performance and the speed reached by the direct method (HMPMR-ICIA) are appropriate for vision-in-the-loop applications.

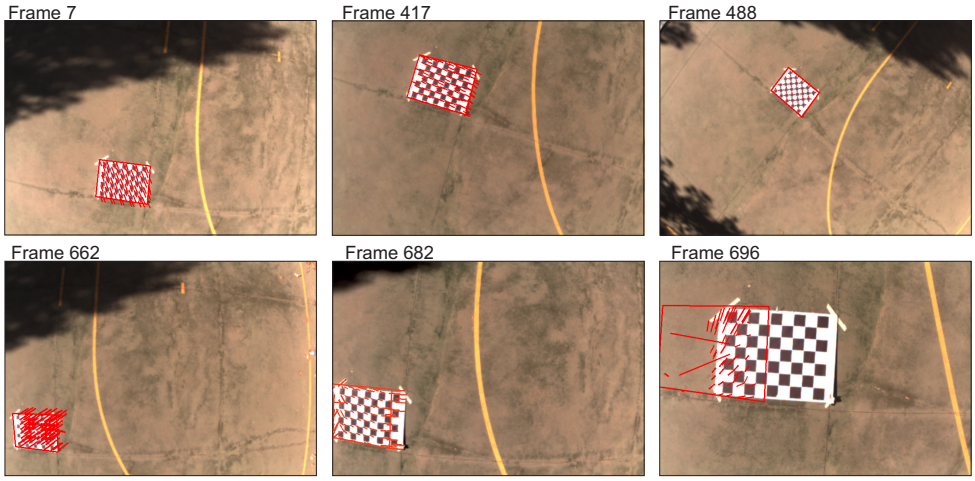


Figure 4.27: Tracking results of the KLT algorithm. The performance of the algorithms is analyzed under the visual conditions present in a vision-based landing task (e.g. large frame-to-frame motions and rapid changes in scale).

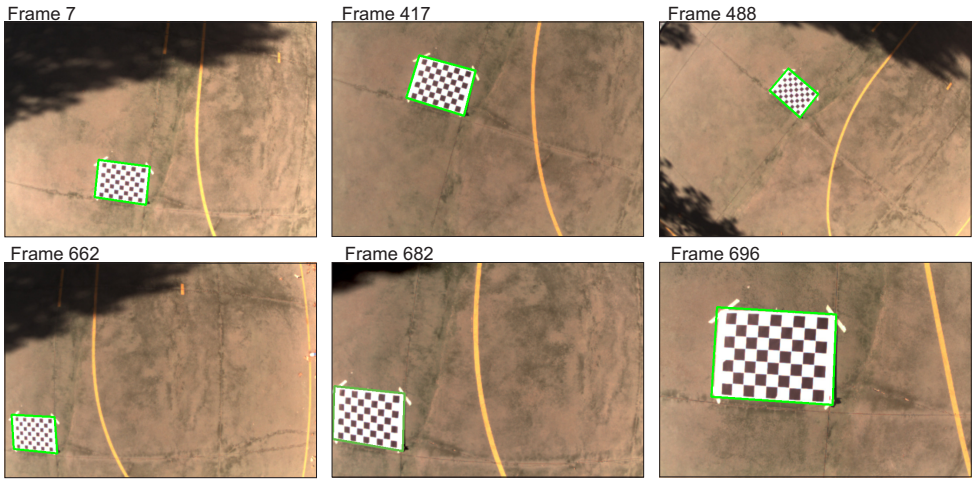


Figure 4.28: Tracking results the HMPMR algorithm. The performance of the algorithms is analyzed under the visual conditions present in a vision-based landing task (e.g. large frame-to-frame motions and rapid changes in scale).

4.6.3. Discussion

Tests were conducted in order to analyze the performance of the HMPMR-ICIA for tracking planar structures (or structures that can be assumed to be planar) with cameras on-board UAVs. The HMPMR-ICIA has been tested with other visual tracking algorithms, and the results show that the proposed HMPMR-ICIA performs better than the KLT feature-based tracker and other configurations of direct methods (as the ICIA without hierarchies and the MR-ICIA).

The results also show that the proposed tracking strategy can deal with large frame-to-frame motions, can recover complex motion models (e.g. the homography), can obtain real-time frame rates (during the tests speeds > 15 FPS were reached), and can recover robust information (i.e. robust homography) that can be used for different vision-based

applications on-board UAVs (e.g. building inspection, landing, take-off).

For vision-based applications, results have shown that the HMPMR-ICIA is able to track a planar template located on the ground, which is affected by the visual conditions of the task, e.g. fast scale changes. It has also been shown that for the landing task, a template update strategy (updating the intensity values of the template) can improve the performance of the algorithm. Template update strategies have the problem of drifting over time. However, taking into account that the landing task lasts for a few seconds, the drift is not a problem in this concrete application. On the contrary, we can take the advantage of the template update strategy to make the algorithm more robust to situations when the changes in scale affect the appearance of the object (e.g. when the UAV approaches the ground, the object can look darker than when the algorithm started).

Videos of the different tests described in this section can be found in [52].

4.7. Results: Tracking for Autonomous Air-to-Air Refuelling

Figure 4.29: Probe and drogue refuelling. Image from [126]

Autonomous refuelling tasks using the probe and drogue method strongly depend on the capacity of the receiver aircraft to sense the drogue's position. Due to the inherent difficulty of the task (the probe must be inserted into the drogue), fast, reliable and accurate relative position estimations are required in order to achieve a successful refuelling. In the literature, there have been a variety of computer vision solutions for the autonomous aerial refuelling problem (see Section 2.3.1), both for the flying boom and for the probe and drogue methods, being most of the approaches based on the detection of features

that are either artificial landmarks placed on-board or features, such as corners, extracted from the vehicles.

The probe and drogue method [176], shown in Figure 4.29, consists in a tanker aircraft that deploys a flexible hose with a drogue attached to its end. This drogue, as shown in Figure 4.29, is comprised of a canopy to provide stability, and of an automatic coupling for the fuel transfer. On the other hand, a receiver aircraft (Figure 4.29) is equipped with a probe that must be inserted into the passive drogue. When this is achieved, the automatic coupling is activated and the refueling process starts. For Autonomous Air-to-Air Refuelling (AAAR) capabilities, the receiver aircraft must be able to detect, track, and estimate the relative position of the drogue, taking into account that the drogue position in the image plane is affected by the effects of the tanker and the receiver aircrafts and by environmental conditions. Additionally, because rapid control corrections are required for docking, especially in turbulence conditions, to enable robust and safe operation of the receiver aircraft high-frequency updates of the drogue's position are also required.

The HMPMR-ICIA algorithm is used for solving the drogue tracking problem in AAAR tasks that are based on the probe and drogue method. In this application, the image template ($\mathbf{T}$) or object to track corresponds to the front of the drogue, as can be seen in Figure 4.30 (red/dark solid box $\mathbf{T}$). Therefore, taking into account the shape of the drogue and that the probe is always imaged by the camera, then the ICIA algorithm presented in Section 3.3.1 has been adapted to the drogue tracking problem. Therefore, instead of minimizing Equation 3.10, the ICIA now minimizes:

$$\sum_{\mathbf{x}} M^j(\mathbf{x})[T^j(\mathbf{W}^j(\mathbf{x};\Delta\mathbf{p})) - I^j(\mathbf{W}^j(\mathbf{x};\mathbf{p}))]^2 \tag{4.5}$$

where the sum is conducted only in those pixels $\mathbf{x}$ of $\mathbf{T}$ that meet the following condition:

$$\forall \mathbf{x} \in \mathbf{T}^j : P^j(\mathbf{W}^j(\mathbf{x};\mathbf{p})) = 0 \tag{4.6}$$

In Equations 4.5 and 4.6, $\mathbf{M}^j$ and $\mathbf{P}^j$ are masks created to exclude some pixels during the minimization process. As shown in Figure 4.30, $\mathbf{M}^j$ is a constant mask that is used in the minimization process to consider only the pixels in $\mathbf{T}^j$ that belong to the drogue (the drogue is circular, and as a consequence not all the pixels in image $\mathbf{T}^j$ should be used). $\mathbf{P}^j$ is another constant mask (the camera is always located in a defined and fixed position with respect to the probe). This mask is used to exclude the pixels of the probe when it is approaching the drogue. This mask plays an important role in the stability of the tracking algorithm, because during the final stage of the refuelling approach the drogue will be occluded by the probe.

On the other hand, the motion model ($\mathbf{W}$) chosen for the AAAR task was the translation + scale motion model (3 parameters, 2D position and scale), that was defined in Section 3.3.2. This motion model has been selected taking into account that due to the symmetric structure of the drogue's appearance, rotations around the Z_c axis (roll motions) and small rotations around the other axes do not have a significant impact on the visual characteristics of the drogue in the image plane. Additionally, $\mathbf{W}$ has been selected taking into account that during the task, when the probe moves towards the drogue, most

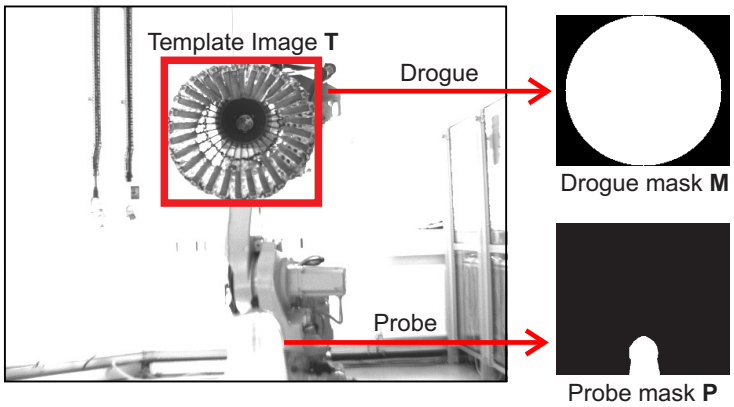


Figure 4.30: Masks **M** and **P**. The drogue mask **M** is used to exclude the pixels that do not belong to the drogue. The probe mask **P** is used to exclude the pixels of the probe when it is approaching the drogue.

of the imaged motion can be represented only by changes in position and in scale of the drogue in the image plane.

Therefore, the HMPMR-ICIA algorithm provides the 2D position estimation of the drogue in the image plane using all the pixels related to the drogue. The resulting estimation is considered to be robust under a large range of motions and with partial occlusions of the drogue. In the implementation of the different techniques that were used, it is assumed that the camera is calibrated, the dimension of the drogue is known, and that the motion of the drogue in the image plane can be modeled using three parameters: the two-dimensional position coordinates in the image plane and a scale value. As an extension, one aspect of the test regime presented in this section was the assessment of the behavior of the tracking algorithm when the motion of the drogue cannot be described by those three parameters.

This section presents results of the HMPMR tracking strategy applied to the area of AAAR. Different tests have been conducted in a realistic laboratory facility (see Figure 4.31), using real visual information of real refuelling hardware (a drogue and a probe). The performance of the tracking algorithm is analyzed in two different conditions. In the first test, the advantages of including the masks **M** and **P** in the tracking algorithm are shown, and the limits of the tracking algorithm in terms of speed and perturbation under which it can continue to function correctly are defined. Additionally, in a second test, motions of the probe towards the drogue simulating the refuelling procedure without turbulence, are used to test the tracking algorithm. In the latter, the performance of the HMPMR-ICIA is compared with the performance of a feature-based algorithm (the KLT).

4.7.1. Experimental Setup

The vision tracking algorithms were tested experimentally in a laboratory using real flight hardware (the AAAR testbed). A robotic cell has been used to reproduce the relative motion of the receiver (R2 in Figure 4.31) and the tanker (R1 in Figure 4.31) aircrafts (i.e. the relative motion of the probe and the drogue); and a camera mounted

at the base of the probe is in charge of sensing the drogue, as shown in Figure 4.31.

This robotic cell consists of two 6 degrees of freedom (DOF) robotic arms, one fixed to the ground at its base and the other mounted on a linear track. Actual flight hardware is mounted on the end of the robotic arms: a drogue is attached to the free end of the grounded robot (R1 in Figure 4.31) and a refuelling probe is fitted to the track-mounted device (R2 in Figure 4.31). The motion data sets for the tracking experiments are generated by a simulation of a tanker and receiver aircraft and their surrounding environment, constructed using Mathworks' MATLAB and Simulink software. Details of the implementation are found in [20]. Different scenarios can be simulated: none turbulence, light turbulence, and moderate turbulence.

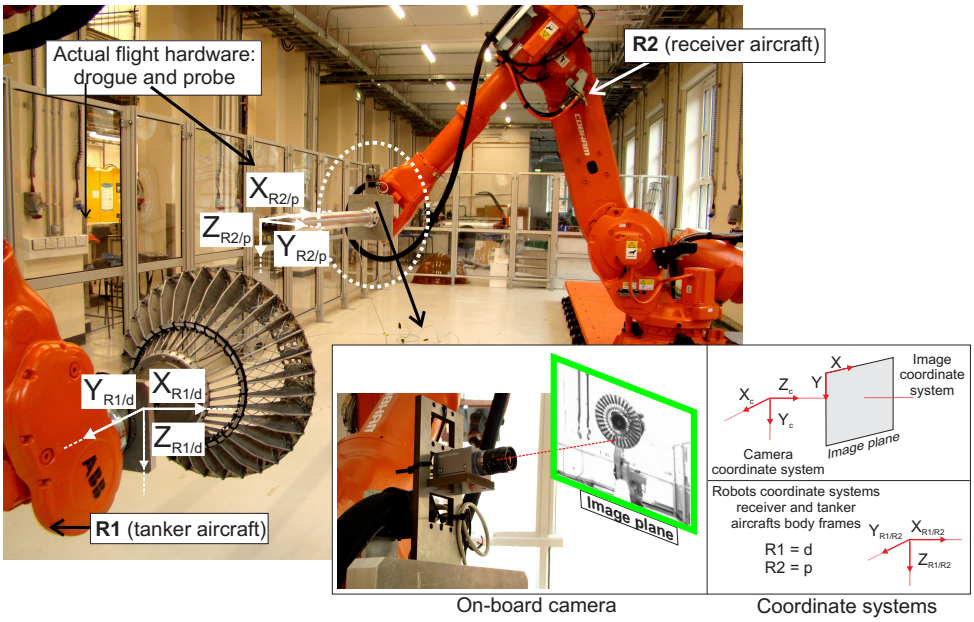


Figure 4.31: Testbed for autonomous air-to-air refuelling. A robotic cell is used to simulate the motion of the receiver (probe) and the tanker (drogue) aircrafts. A camera mounted at the base of the probe is in charge of sensing the drogue.

In order to compare the performance of the HMPMR-ICIA tracking the drogue, the well known feature-based KLT algorithm [22] is used. The configuration of the KLT is based on the implementation of the tracker included in the OpenCV libraries [24]. Additional functions were included in order to allow the estimation of different motion models. The maximum number of features was defined as 200, a window size of 5 was used, and three pyramid levels were used in the multi-resolution structure of the algorithm. The KLT estimates the 3 parameters motion model (see Section 3.3.2); and the HMPMR-ICIA algorithm estimates the following configuration of parameters in its MP structure: 3-2-2, the translation+scale motion model in the lowest level of the pyramid (3 parameters), and the translation motion model (2 parameters) in the higher level of the pyramid (lower resolution). On the other hand, the number of levels of the MR structure has been defined taking into account the distance at which the tracking algorithm starts operating ($\approx$ 6 m) and the size of the drogue in the image plane at that distance ($\approx$ 90 pixels). Thus, $pL = 3$.

4.7.2. Performance Evaluation of the HMPMR-ICIA for AAAR Tasks

4.7.2.1. Test 1: Advantages of Using the Masks M and P

In this test, an image sequence that contains different movements of the drogue and the probe is used to analyze the advantages of including the masks **M** and **P** presented in Equation (4.5). The analysis of the results is conducted visually, by analyzing the ROI (the red/dark square) found by the tracking algorithm.

The image sequence includes: basic motions of the drogue (moving left, right, up and down) that cause background information changes (e.g. Figure 4.32, Frames 236 and 2504); spiral movements of the probe that cause, in some situations, occlusions of the drogue by the probe (e.g. Figure 4.32, Frames 2716 and 2720); and inclinations of the drogue that produce changes in the appearance of the drogue (e.g. Figure 4.32, Frame 2782, in which due to the inclination the black circular center of the drogue is not well perceived).

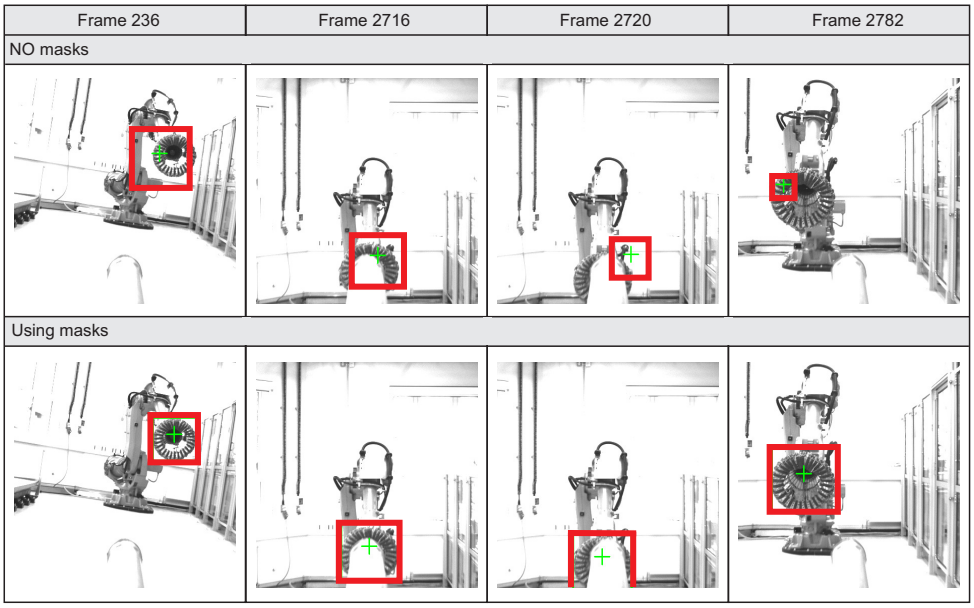


Figure 4.32: Tracking results including masks **M** and **P**. The first row shows the tracking results when the masks are not included in the tracking algorithm. As can be seen in the second row, the performance of the algorithm markedly improves when using the masks.

The first row of Figure 4.32 presents the results of the tracking task when the masks are not included in the tracking algorithm, and the second row shows the results when these masks are included. As can be seen in these images, by including the masks (**M** and **P**), the performance of the tracking algorithm improved. In all the situations where the algorithm without the mask failed, the algorithm that included the masks was able to track the drogue. This is because with the mask **M** the background information that surrounds the drogue is excluded, preventing changes in the background information from affecting the performance of the algorithm (see e.g. Figure 4.32, Frames 236, 2504, and 2782).

On the other hand, by including the mask **P**, the pixels that belong to the probe are also excluded from the minimization process. Therefore, the algorithm is more able to tolerate this kind of occlusion, as can be seen in Figure 4.32, Frames 2716 and 2720.

4.7.2.2. Test 2: Limits of the HMPMR-ICIA Algorithm

This test analyzes the limits of the HMPMR-ICIA algorithm for tracking the drogue. Specific motions of the probe and the drogue performed at a range of different speeds and with varying extent are used. The five motions are: drogue movements in the X_{R1}, Y_{R1}, and Z_{R1} directions at three different speeds: 400 mm/s, 1000 mm/s, and 2000 mm/s; a wave motion, in the form of combined sinusoidal variations of the vertical translation and the pitchwise orientation using three different angle ranges: $\pm 5°$, $\pm 10°$, and $\pm 20°$; and a spiral movement of the probe in the transverse plane combined with a steady approach towards the stationary drogue.

The first three sets of tests, in the X_{R1}, Y_{R1}, and Z_{R1} directions, are intended to identify the extent of motion in each direction that can be accommodated by the algorithm between two image frames before the accuracy of the tracking is degraded. In the fourth test, the changing inclination of the drogue is used to examine the response of the tracking system to a motion that violates the assumptions made in the development of the algorithms: that the motion of the drogue can be described by two translational parameters and a scale factor that represents the longitudinal translation. Finally, the spiral approach exhibits representative occlusions of the drogue by the probe that are likely to be seen in a real refuelling exercise.

For each of the previous situations, the motion was repeated three times, and the tests were performed off-line with the tracking algorithm running continuously throughout each test. Figure 4.33 presents examples of the tracking task performance for the different motion cases and the identified limits of the tracking algorithm.

In the case of changes in position, as can be seen in Figure 4.33, first row, the algorithm presented an unstable behavior (the red/dark box is shifted to the right) in some parts of the sequence when the speed changed from 400 to 2000 mm/s (when moving in the plane Y_{R1}, Z_{R1}). This instability was always detected in the same point of the sequence (drogue moving to the left in the image plane). Concerning this, we consider the reason of the instability to be more likely a change in the appearance of the drogue in those points than a change in the speed. Nonetheless, in spite of these minor instabilities, the algorithm tracked the drogue throughout the whole sequence when the changes in position were applied in the Y_{R1} and Z_{R1} axes, and also when significant changes in scale were applied at the three different speeds (in the longitudinal X_{R1} axis, the position changes from 0 m to 5 m).

When the wave movements were conducted (see Figure 4.33, second row), the performance of the algorithm was affected when the inclination of the drogue increased from 10° to 20°. Under this angle, the appearance of the drogue changes drastically, whereupon the exact position is not found. Nonetheless, it is important to highlight the good performance of the algorithm under these movements. These movements are not included in the 3 parameters motion model defined in Equation (3.14), but the algorithm is in any case able to tolerate them in some degree.

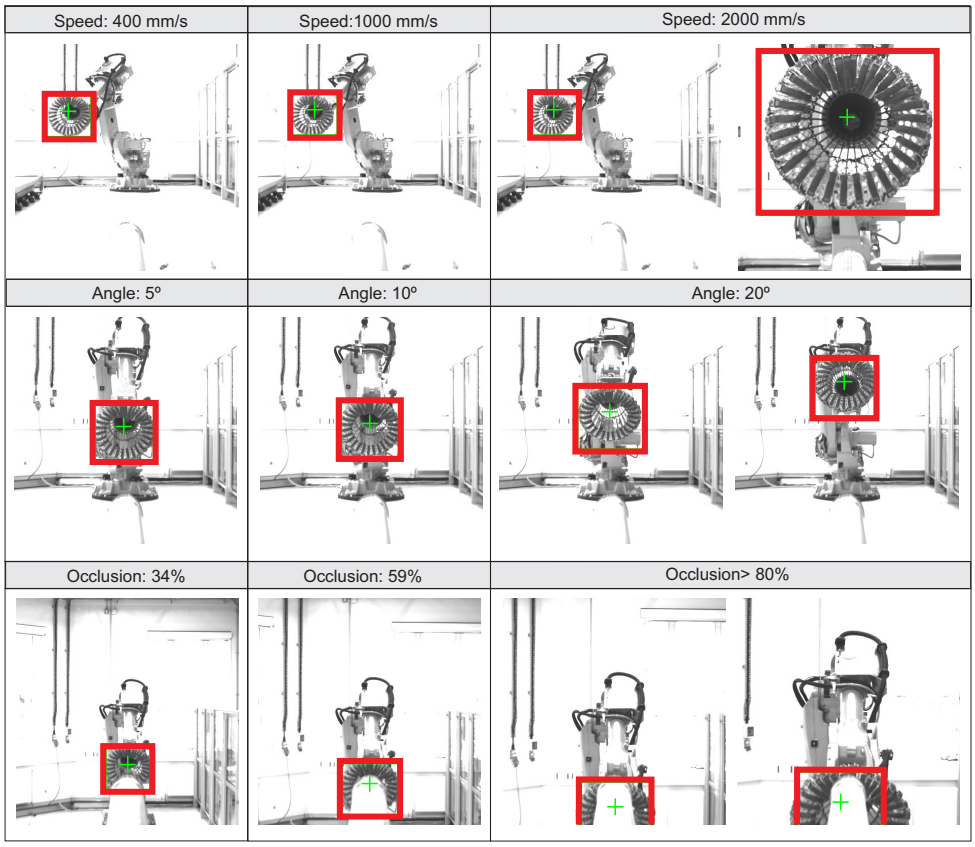


Figure 4.33: Limits analysis. The green/light crosshair, and red box/dark box indicate the tracking results. Different motions are analyzed: first, different 3D movements at different speeds (first row). The speed did not affect the performance of the algorithm. Second, wave movements with different angles (second row). With angles $> 10°$ the performance of the algorithm was affected. Third, spiral movements produced different type of occlusions (third row). When this percentage was $> 80\%$, the drogue was not tracked correctly.

Finally, Figure 4.33, third row, shows the results of the algorithm under spiral movements. As can be seen, one of the advantages of the proposed strategy based on direct methods is that by using the information of all the pixels that belong to the drogue, the algorithm is able to continue tracking the drogue under partial occlusions of it or when part of the drogue is out the FOV of the camera. Nonetheless, when the percentage of pixels occluded in $\mathbf{T}$ was greater than 80%, or in other words, when the percentage of pixels in $\mathbf{T}$ that are used in the minimization process of the tracking algorithm was less than 20%, the drogue was not tracked correctly (see Figure 4.33, third row). These percentages were calculated taking into account the number of pixels in $\mathbf{T}$ that were excluded in the minimization process. This includes the pixels occluded by the probe and the pixels outside of the FOV of the camera.

4.7.3. Aerial Refuelling Test Under Non-Turbulence Effects

In this test, simulated aircraft motion data is used. This data is sent to one of the robots (the one that simulates the receiver aircraft, shown in Figure 4.31), in order to

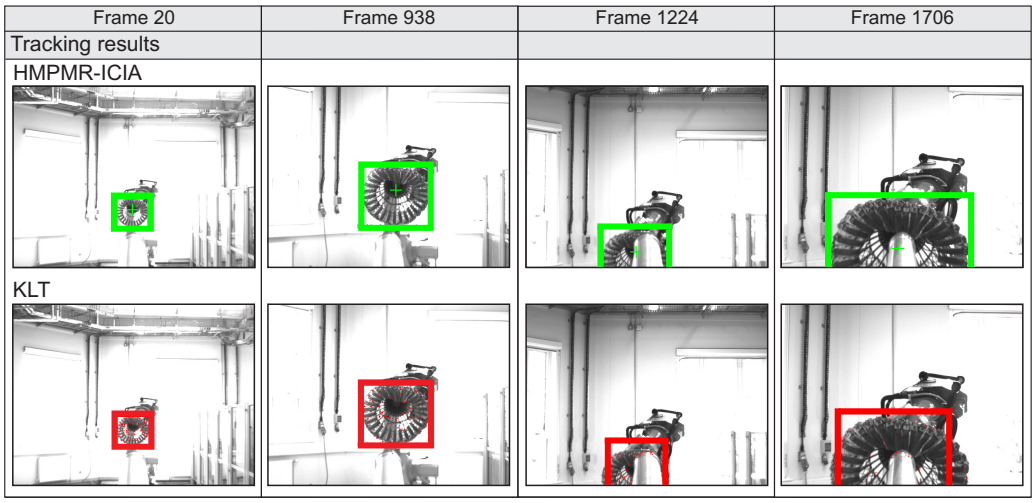


Figure 4.34: Tracking results during a refuelling test. The first column shows the results of the HMPMR-ICIA algorithm (green/light box), and the second column shows the results of the KLT algorithm (red/light box). The color boxes indicate the estimated position of the drogue.

recreate a refuelling procedure. The data correspond to motions of the probe towards the drogue (the drogue is kept stationary), simulating the aerial refuelling task (no turbulence effects are considered). Challenging conditions are present in the image sequence used in this test: the drogue goes out of the FOV, the appearance of the drogue changes in the image plane (due to roll and pitch effects), and there are situations in which the drogue is occluded by the probe.

In the test, the HMPMR-ICIA algorithm is compared with the KLT feature-based algorithm when tracking the drogue during the aerial refuelling pocedure. The evaluation of the performance of both algorithms is based on a visual examination of the tracking results: analyzing if the drogue is tracked in the sequence (the red/dark box must be covering the drogue), and also analyzing the motion model recovered by each algorithm.

Figure 4.34 presents representative images that illustrate the performance of the tracking algorithms. The first row shows the results of the HMPMR-ICIA algorithm during the refuelling task (green/light box), and the second row shows the results of the KLT algorithm (red/dark box).

As can be seen, the HMPMR-ICIA algorithm (first row) tracks the drogue during the entire refuelling task, despite occlusions of the drogue by the probe and the fact that part of the drogue is outside of the FOV of the camera (e.g. Frames 1224 and 1706). In Figure 4.34, second row, it can be seen that the KLT algorithm fails to find a good motion model for tracking the drogue, especially in the final part of the refuelling procedure (e.g. the red/dark box in Frames 1224 and 1706 does not cover the extent of the drogue).

On the other hand, Figure 4.35 shows a visual analysis of the motion models recovered by each algorithm. In each frame, the estimated motion model is used to transform the current frame to the coordinate frame of the first image (i.e. back-warping the current image), where the template image was initialized. If the motion model is correctly estimated, the template image and the back-warped template should be very similar (in size).

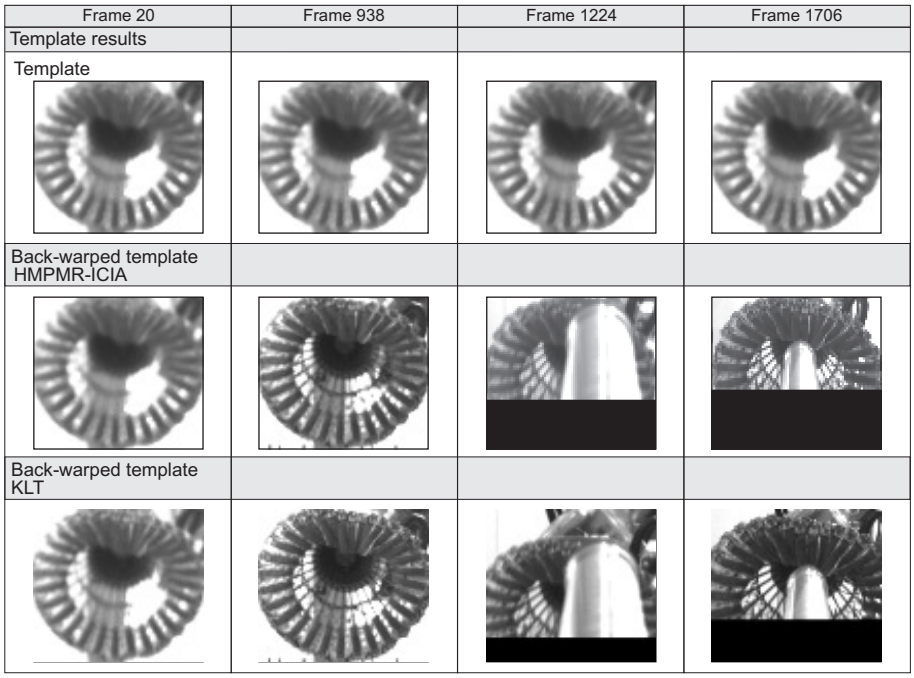


Figure 4.35: Visual analysis of the estimated motion. The first row shows the template image (i.e. the object to track). The motion estimated by each algorithm (the HMPMR-ICIA and the KLT) is used to transform the current frame to the coordinate frame of the template image (back-warp). The second and third rows show the back-warped templates.

In Figure 4.35, the first row shows the template image. The second row shows the back-warped template found using the motion model estimated by the HMPMR-ICIA algorithm; and the third row shows the back-warped template found using the motion model estimated by the KLT algorithm. In this figure, it can be seen that the transformation recovered by the HMPMR-ICIA (second row) allows to appropriately transform the current frame to the coordinate frame of the first image. Whereas in the third row, it can be seen that in some frames, e.g. in Frames 1224 and 1706, the KLT can not find a good motion model that permits to recover the correct position and scale of the drogue in the image plane in the final part of the refuelling test (the most critical one). As a consequence, as was also shown in Figure 4.34, the KLT fails to track the drogue.

4.7.4. Discussion

In this section, the HMPMR-ICIA algorithm has been adapted to solve the drogue tracking problem in aerial refuelling tasks (based on the probe and drogue method). As was shown in Chapter 2, the vast majority of approaches presented in the literature make use of feature-based algorithms to conduct the tracking task, requiring in some situations the placement of specific landmarks on-board. Nonetheless, by using direct methods, all the pixel information that belongs to the drogue is used in the motion estimation process. Because of this, the tracking task can be conducted when the center of the drogue is occluded (e.g. occlusions by the probe) or in situations where part of the drogue goes out of the FOV of the camera.

The HMPMR-ICIA algorithm for AAAR tasks has been tested in a a robotic laboratory facility using real visual information from real refuelling hardware (a drogue and a probe). In the tests that were conducted, basic motions of the drogue were used to show the advantages of including the masks **P** and **M** in the image registration algorithm. Additionallly, these motions were used to analyze the limits, in terms of speed and inclinations, that the algorithm can tolerate during the tracking task. On the other hand, motions of the probe during a refuelling approach using simulated aircraft motion data in non-turbulence mode were used.

In all these tests, the HMPMR-ICIA algorithm that was adapted to the drogue tracking task proved to be robust under adverse conditions, including rapid motions, large changes in proximity and scale, varying orientations of the drogue, and significant occlusions of the drogue by the probe. Additionally, despite the assumption of a simplified motion model (3 parameters), the algorithm tracked the drogue under high percentages of occlusion (this situation is very difficult to overcome with strategies that exclusively track the inner part of the drogue); and also under motions that are not modeled by the assumed motion model, for example inclinations of the drogue, because a range of orientations up to $\pm 10^{\circ}$ was tolerated, as was shown in Section 4.7.2.2. Results also demonstrate that, for the drogue tracking task, the implemented direct method obtains better results than the feature-based KLT algorithm. With the direct method, robust position estimations of the drogue at real-time frame rates (> 30 FPS) were obtained overcoming some adverse conditions, such as those present when the drogue is being occluded by the probe, or when part of the drogue goes out of the FOV of the camera.

4.8. Summary

In this chapter the Hierarchical Multi-Parametric and Multi-Resolution (HMPMR) strategy that improves the performance of a tracking algorithm based on direct methods, has been introduced. The different components of this strategy has been presented. The efficient Inverse Compositional Image Alignment algorithm (ICIA) has been extended with the HMPMR strategy, and the different steps of the HMPMR-ICIA algorithm have been explained.

An important aspect of the proposed strategy is the selection of the MP hierarchy. In this chapter, we have presented an analysis of the distribution of parameters in the MR structure and its influence in the tracking results. It has been found that the best distribution of parameters that allows to obtain stable results when tracking objects is the one that: at the lowest resolution level estimates the translation (2 parameters); at the highest resolution level estimates the best motion model that describes the motion of the object to track in the image plane with the lowest number of parameters (in order to obtain a fast response); and that in the intermediate levels estimates a progressive distribution of parameters in order to obtain a smooth increment of the number of parameters throughout the MR pyramid.

In the different tests that were conducted, the performance of the HMPMR strategy has been analyzed by comparing it with different state of the art algorithms, based both on features and direct methods, in two different applications: tracking objects on-board

UAVs and in Autonomous Air-to-Air Refuelling (AAAR). In both applications, image data obtained from: real flight tests, using a UAV; and from a laboratory testbed for AAAR tasks, using actual flight hardware (probe and drogue), was used.

In both applications, the requirements of direct methods, especially the small frame-to-frame motion constraint, are easily unsatisfied, due to factors as vehicle vibrations, problems caused by outdoors operations, or turbulence effects, among others. However, in this chapter it has been shown that by configuring the image registration algorithm with an MR and an MP hierarchy, the results of the tracking task have a more robust behavior than when none of the hierarchies or when only an MR hierarchy are used. Therefore, with the HMPMR-ICIA algorithm, it is possible to obtain robust estimations under the presence of large frame-to-frame motions, estimating both simple and complex motion models.

Previous works in the tracking area have often been based on feature methods. Nonetheless, it has been shown that the tracking strategy performs better than well-known feature-based algorithms (SIFT and KLT) and well-known configurations of direct methods (MR-ICIA) in the presence of strong changes in position, fast changes in appearance, situations where part of the template is out of the FOV of the camera, and under constant vibrations. Concerning the latter aspect, this is accomplished without requiring any specific hardware or software for video stabilization. Therefore, by using the ICIA algorithm, an efficient tracking algorithm can be achieved using direct methods; and by integrating it with the HMPMR structure, robust motion estimations can be achieved, allowing to track objects during long periods of time at real-time frame rates, and to estimate motion models with large and low numbers of parameters.

Due to the amount of information that direct methods have to evaluate, these kinds of methods are not commonly used for real-time applications. Nonetheless, we have shown that by using the proposed strategy, direct methods can be employed for real-time tracking, being able to achieve frequencies of 16 FPS when estimating 8 parameters. It is important to notice that the speed is highly dependent on the number of parameters estimated (faster responses, 30 and 50 FPS, are achieved when estimating motion models with a lower number of parameters). Additionally, the speed of the algorithm is dependent on the size of the template and the number of parameters estimated in each level of the pyramid.

Limitations of the tracking algorithm can be related to the characteristics of the object to track, taking into account that direct methods require texture information to align the images. Another limitation can be related to the size of the template, that when exceedingly large can make the algorithm run slowly, since direct methods make a pixel by pixel search of the parameters, and this search will be repeated in each level of the HMPMR structure. Nonetheless, in Section 4.5 it has been shown that when the template images are big and when the required number of pyramid levels is also high, by using some of the pixels of the template in the higher resolution levels robust estimations at real-time frame rates can still be obtained. It is also important to notice that depending on the application, speeds can be improved, e.g. a faster algorithm can be obtained by using lower image sizes or template sizes, and by adopting a dynamic strategy in terms of the number of pixels that take part in the minimization process in each level of the pyramid.

Videos of the different tests described in this section can be found in [52].

Chapter 5

Monocular On-Board Pose Estimation

5.1. Introduction

In this chapter[1], we present two pose estimation algorithms based on the image information captured by a camera on-board a UAV: one for estimating the global pose of the UAV (a visual odometry algorithm), and a second one for estimating relative positions with respect to an object of interest. In both cases, we address the pose estimation problem based on direct methods instead of on feature-based methods, using image registration and homography decomposition techniques, and assuming that the object that is imaged by the camera is planar or can be assumed to be planar (e.g. when flying at high altitudes). The homographies required by the pose estimation algorithms are obtained with the HMPMR-ICIA algorithm, presented in Section 4.3, which has been adapted to the requirements of each pose estimation technique.

The first pose estimation strategy that is presented is based on the decomposition of the frame-to-frame homography induced by the ground plane in order to obtain the motion of the camera. The second strategy estimates the relative position of the camera

[1]Publications of the author related to this chapter:

- A Multi-Resolution Image Alignment Technique Based on Direct Methods for Pose Estimation of Aerial Vehicles [122]
- A Vision-Based Strategy for Autonomous Aerial Refueling Tasks [126]

with respect to an object of interest, based on the decomposition of the homography that relates points on a plane in the 3D space with their projection in the image plane. These strategies have been used for solving the pose estimation problem in two different scenarios: for state estimation of aerial vehicles (global and relative estimations), and for autonomous air-to-air refuelling.

5.2. Pose Estimation Based on the Decomposition of Homography $\mathbf{H_p}$

If the object that is being imaged by the camera is planar or can be assumed to be planar, the camera motion (i.e. the frame-to-frame motion) can be extracted from the homography induced by that plane. In Figure 5.1, it can be seen that points that lie on the image plane of one view (${}^{f_1}\mathbf{x}$, Figure 5.1) are related to the corresponding image points in a second view (${}^{f_2}\mathbf{x}$, Figure 5.1) by a planar homography $\mathbf{H_p}$ [81].

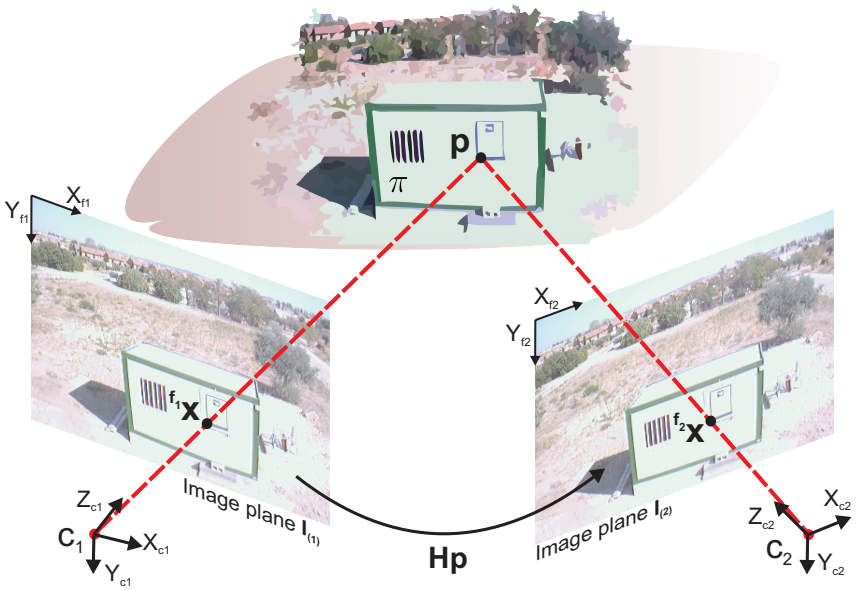


Figure 5.1: The homography induced by a plane. The homography induced by the plane maps points in image 1 to points in image 2.

When two images, as the ones shown in Figure 5.1 ($\mathbf{I}_{(0)}$ and $\mathbf{I}_{(1)}$) of the same planar object are captured at two different instants of time and the camera is moving, their corresponding camera coordinate systems are related by a rigid transformation, as follows:

$$ {}^{c_1}\mathbf{x} = {}^{c_1}\mathbf{R}_{c_0}\, {}^{c_0}\mathbf{x} + {}^{c_1}\mathbf{t}_{c_0} \tag{5.1} $$

Where ${}^{c_0}\mathbf{x}$, and ${}^{c_1}\mathbf{x}$ are the coordinates of a 3D point $\mathbf{p}$, located on the plane π, relative to each camera frame; and ${}^{c_1}\mathbf{R}_{c_0}$ and ${}^{c_1}\mathbf{t}_{c_0}$ are the rotation matrix and the translation vector, respectively.

If the ground plane π is characterized by the unit normal vector ${}^{c_0}\mathbf{n}$ with respect to the c_0 camera reference frame, and by its distance ${}^{c_0}d$ from the plane π to the origin of

the reference frame c_0 (i.e. to the optical center of the camera), as follows:

$$\begin{aligned} {}^{c_0}\mathbf{n}^{T}\,{}^{c_0}\mathbf{x} = n_1 x + n_2 y + n_3 z = {}^{c_0}d \\ \frac{1}{{}^{c_0}d}\,{}^{c_0}\mathbf{n}^{T}\,{}^{c_0}\mathbf{x} = 1 \end{aligned} \tag{5.2}$$

then, substituting Equation (5.2) into Equation (5.1), gives:

$$\begin{aligned} {}^{c_1}\mathbf{x} &= {}^{c_1}\mathbf{R}_{c_0}\,{}^{c_0}\mathbf{x} + {}^{c_1}\mathbf{t}_{c_0}\,\frac{1}{{}^{c_0}d}\,{}^{c_0}\mathbf{n}^{T}\,{}^{c_0}\mathbf{x} \\ {}^{c_1}\mathbf{x} &= \left({}^{c_1}\mathbf{R}_{c_0} + \frac{1}{{}^{c_0}d}\,{}^{c_1}\mathbf{t}_{c_0}\,{}^{c_0}\mathbf{n}^{T}\right){}^{c_0}\mathbf{x} \end{aligned} \tag{5.3}$$

Where

$$\mathbf{H_e} = {}^{c_1}\mathbf{R}_{c_0} + \frac{1}{{}^{c_0}d}\,{}^{c_1}\mathbf{t}_{c_0}\,{}^{c_0}\mathbf{n}^{T} \tag{5.4}$$

is called in [116] as the euclidean homography matrix that denotes a linear transformation from ${}^{c_0}\mathbf{x} \in \mathbb{R}^3$ to ${}^{c_1}\mathbf{x} \in \mathbb{R}^3$ (It transforms one 3D point in projective coordinates from one frame to the other, again up to a scale factor) [116], as follows:

$${}^{c_1}\mathbf{x} = \mathbf{H_e} = {}^{c_0}\mathbf{x} \tag{5.5}$$

This matrix depends on the motion parameters $({}^{c_1}\mathbf{R}_{c_0},\, {}^{c_1}\mathbf{t}_{c_0})$ as well as on the structure parameters $({}^{c_0}\mathbf{n}^{T},\, {}^{c_0}d)$ of the plane π. Considering the scale ambiguity in the term $\frac{{}^{c_0}\mathbf{t}_{c_1}}{{}^{c_0}d}$, it is expected to recover from $\mathbf{H_e}$ the ratio of the translation scaled by the distance ${}^{c_0}d$.

On the other hand, if we assume that the camera calibration matrix is constant (i.e. that from frame-to-frame the focal length does not change), and if the pinhole camera model is considered (Appendix A), Equation (5.3) can be expressed in terms of image coordinates, as follows:

$${}^{f_1}\mathbf{x} = \gamma\mathbf{K}\,\left({}^{c_1}\mathbf{R}_{c_0} + \frac{1}{{}^{c_0}d}\,{}^{c_1}\mathbf{t}_{c_0}\,{}^{c_0}\mathbf{n}^{T}\right)\,\mathbf{K}^{-1}\,{}^{f_0}\mathbf{x} \tag{5.6}$$

Where $\gamma = \frac{{}^{c_1}z}{{}^{c_0}z}$, $\mathbf{K}$ is the calibration matrix, and

$$\mathbf{H_p} = \gamma\mathbf{K}({}^{c_1}\mathbf{R}_{c_0} + \frac{1}{{}^{c_0}d}\,{}^{c_1}\mathbf{t}_{c_0}\,{}^{c_0}\mathbf{n}^{T})\mathbf{K}^{-1} \tag{5.7}$$

Equation (5.7) is also a homography (called in [116] as the projective homography $\mathbf{H_p}$), and allows to map points from image $\mathbf{I}_{(0)}$ to points in image $\mathbf{I}_{(1)}$. From Equation (5.7), the euclidean homography $\mathbf{H_e}$ can be calculated as shown in Equation (5.8), using the intrinsic camera parameters $\mathbf{K}$, and recovering the scale factor as $\gamma = \text{med}(\text{SVD}(\mathbf{H_L}))$ [116, 120], where $\mathbf{H_L}$ is:

$$\begin{aligned} \mathbf{H_L} &= \mathbf{K}^{-1}\mathbf{H}\mathbf{p}\,\mathbf{K} \\ \mathbf{H_e} &= \frac{\mathbf{H_L}}{\gamma} = ({}^{c_1}\mathbf{R}_{c_0} + \frac{1}{{}^{c_0}d}\,{}^{c_1}\mathbf{t}_{c_0}\,{}^{c_0}\mathbf{n}^{T}) \end{aligned} \tag{5.8}$$

These two homographies, $\mathbf{H_e}$ and $\mathbf{H_p}$, are essential for recovering the camera motion based on visual information.

The visual tracking algorithm provides the pose estimation algorithm with a homography $\mathbf{H_p}$, as the one shown in Equation (5.7), that maps points in one image to points in another image. Based on $\mathbf{H_p}$, the homography $\mathbf{H_e}$ must be estimated in order to recover, from this matrix, the motion of the camera (${}^{c_1}\mathbf{R}_{c_0}$, ${}^{c_1}\mathbf{t}_{c_0}$) and the structure parameters ($\frac{1}{{}^{c_0}d}$, ${}^{c_0}\mathbf{n}^{\mathrm{T}}$). The process of extracting these parameters is known as homography decomposition.

In the following sections, the visual tracking algorithm adapted to the requirements of the pose estimation technique is presented, and the pose estimation algorithm for estimating the state of aerial vehicles is derived.

5.2.1. Visual Tracking

As mentioned previously, the visual tracking algorithm provides the pose estimation algorithm with a homography $\mathbf{H_p}$ for each pair of images, which maps points in one image to points in another image. In the tracking algorithm presented in Section 4.3, the visual tracking task consisted in estimating the location of an object of interest in the image plane by propagating the motion model from frame-to-frame. Nonetheless, in this section we examine the use of the appearance of all the pixels in the image to directly estimate the frame-to-frame motion. Therefore, the HMPMR-ICIA algorithm presented in Algorithm 5 is modified to not track objects but to register pairs of images, as outlined in Algorithm 6. The main differences with Algorithm 5 for tracking, are:

- The initialization stage is conducted every time a new frame is analyzed. This is because the template image $\mathbf{T}$ changes with each new frame.
- Step 8 in Algorithm 5 is not conducted, because the motion model is not propagated to the next frames.
- At the end of the registration stage, the template image is initialized based on the current image (i.e. the template image changes with each new frame).

Figure 5.2 shows the HMPMR-ICIA used to register pairs of images in order to estimate the frame-to-frame homography $\mathbf{H_p}$. As can be seen in the figure, the parameters of the motion model are not propagated to the other frames, and the current frame is always registered with the previous one using all the pixels contained in a predefined subregion that covers 80% of the image (the same area is used in all the frames).

input : Image $\mathbf{I}_{(0)}$; $F = 1$
foreach *new image* $\mathbf{I}_{(F)}$ **do**
  Initialization stage
  1. **Initialize** $\mathbf{T}_{(F-1)}$ from $\mathbf{I}_{(F-1)}$
  2. **Compute** steps 1-6 Algorithm 5, for image $\mathbf{T}_{(F-1)}$

  Registration stage
  3. **Compute** step 7 Algorithm 5, for image $\mathbf{I}_{(F)}$
  4. **Compute** steps 9-14 Algorithm 5, for images $\mathbf{I}_{(F)}$ and $\mathbf{T}_{(F-1)}$ in order to estimate $\mathbf{W}$, i.e. the homography between $\mathbf{I}_{(F)}$ and $\mathbf{T}_{(F-1)}$
  5. **Show** results
  6. **Initialize** $\mathbf{T}_{(F)}$ from $\mathbf{I}_{(F)}$
  7. **Compute** $F = F + 1$
end

Algorithm 6: Outline of the HMPMR-ICIA for image registration.

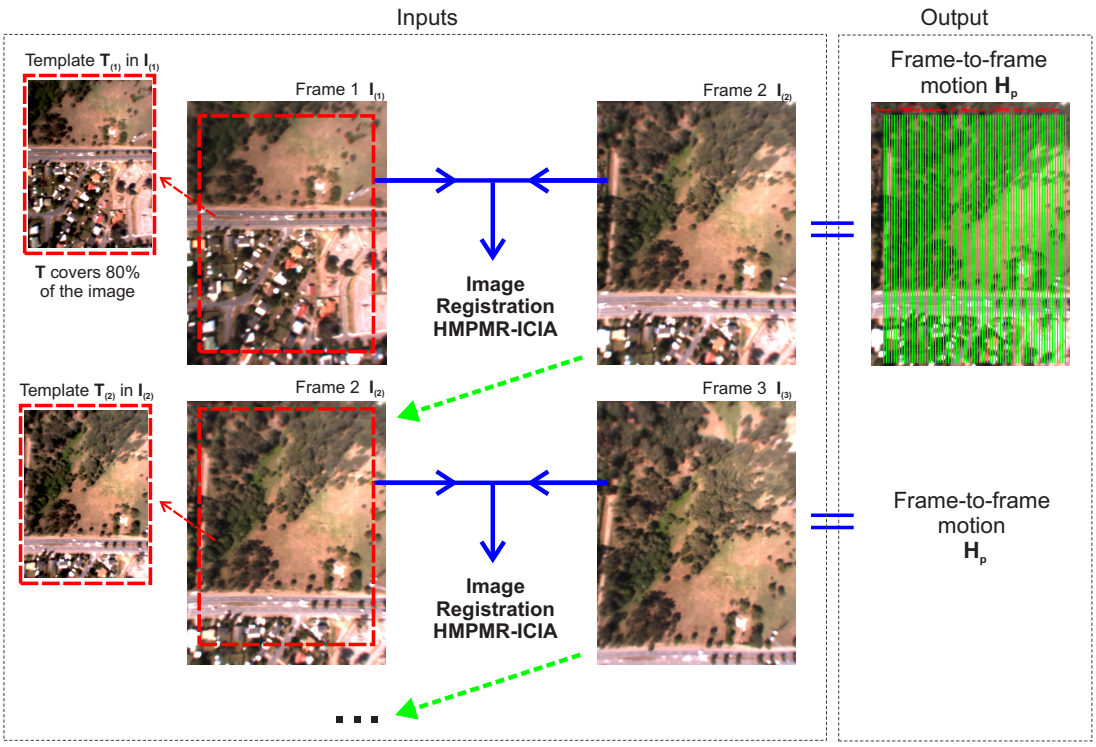


Figure 5.2: HMPMR-ICIA used to register pair of images.

5.2.2. Pose Estimation

This section presents the proposed algorithm for estimating the state of aerial vehicles. In general terms, the motion between frames (rotation and translation) is recovered by decomposing the frame-to-frame homography ($\mathbf{H_p}$), obtained with the previously explained registration algorithm which is applied to a patch that covers around 80% of the image. When the visual state estimation is required (e.g. during a GPS drop-out), this motion

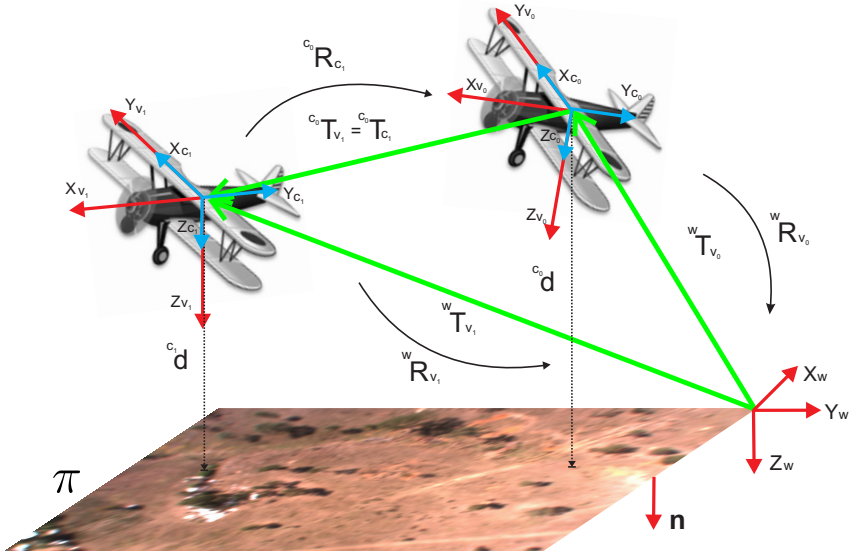


Figure 5.3: Pose estimation strategy for aerial vehicles, based on $\mathbf{H_p}$ decomposition.

is integrated with the previously known estimation of the vehicle's state, obtained from the on-board sensors (GPS/IMU); and the subsequent estimations are based only on the vision-based motion estimations.

The pose estimation strategy assumes that a camera, in a downwards-looking configuration, is located on-board an aerial vehicle, and that the position of its coordinate system $\mathrm{X_c}$ coincides with the vehicle's body frame $\mathrm{X_v}$, as shown in Figure 5.3. Thus, the transformation from $\mathrm{X_c}$ to $\mathrm{X_v}$ is defined by a fixed rotation ${}^{\mathrm{v}}\mathbf{R}_{\mathrm{c}}$ of 90° around the Z axis. Therefore, the motion of the UAV can be inferred from the estimated camera motion. The proposed strategy also assumes that when the algorithm starts operating, an initial estimation of the position ${}^{\mathrm{w}}\mathbf{t}_{\mathrm{v}_0}$ and the orientation of the vehicle ${}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}_0}$ with respect to a world coordinate system are known (e.g. GPS/IMU estimation).

Therefore, the inter-frame motion is estimated as follows: from the matrix $\mathbf{H_p}$ estimated in each frame, the $\mathbf{H_e}$ homography is obtained (see Equation (5.8)). From this matrix, the motion and structure parameters $({}^{\mathrm{c}_1}\mathbf{R}_{\mathrm{c}_0}, {}^{\mathrm{c}_1}\mathbf{t}_{\mathrm{c}_0}, \frac{1}{{}^{\mathrm{c}_0}d}, {}^{\mathrm{c}_0}\mathbf{n}^{\mathrm{T}})$ are recovered using the method described in [116]. This method provides four possible solutions for the decomposition of $\mathbf{H_e}$. The correct solution is chosen, assuming the positive depth constraint (that the points are in front of the camera.), and also assuming that the ground plane is not sloped (i.e. $\mathbf{n} = [0, 0, 1]^{\mathrm{T}}$).

With the previously explained method, the rotation matrix ${}^{\mathrm{c}_0}\mathbf{R}_{\mathrm{c}_1}$ and the scaled translation vector $\frac{{}^{\mathrm{c}_0}\mathbf{t}_{\mathrm{c}_1}}{{}^{\mathrm{c}_0}d}$ are recovered. The absolute translation ${}^{\mathrm{c}_0}\mathbf{t}_{\mathrm{c}_1}$ can be recovered using a measured or calculated height [100]. With the knowledge of all this information, the pose with respect to the world coordinate system is found as follows.

The absolute rotation ${}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}_1}$ is found as:

$$ {}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}_1} = {}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}_0}\, {}^{\mathrm{v}_0}\mathbf{R}_{\mathrm{c}_0}\, {}^{\mathrm{c}_0}\mathbf{R}_{\mathrm{c}_1}\, {}^{\mathrm{c}_1}\mathbf{R}_{\mathrm{v}_1} \tag{5.9} $$

where, ${}^{\mathrm{v}_0}\mathbf{R}_{\mathrm{c}_0} = {}^{\mathrm{v}}\mathbf{R}_{\mathrm{c}}$ and ${}^{\mathrm{c}_1}\mathbf{R}_{\mathrm{v}_1} = {}^{\mathrm{v}}\mathbf{R}_{\mathrm{c}}^{\mathrm{T}}$. From ${}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}_1}$, the Euler roll ($\phi$), pitch ($\theta$), and

yaw (ψ) angles, can be obtained as follows. If $\mathbf{R} = {}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}_1}$, then:

$$\begin{aligned} \phi &= \mathrm{atan2}(r_{21}, r_{11}) \\ \theta &= \mathrm{atan2}(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \psi &= \mathrm{atan2}(r_{32}, r_{33}) \end{aligned} \tag{5.10}$$

The translation vector ${}^{\mathrm{w}}\mathbf{t}_{\mathrm{v}_1} = {}^{\mathrm{w}}\mathbf{t}_{\mathrm{c}_1}$ is recovered as:

$${}^{\mathrm{w}}\mathbf{t}_{\mathrm{v}_1} = {}^{\mathrm{w}}\mathbf{R}_{\mathrm{c}_0}\, {}^{\mathrm{c}_0}\mathbf{t}_{\mathrm{c}_1} + {}^{\mathrm{w}}\mathbf{t}_{\mathrm{c}_0} \tag{5.11}$$

where ${}^{\mathrm{w}}\mathbf{R}_{\mathrm{c}_0} = {}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}_0}\, {}^{\mathrm{v}_0}\mathbf{R}_{\mathrm{c}_0}$.

When a new image is analyzed. If the information on the distance to the plane is not available from an on-board sensor (e.g. altimeter), this distance can be calculated from previously known data, as follows:

$${}^{\mathrm{c}_1}d = {}^{\mathrm{c}_0}d + {}^{\mathrm{c}_0}\mathbf{t}_{\mathrm{c}_1} \cdot {}^{\mathrm{c}_0}\mathbf{n} \tag{5.12}$$

This distance is then used to recover the absolute translation from Frame 1 to Frame 2. The process is repeated after this, and the vision-based pose of the vehicle is propagated.

5.3. Relative Pose Estimation Based on the Decomposition of Homography $\mathbf{H_w}$

In this section, a pose estimation algorithm widely used in the literature [172, 30, 142] is presented. The method is based on the homography $\mathbf{H_w}$, that relates points on the world plane to points on the image plane [172], as shown in Figure 5.4. In this pose estimation strategy, we assume that the visual tracking algorithm is capable of obtaining the ROI (region of interest) that defines where an object of interest is located (a planar object or one that can be assumed to be planar). Therefore, the pose estimation algorithm uses that ROI in order to calculate $\mathbf{H_w}$, and then, from this homography, the algorithm extracts the position and orientation between the world and the camera coordinate systems.

Using the pinhole camera model [81] (Appendix A), a 3D point located on the world plane π is projected onto the image plane via the 3×4 projection matrix $\mathbf{P}$ [81], as follows:

$$\begin{aligned} {}^{\mathrm{f}}\mathbf{x} &= \mathbf{P}\, {}^{\mathrm{w}}\mathbf{x} \\ {}^{\mathrm{f}}\mathbf{x} &= \lambda \mathbf{K}[\mathbf{R} \mid \mathbf{t}]{}^{\mathrm{w}}\mathbf{x} \end{aligned} \tag{5.13}$$

Where ${}^{\mathrm{f}}\mathbf{x} = (x, y, 1)$ are the 2D image coordinates of a point, ${}^{\mathrm{w}}\mathbf{x} = (x_{\mathrm{w}}, y_{\mathrm{w}}, z_{\mathrm{w}}, 1)$ represents the 3D world coordinates of the same point, λ is a scale factor, $\mathbf{R}$ and $\mathbf{t}$ are the orientation and position of the world reference frame in the camera coordinate system (as shown in Figure 5.4), and $\mathbf{K}$ is the camera calibration matrix, which is found by an off-line calibration process [210] (e.g. using the camera calibration toolbox for Matlab [23]). If it is assumed that the object of interest is planar (or can be assumed to be planar), then the 3D coordinates of a point located on the object of interest have the

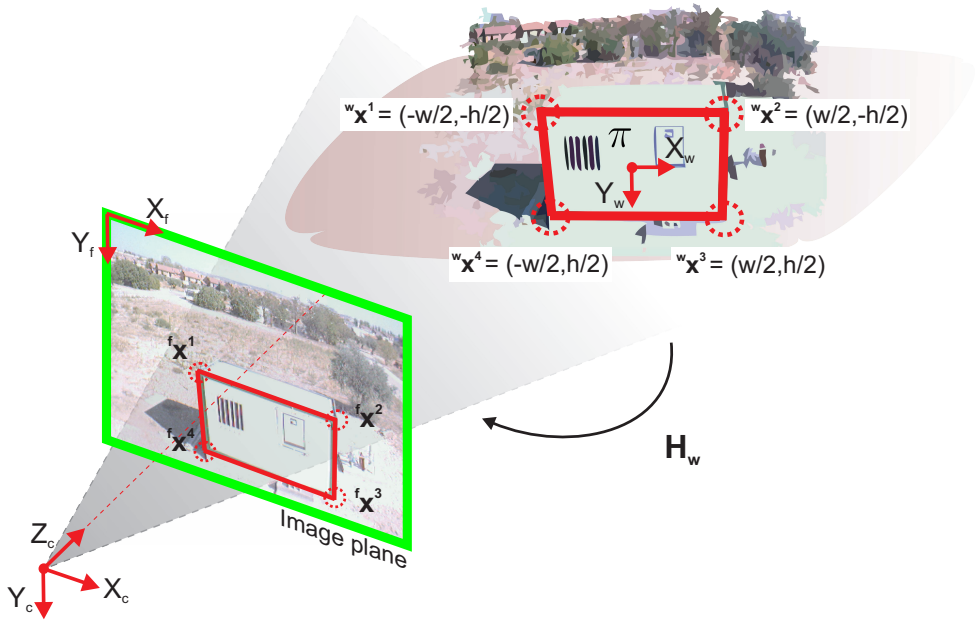


Figure 5.4: Pose estimation based on $\mathbf{H_w}$ decomposition. Assuming that the camera is calibrated and that the dimensions of a planar object are known, points on the world plane and points on the image plane can be used to calculate $\mathbf{H_w}$. From this homography, the pose between the world and the camera coordinate systems can be obtained.

form $^{w}\mathbf{x} = (^{w}x, ^{w}y, 0, 1)$. Therefore, Equation (5.13) is simplified for the planar case, as follows:

$$\begin{aligned} ^{f}\mathbf{x} &= \lambda \mathbf{K}[\mathbf{r_1}\ \mathbf{r_2} \mid \mathbf{t}]^{w}\mathbf{x} \\ ^{f}\mathbf{x} &= \mathbf{H_w}\,^{w}\mathbf{x} \end{aligned} \tag{5.14}$$

Where $\mathbf{H_w}$ is the planar homography that transforms points in the world plane into points in the image plane, as shown in Figure 5.4. This homography is a 3×3 matrix, but only 8 of its 9 elements are significant (scale does not affect the equation) [51].

5.3.1. Pose Estimation

The pose estimation strategy estimates the relative position and orientation between the camera frame and the world frame located on the 3D plane π, shown in Figure 5.4. The algorithm is based on decomposing the $\mathbf{H_w}$ homography derived previously in Equation (5.14). It also assumes that a tracking algorithm is in charge of estimating, in each frame, the 2D position of a planar object whose dimensions in the 3D space are known.

In order to estimate $\mathbf{H}_w$ in each frame, the known dimensions of the planar object are used to define the 3D plane π where the world coordinate system is located. As can be seen in Figure 5.4, with the dimensions of the object (width w and height h), the position of four 3D points that lie on plane π can be defined (see Figure 5.4, $^{w}\mathbf{x}^1 \ldots {}^{w}\mathbf{x}^4$). On the other hand, assuming that the tracking algorithm robustly estimates, in each frame, the 2D position of the planar object, then from these positions four 2D points can be defined in the image plane: $^{f}\mathbf{x}^1 \ldots {}^{f}\mathbf{x}^4$, which correspond to the projection of the four 3D points.

Afterwards, the four 3D points and the four 2D points are used to estimate $\mathbf{H}_{\mathrm{w}}$, as follows:

$$\begin{pmatrix} {}^{\mathrm{f}}x^i \\ {}^{\mathrm{f}}y^i \\ 1 \end{pmatrix} = \mathbf{H}_{\mathrm{w}} \begin{pmatrix} {}^{\mathrm{w}}x^i \\ {}^{\mathrm{w}}y^i \\ 1 \end{pmatrix} \tag{5.15}$$

where ${}^{\mathrm{w}}\mathbf{x}^i$ corresponds to the coordinates of the four 3D points that lie on plane π, ${}^{\mathrm{f}}\mathbf{x}^i$ corresponds to the coordinates of the four 2D points in the image plane, and the index i represents each corner ($i = \{1, 2, 3, 4\}$).

Therefore, with the point-to-point 2D-3D correspondence of the four corners of the planar object, and reorganizing Equation (5.15), a system of equations of the form $\mathbf{A}\mathbf{h}_{\mathbf{w}} = \mathbf{b}$ can be created in order to estimate $\mathbf{H}_{\mathbf{w}}$, where $\mathbf{h}_{\mathbf{w}}$ corresponds to the components of $\mathbf{H}_{\mathbf{w}}$ stacked into a vector. Once $\mathbf{H}_{\mathbf{w}}$ is estimated, the rotation matrix and the translation vector are estimated using the method described in [211], assuming that the camera parameters are known.

The translation vector ${}^{\mathrm{c}}\mathbf{t}_{\mathrm{w}}$ is found based on Equation (5.15) and taking into account that $\|\mathbf{r_1}\| = \|\mathbf{r_2}\| = 1$, as follows:

$$\begin{aligned} \mathbf{H}_{\mathbf{w}} &= [\mathbf{h_{w1}}\ \mathbf{h_{w2}}\ \mathbf{h_{w3}}] = \lambda \mathbf{K}[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}] \\ \lambda &= \|\mathbf{K}^{-1}\mathbf{h_{w1}}\| = \|\mathbf{K}^{-1}\mathbf{h_{w2}}\| \\ {}^{\mathrm{c}}\mathbf{t}_{\mathrm{w}} &= \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{h_{w3}} \end{aligned} \tag{5.16}$$

and the rotation matrix is estimated as follows:

$$\begin{aligned} \mathbf{r}_1 &= \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{h_{w1}} \\ \mathbf{r}_2 &= \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{h_{w2}} \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ {}^{\mathrm{c}}\mathbf{R}_{\mathrm{w}} &= \mathbf{R} = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3] \end{aligned} \tag{5.17}$$

5.4. Results: State Estimation of Aerial Vehicles

The pose estimation algorithm that is based on the decomposition of $\mathbf{H}_{\mathbf{p}}$ (see Section 5.3.1) is tested in this section. Different tests have been conducted to analyze the performance of the algorithm. The analysis is done in three different stages of a flight using real flight and image data: take-off, cruise, and landing, being two of those stages considered critical: take-off and landing.

The performance of the pose estimation strategy is analyzed by comparing it with the GPS/IMU estimations. The evaluation of the results is presented in terms of the RMSE (Root Mean Square Error) between the estimations. On the other hand, an analysis of the final frame rate of the algorithm is presented.

As depicted in Figure 5.3, an on-board camera in a downwards-looking configuration, and the assumption of planar scenes, are the bases of the algorithm. The HMPMR-ICIA

algorithm is applied to a patch that covers around 80% of the image and is used to estimate the frame-to-frame homography $\mathbf{H_p}$, which is required by the pose estimation algorithm. The motion between frames (rotation and translation) is recovered by decomposing this homography.

When the visual state estimation is required (e.g. during a GPS drop-out), the estimated frame-to-frame motion is integrated with the previously known estimation of the vehicle's state, obtained from other on-board sensors (GPS/IMU); and the subsequent estimations are based only on vision-based estimations.

As was shown in Section 2.2, most of the work presented in the literature makes use of features to determine the frame-to-frame homography matrix $\mathbf{H_p}$. Nonetheless, in this section the inter frame motion is estimated based on direct methods (using the HMPMR-ICIA algorithm). Therefore, in order to deal with the high computational cost of the proposed approach (because it uses 80% of the pixels of the image), we vary the number of pixels that intervene in the motion estimation process in each MR level.

5.4.1. Experimental Setup

The data used in the experiments correspond to a flight of more than 90 minutes. The image data was collected by the Airbone System Laboratory (ASL) [75]. This laboratory consists of a Cessna 172, as shown in Figure 5.5, owned and operated by the Australian Research Centre of Aerospace Automation (ARCAA).

Figure 5.5: The Airbone System Laboratory (ASL). This laboratory consists of a Cessna 172 equipped with an x86 computer running Linux; a NovAtel SPAN, which computes a tightly-coupled GPS/INS solution for position and attitude data; two cameras: one pointing forwards and the other one pointing downwards; and custom electronics for data synchronization. The ASL is capable of capturing images at a rate of 30 HZ with a resolution of 1024×768.

The system is equipped with an x86 computer running Linux; a NovAtel SPAN, which computes a tightly-coupled GPS/INS solution from measurements taken by a NovAtel OEMV-3 GNSS receiver and an iMAR-FSAS IMU for position and attitude data; two *Point Gray Flea* cameras mounted on a bracket: one pointing forwards and the other one pointing downwards (see Figure 5.5); and custom electronics for data synchronization.

The ASL is capable of capturing images from the on-board cameras at a rate of 30 HZ with a resolution of 1024×768 using the open-source Videography software package [196]. The data used in the tests is recorded from an on-board camera in a downwards-looking configuration. This camera is externally triggered, allowing the captured images to be precisely timestamped. The position and attitude data used as ground truth is obtained by an on-board GPS/INS system.

The data used in the experiments corresponds to a flight of more than 90 minutes, conducted around South-East Queensland, Australia. The flight path was chosen in order to maximize the terrain variability (flying over mountains, flat terrains, and the sea). From the available data, a collection of images that correspond to three stages of the flight was selected, these stages being: take-off, cruise, and landing. Figure 5.6 shows some of the images selected for the tests. As can be seen in it, the strong variability of the terrain, and the different conditions of the flight (different heights) represent a big challenge to the visual algorithm.

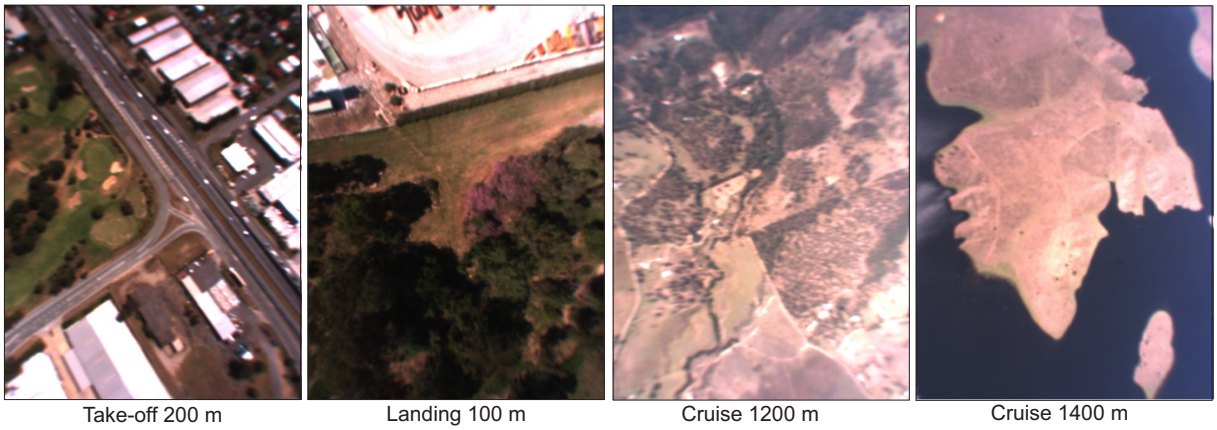


Figure 5.6: Image data. A collection of images that correspond to three stages of the flight were selected for the tests: take-off, cruise, and landing. As can be seen, the variability of the data (mountains, flat terrains, sea) and the different heights play an important role for testing the proposed strategy, and represent a big challenge for the visual algorithm.

5.4.2. Take-off, Cruise, and Landing Tests

5.4.2.1. Test 1: Take-Off

In this test, images from a take-off maneuver are used to analyze the performance of the pose estimation algorithm in this critical stage. Figure 5.7 presents some of the images used in the test. The HMPMR-ICIA algorithm for registering pairs of images,

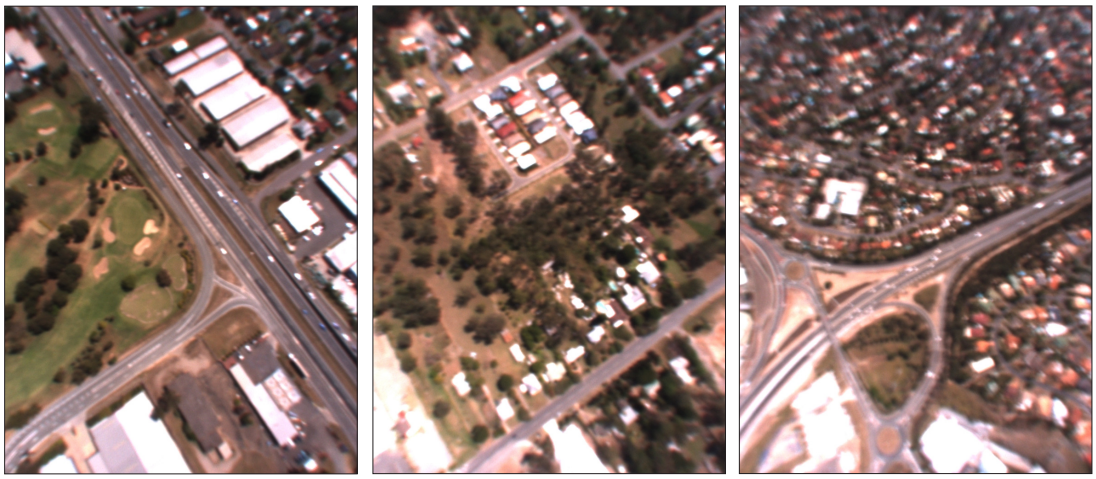


Figure 5.7: Take-off test images. From the available data set (a 90 minutes flight around South-East Queensland, Australia), a collection of images during take-off was selected.

explained in Section 5.2.1, Algorithm 6, has been configured to estimate the frame-to-frame homography required to calculate the inter-frame motion. It has been configured to have the following combination of parameters: 8-4-3-2, the homography in the lowest level of the pyramid (8 parameters), the translation in the highest level (2 parameters), and the similarity (4 parameters) and rotation+translation (3 parameters) in the intermediate levels.

Additionally, because 80% percent of the image is used to estimate the motion in each level of the multi-resolution pyramid, it is necessary to use a strategy to alleviate this high computational cost in order to not compromise the real-time operation of the algorithm. To deal with this, in each level of the hierarchical structure of the HMPMR-ICIA algorithm the number of pixels that are considered in the minimization process vary according to the resolution, as follows: in the highest level (lowest resolution) all the pixels are used; however, in the lowest level (highest resolution), 1 of every 5 pixels is employed. With this criteria, real-time frame rates are achieved without compromising the accuracy of the estimation.

In Figure 5.8, the results obtained with the proposed algorithm during the take-off stage are shown. In this figure, the visual estimation (red/dark line) is compared with the GPS/IMU data (green/light line) with the intention of comparing the behavior of the estimated data. Errors are also calculated and presented. As can be seen in Figure 5.8, the recovered positions and orientations present a behavior that is similar to the GPS/IMU data. The RMSEs in position are [152.69 m, 273.93 m, 50.6591 m] for the X, Y, and Z axes, respectively; and the RMSEs in orientation are $[3.72°, 1.17°, 1.58°]$ for roll, pitch, and yaw angles, respectively. These errors are relatively low, considering that the tests correspond to a manual flight of approximately 1 minute in length with a mean speed of $\approx$ 211 km/hr, and also considering that the estimation is based only on visual information. These errors, especially the position errors, should be analyzed carefully, taking into account that GPS estimation is not an appropriate ground truth data.

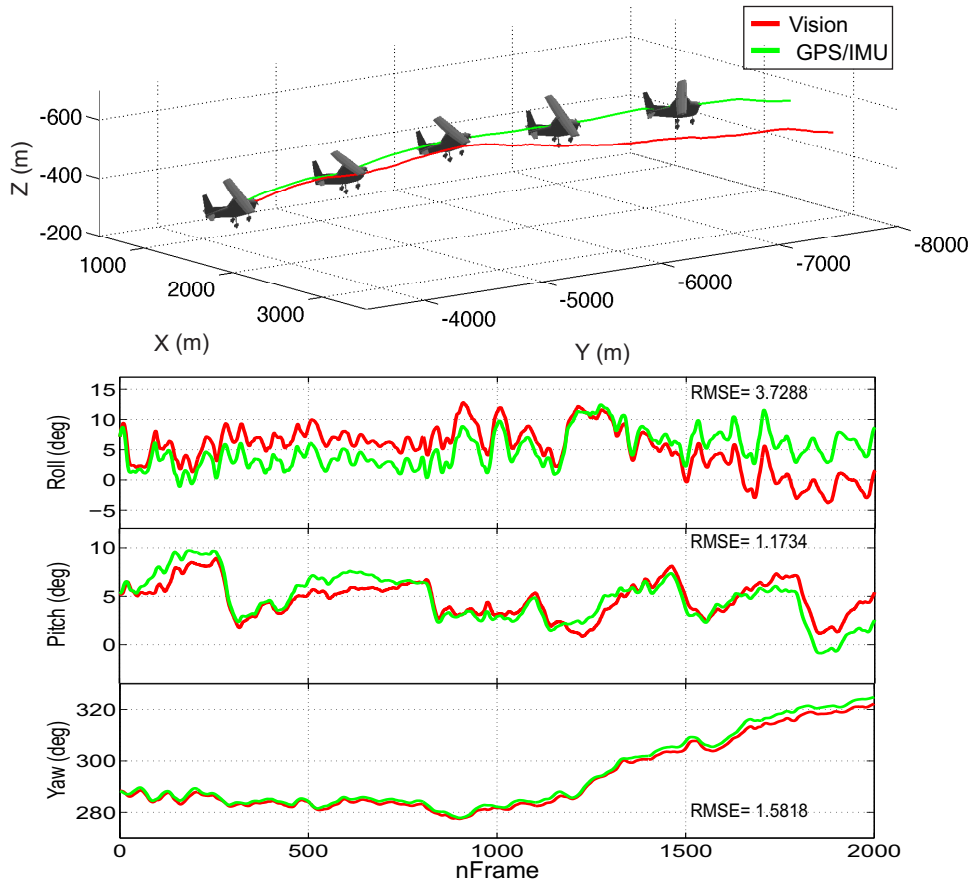


Figure 5.8: Results of the take-off stage. The visual estimation (red/dark line) is compared with the GPS/IMU (green/light line) data. As can be seen, the signals have a similar behavior.

5.4.2.2. Test 2: Cruise

In this test, images from the cruise stage extracted from the available data set are used to analyze the performance of the pose estimation algorithm. Figure 5.9 presents some of the images used in the test. The airplane was flying over different kinds of terrains, as can be seen in Figure 5.7.

For this test, the HMPMR-ICIA algorithm was configured to have the same combination of motion models as the one of the previous test: the homography in the lowest level of the pyramid (8 parameters), the translation in the highest level (2 parameters), and the similarity (4 parameters) and rotation+translation (3 parameters) in the intermediate levels.

Figure 5.10 shows the results of the algorithm in the cruise stage. The visual estimation (red/dark line) shows a behavior that is similar to that of the GPS/IMU data (green/light line). The 3D reconstruction of the flight using both the on-board sensors (green/dashed line) and the visual estimation (red/solid line) can be seen in the upper graphic. As mentioned before, the visual estimation shows a behavior that is similar to that of the GPS/IMU data.

The RMSEs obtained in the angles' estimation are in the range of 4 degrees, and the RMSEs in position are [376.27 m, 467.15 m, 60.42 m] for X, Y, and Z, respectively. The

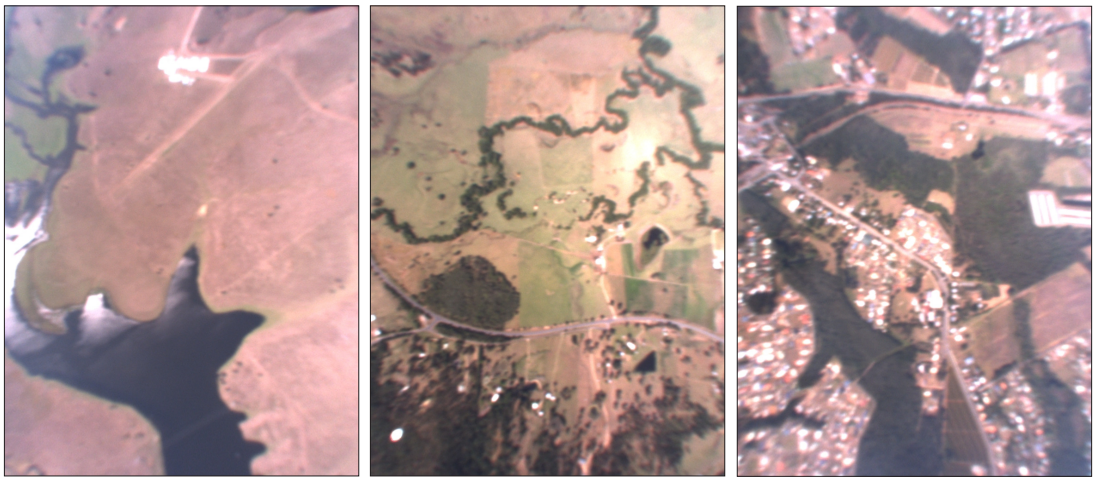


Figure 5.9: Cruise test images. From the available data set (a 90 minutes flight around South-East Queensland, Australia), a collection of images during cruise flight was selected.

previously mentioned errors in position can be considered low if we take into account the total traversed distance and that the mean speed was $\approx$ 211 km/hr. Therefore, those errors represent 7% and 4% of the total traversed distance for the X and Y axes, respectively (6470 m and 7700 m).

5.4.2.3. Test 3: Landing

Images from the landing stage are used to analyze the performance of the pose estimation algorithm. Figure 5.11 presents some of the images used in the test. Because during the landing stage the frame-to-frame motion increases, the HMPMR-ICIA algorithm has been configured to have a combination of motion models in the form 8-4-3-2-2, to deal with the large inter-frame motion.

During the landing stage (see Figure 5.12), the vision algorithm was able to estimate the vehicle's state until the vehicle reached a height of 60 m. From that point on (taking into account the characteristics of the on-board camera), the visual estimation was not robust. This happened because after the vehicle reached that height the conditions of the terrain were not the appropriate ones for the registration algorithm (there was not enough texture information as seen in the right image in Figure 5.11), and because when the vehicle was close to the ground there was not a common frame-to-frame visual information that allowed the estimation of the vehicle's relative state. Therefore, the performance was degraded during the final part of this stage.

In Figure 5.12, it is shown that the visual estimation has a behavior that is similar to the one of the GPS/IMU data. For this test, the obtained RMSEs in orientation are $[2.4°, 3.9°, 5°]$ for roll, pitch, and yaw angles, respectively; and in position the RMSEs are [468.13 m,150.68 m,10.91 m] for X, Y, and Z, respectively.

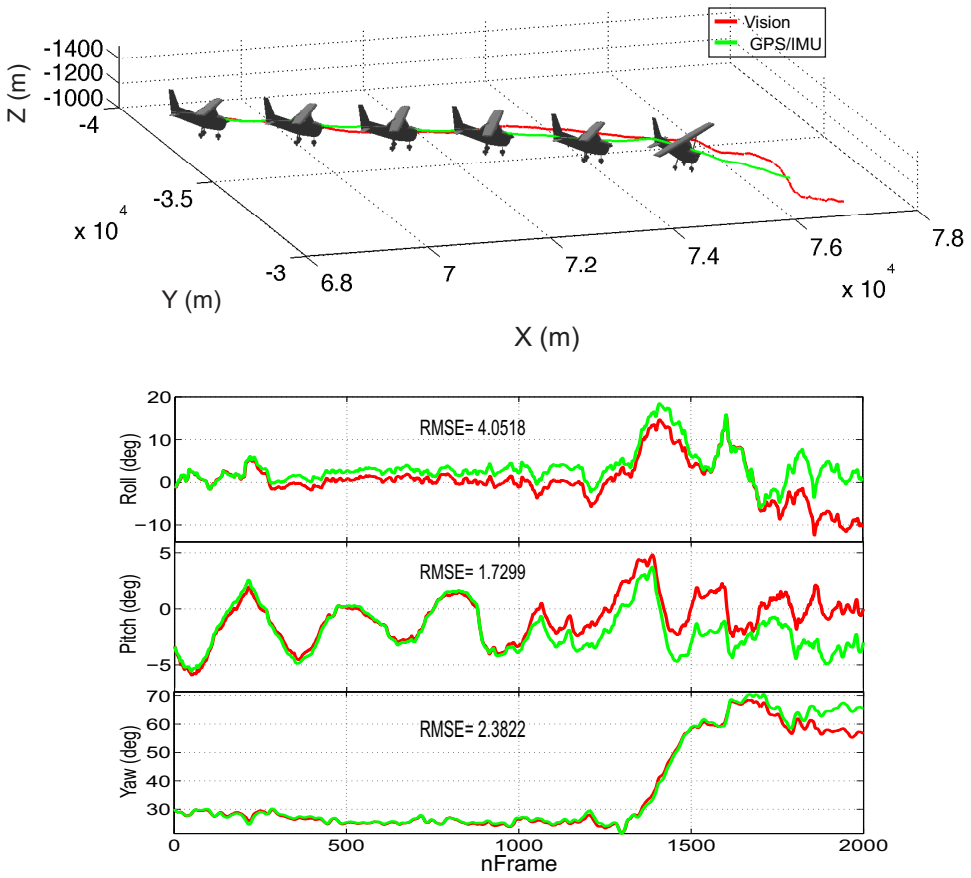


Figure 5.10: Results of the cruise stage. The visual estimation (red/dark line) is compared with the GPS/IMU (ligth/dashed line) data.

5.4.3. Discussion

Results were obtained from tests in different stages of a flight, being two (landing and take-off) of the three selected stages considered critical. In general, the tests show that the vision-based proposed strategy obtains an adequate estimation of the state of the vehicle, and that the behavior of the estimation is comparable to the one obtained with the GPS/IMU sensors. In terms of errors, it can be seen that depending on the stage of the flight, the errors in position are stable, and also low if the total traversed distance and the speed (150-200 km/hr) are considered in the analysis of the results. Therefore, the visual system can offer a good approximation of the vehicle' state when it is so required. The same situation is reflected in the estimation of the vehicle' attitude, where the obtained errors were $< 5°$ during all the stages (see Figures 5.8, 5.10, and 5.12), being all the results obtained for a flight of more than 1 minute. On the other hand, the tests reveal that the assumption of the planar surface to solve the homography decomposition ambiguity is valid in the three analyzed cases.

The limitations of the system are related to the kind of terrain analyzed and the configuration of the on-board camera (i.e. the camera field of view). The algorithm requires texture information to register pairs of images, and from the tests it was found that during

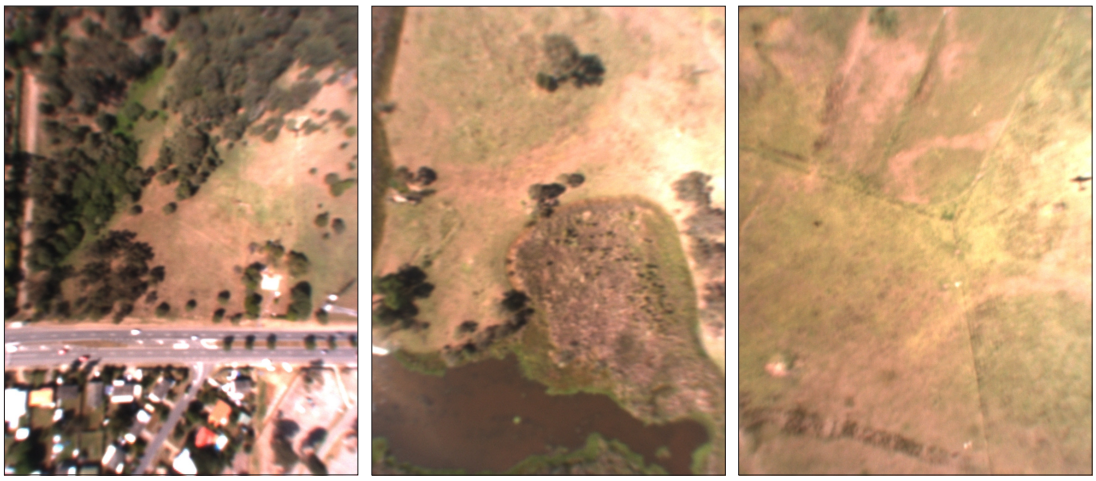


Figure 5.11: Landing test images. From the available data set (a 90 minutes flight around South-East Queensland, Australia), a collection of images from the landing stage was selected.

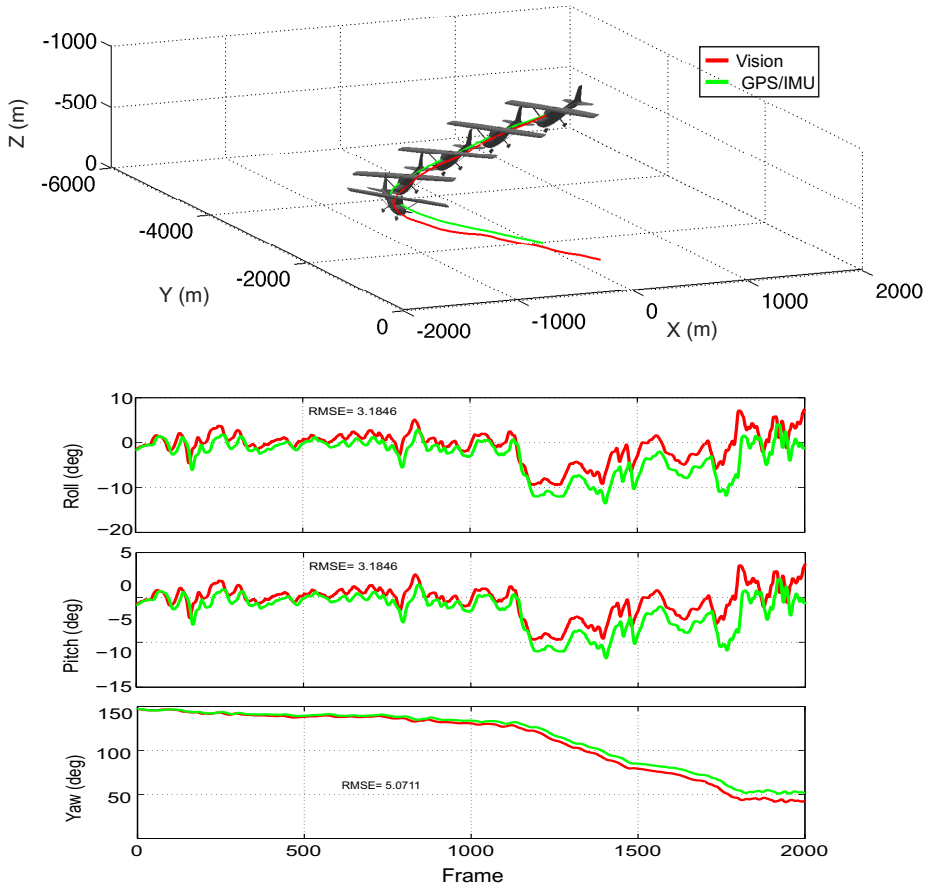


Figure 5.12: Results of the landing stage. The visual estimation (red/dark line) is compared with the GPS/IMU (green/light line) data during the landing stage. As can be seen, the signals have a similar behavior, and the vision algorithm was able to estimate the vehicle's state until reaching a height of 60 m.

take-off and most of the cruise sequences high texture information was available. However, in the last stage of the landing (when the altitude was lower than 60 m), it is seen that due to the low texture information in the sequence (see the right image in Figure 5.11), and also due to the current configuration of the system (the camera field of view), there is no common information in consecutive images, and so the image registration technique is not able to find the appropriate homography between images.

On the other hand, in the tests it has been shown that the assumption of the planar surface to solve the homography decomposition ambiguity is valid in the three analyzed cases, and that the adopted image registration strategy allows to obtain real-time frame rates in the operation of the algorithm: 12 FPS taking into account that the image size is 1024×768.

In terms of performance, it has been shown that the adopted image registration strategy (HMPMR-ICIA + variation of the number of pixels used in each level of the pyramid) allowed to achieve the pose estimation at real-time frame rates (12 FPS, using images of 1024×768 pixels size), without compromising the accuracy of the estimation. The advantage of this strategy, compared with previous approaches in the literature, is that in this strategy direct methods are used. Hence, the motion is solved without intermediate steps such as feature extraction and matching. Additionally, since the HMPMR-ICIA is used, real-time frame rates are obtained, especially under large image motions. Therefore, the results obtained in [158] are extended. In it, an algorithm based on direct methods was proposed but its complexity did not permit its real-time operation.

On the other hand, the speeds reached by the algorithm can sometimes be improved, depending on the application and the degree of precision the system requires: a faster speed can be obtained by using lower image sizes (the results were obtained with an image resolution of 1024×768 pixels) and by estimating different parameters in the different levels.

Videos of the different tests described in this section can be found in [52].

5.5. Results: Position Estimation for VTOL UAVs

In this section, the pose estimation algorithm based on the decomposition of $\mathbf{H_w}$ (see Section 5.3) is used to estimate the position of the UAV, when it is tracking a helipad. The HMPMR-ICIA algorithm, explained in Section 4, is used for tracking a planar template. In Section 4.6, the performance of the algorithm under the visual conditions present in a vision-based landing task (large frame-to-frame motions, rapid changes in scale) was analyzed. In this test, the information that is recovered by the HMPMR-ICIA algorithm is used to estimate vision-based position information of the state of the UAV, which can be used later for autonomous landing and take-off tasks (see Chapter 7).

As can be seen in Figure 5.13, an on-board camera either, in a downwards-looking or a forwards looking configuration, can be used to capture images from a planar target (e.g. a helipad, a window of a building). If the dimensions of the target are known, the homography $\mathbf{H_w}$ can be estimated, and the relative pose between the camera and the target can be calculated by decomposing the $\mathbf{H_w}$ homography, as was presented in Section 5.3. Following this strategy, two tests are presented in this section.

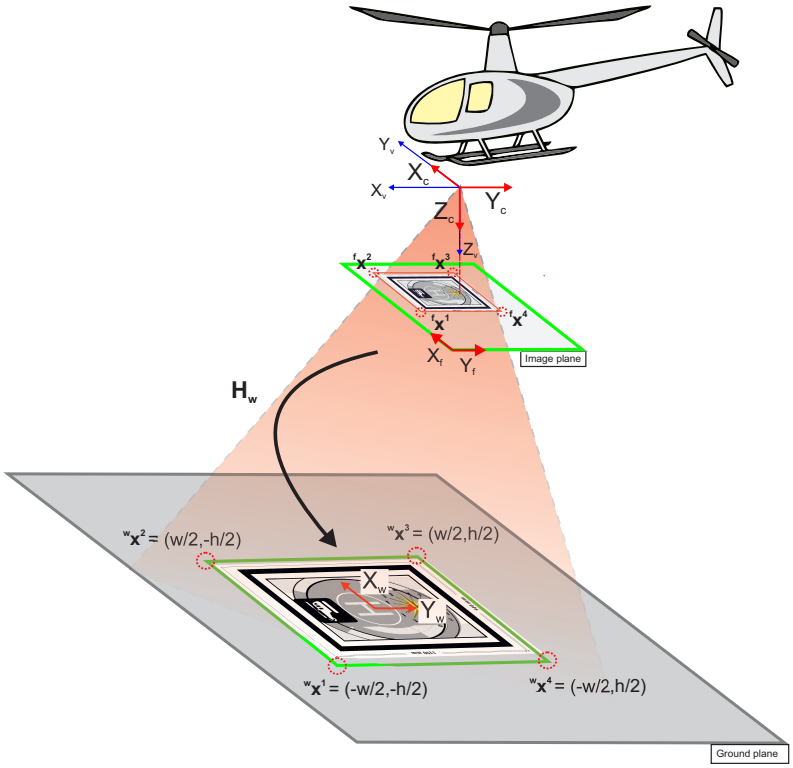


Figure 5.13: Position estimation strategy. The 2D positions of the object to track in the image plane are transformed into 3D positions assuming that the dimensions of the object are known, that the known 3D points lie on a plane, and that the camera calibration parameters are known.

In the first test, accurate position estimations of the camera obtained using a VICON [195] system are used to compare the vision-based position estimation obtained with the proposed algorithm. This test was conducted in a laboratory facility using a scaled helipad, and moving the camera manually recreating the descend and ascend processes during a landing task.

In a second test, the Rotomotion SR20 electric helicopter (the Colibri III system), shown in Figure 4.21 (right image), is used to capture images of a helipad. The position of the helipad in the image plane, and its known size, are used to estimate the UAV's position and orientation. In this test, we compare the results obtained with the HMPMR-ICIA algorithm based on direct methods with the ones obtained with a feature-based algorithm (the KLT algorithm).

In each test, the analysis of the results is based on the comparison of the vision-based estimations with the estimations obtained with other on-board sensors (the GPS and the IMU). Additional information of the configuration of the vision algorithms used in the tests is found in Section 4.4.

5.5.1. Test 1: Laboratory Test

The pose estimation algorithm based on the decomposition of $\mathbf{H_w}$ (see Section 5.3) is tested in a laboratory facility using a VICON system. Figure 5.14 shows the configuration

of the test.

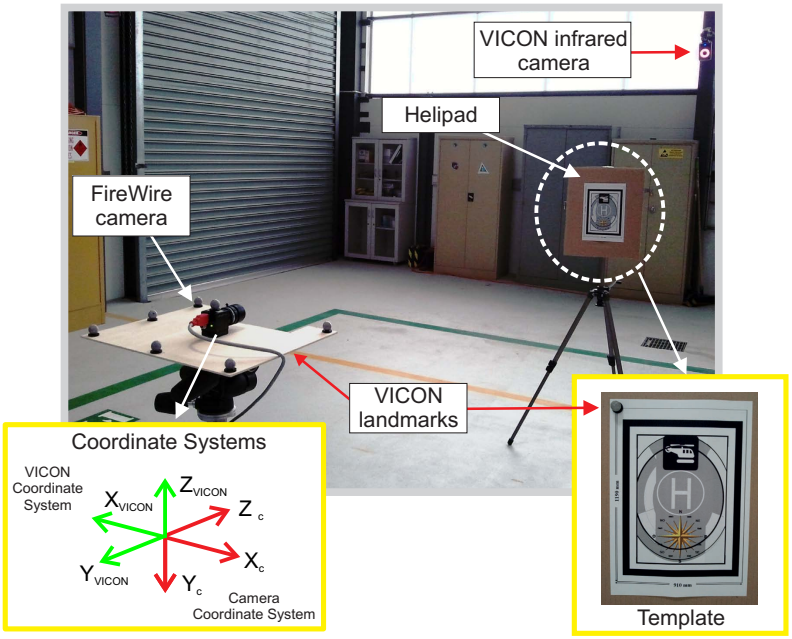


Figure 5.14: Experimental setup. A FireWire camera that is moved manually simulates the UAV during take-off and landing tasks. This camera captures images of a scaled helipad. Ground truth data is generated using a VICON system that tracks infrared landmarks located on the FireWire camera and on the helipad.

For the experiment, a scaled helipad is used as image template (the object to track). A FireWire camera moves forwards and backwards simulating the take-off and landing processes (from the image point of view). This camera captures the image data used in the test. It captures images of size 1024×740 pixels at a frame rate of 7.5 FPS in order to generate image data with a large frame-to-frame motion.

The VICON system [195], composed of five infrared cameras, is in charge of detecting the position and orientation of the helipad and the FireWire camera, by detecting and tracking infrared landmarks (see Figure 5.14). The system provides accurate 3D position information (with sub-milimeter and sub-degree precision) of the helipad and the FireWire camera with respect to the VICON coordinate system, shown in Figure 5.14, at real-time frame rates (100 Hz). This information is used as ground truth data in order to analyze the visual estimation obtained with the pose estimation algorithm.

The MP structure of the HMPMR-ICIA algorithm has been configured to obtain the homography matrix in the lowest level of the pyramid (8 parameters), taking into account that the pose estimation algorithm requires this data to estimate the position. Therefore, the MP configuration of the HMPMR-ICIA that is used is the 8-4-3-2: i.e. the homography in the lowest level of the pyramid (8 parameters), the translation in the highest level (2 parameters), and the similarity (4 parameters) and rotation+translation (3 parameters) motion models in the intermediate levels.

Figure 5.15 presents a collection of images illustrating the performance of the tracking task conducted by the HMPMR-ICIA. The green/light box indicates the results in each

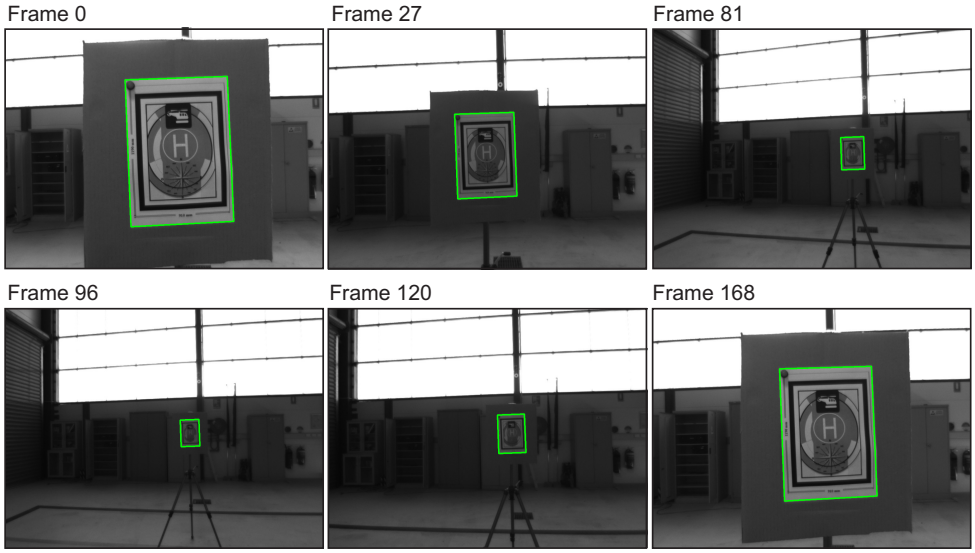


Figure 5.15: Tracking results of the laboratory test. The green/light box indicates the results of the HMPMR-ICIA tracking the helipad.

frame. The helipad was tracked during the entire task, in spite of the different changes in scale (e.g. see Figure 5.15, Frames: 0, 95, 168), the quality of the images (dark images), vibrations (the camera was moved manually), and the large frame-to-frame motion of the sequences (images were acquired at 7.5 FPS).

The 2D positions of the four corners of the helipad in the image plane recovered by the HMPMR-ICIA algorithm and the known 3D position of these corners in the world plane (see Figure 5.13) are used to estimate the 3D position of the FireWire camera. The vision-based positions are obtained with respect to the camera coordinate system, and then transformed to the VICON coordinate system shown in Figure 5.14. This position estimation, as explained in Section 5.3, is found considering that the dimensions of the helipad and the camera calibration matrix are known (the latter has been found using the camera calibration toolbox for Matlab [23]).

Figure 5.16 (upper plots and bottom left plot) shows the comparison of the position estimation obtained by the VICON system (green/light line) with the position estimated using the homography recovered by the HMPMR-ICIA algorithm (red/dark line). As can be seen, the position estimated by the HMPMR-ICIA algorithm (red/dark line) shows behavior and values that are similar to the ones estimated by the VICON system (green/light line). The RMSE (Root Mean Squared Errors) obtained in the three axes are < 6 cm.

The bottom-right plot of Figure 5.16 shows the errors in each axis. We can see that during the whole sequence, the errors were always below 10 cm, and only in one point an error of 25 cm was obtained in the Y axis (the depth, the one that would correspond to the UAV height estimation). Nonetheless, these errors are low taking into account that the vision estimations are from a monocular system. These estimations are considered good taking into account that GPS-based position estimations are around 1 m under good conditions.

Thumbnail images in Figure 5.16 show the correlation of the visual data with the estimated data. These images have been manually enhanced (compared with the real

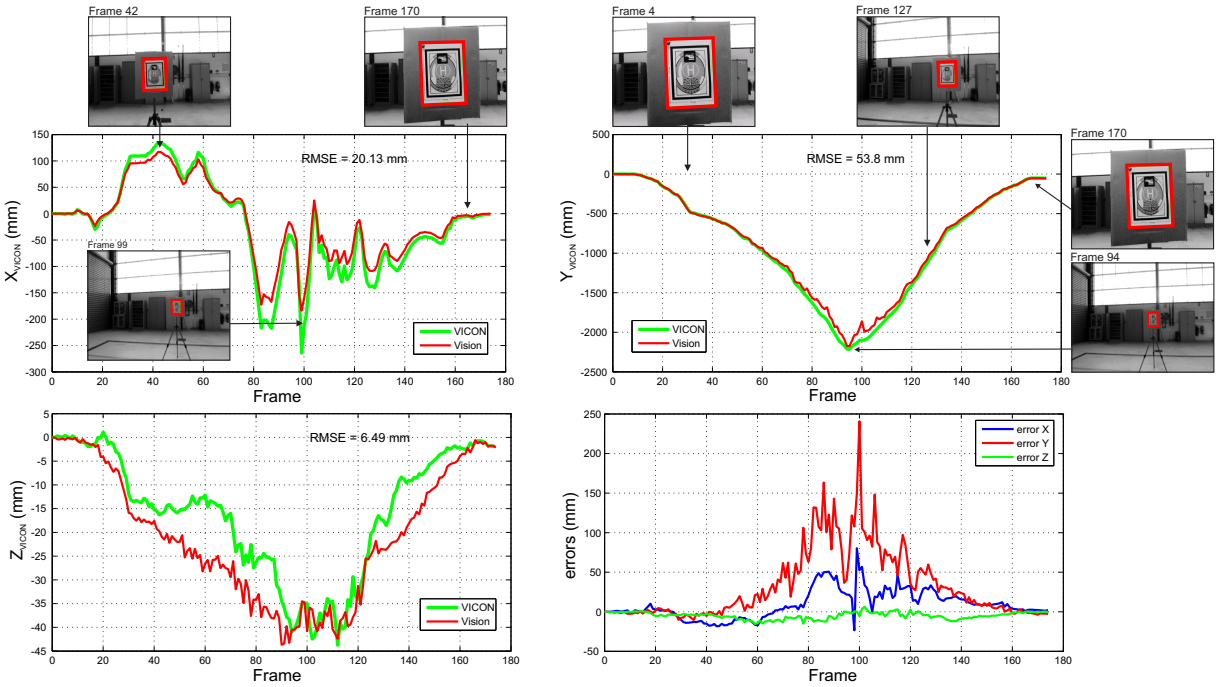


Figure 5.16: Comparison with ground truth data. The position of the FireWire camera estimated by the VICON system (green/light line) is compared with the position estimated using the homography recovered by the HMPMR-ICIA algorithm (red/dark line). The bottom-right plot shows the errors obtained in each axis. Both data are expressed with respect to the VICON coordinate system.

ones shown in Figure 5.15) to allow a clear distinction of the template image and the results of the tracking algorithm.

5.5.2. Test 2: On-Board UAVs

In this test, vision-based position and orientation data of a UAV estimated using a direct method (the HMPMR-ICIA) and a feature-based method (the KLT) are analyzed. The performance of the algorithms is examined comparing the vision-based position estimations with the UAV's state estimation obtained from other on-board sensors (the GPS/IMU). The MP configuration of the HMPMR-ICIA that is used in the test is 8-4-3-2: i.e. the homography in the lowest level of the pyramid (8 parameters), the translation in the highest level (2 parameters), and the similarity (4 parameters) and rotation+translation (3 parameters) motion models in the intermediate levels. On the other hand, the KLT algorithm was configured to recover the homography as well.

Figure 5.17 shows the state of the UAV during the test (green/light line) and some of the captured images. The Rotomotion SR20 electric helicopter (the Colibri III system), shown in Figure 4.21 (right image), is used to collect the image data. An on-board camera (FireWire camera) placed in a downwards-looking configuration is used to capture the images (640×480 pixels). The UAV is flying over the helipad following the trajectory described in Figure 5.17: the UAV moves to the left (the object in the image moves to the right, see Frame 114), and then the helicopter moves forward (the object seen in the

image plane moves backwards, see Frame 386).

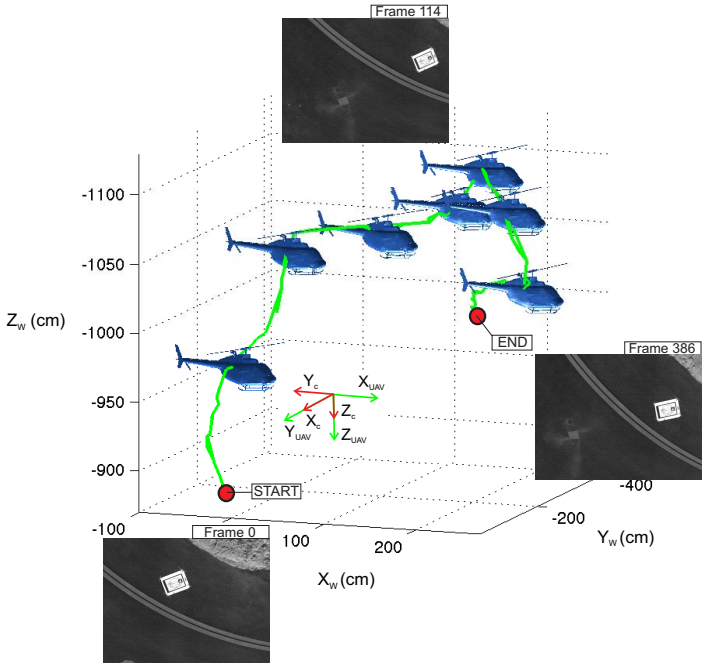


Figure 5.17: Flight test. A helicopter is flying over a helipad, and the on-board camera is used to capture the images. On-board sensors (GPS/IMU) are in charge of estimating the UAV's state (green line), and will be used to compare the vision-based results.

A collection of images illustrating the results of the tracking task obtained by the HMPMR-ICIA (first row) and the KLT algorithms (second row) is shown in Figure 5.18. As can be seen in Frames 224 and 384, the adverse conditions of the task (constant vibrations) make the KLT algorithm unable to track the template in all the frames of the sequence, whereas the HMPMR-ICIA algorithm tracked the template properly (see Figure 5.18, first column).

The known dimensions of the helipad are used to define the 3D coordinates of the helipad with respect to the world reference frame located in its center, as can be seen in Figure 5.13. With these 3D positions and the 2D positions of the corners of the helipad in the image plane recovered by the tracking algorithms (the HMPMR-ICIA and the KLT), the $\mathbf{H_w}$ homography is estimated and decomposed (see Section 5.3), in order to recover the 3D motion of the UAV. The vision-based positions are obtained with respect to the camera coordinate system shown in Figure 5.17, and then are transformed to the UAV coordinate system shown in Figure 5.17.

The different plots of Figure 5.19 show the comparison of the UAV's state estimated by the GPS/IMU sensors (green/light-solid line) with the one calculated with the data obtained with the tracking algorithms: the KLT (red/dark-dashed line) and the HMPMR-ICIA (blue/dark-solid line). As can be seen, the estimations based on the results of the HMPMR-ICIA tracking algorithm (blue/dark-solid line) show a behavior that is similar to the one obtained by the GPS/IMU sensors (green/light-solid line). The RMSE (root mean square error) for the estimations based on the HMPMR-ICIA are [0.31 m, 0.25 m, 0.16 m] for the X_w, Y_w, Z_w axes, respectively; and 1.7° for the yaw angle.

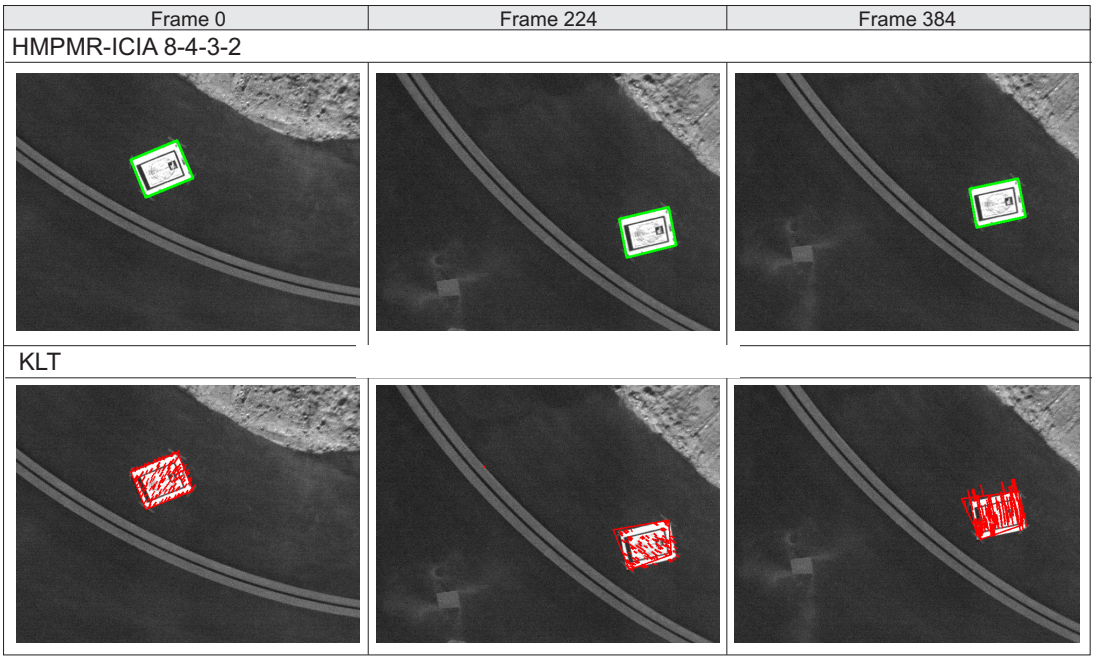


Figure 5.18: Tracking results flight test. The first row shows the results of the HMPMR-ICIA, and the second row the ones of the KLT. The KLT is unable to track the template in all the frames of the sequence, whereas the HMPMR-ICIA algorithm manages to perform the tracking task.

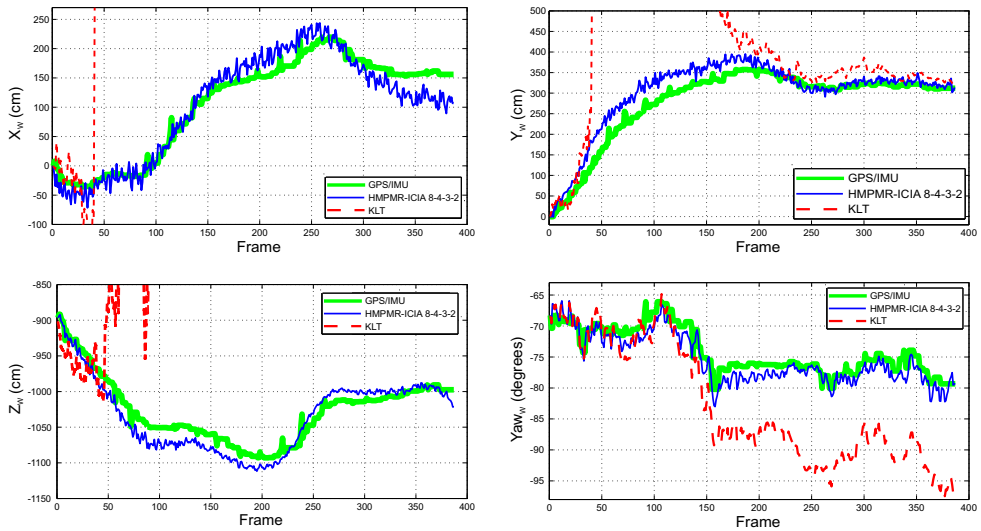


Figure 5.19: Comparison with IMU/GPS data. The UAV's state estimated by the HMPMR-ICIA algorithm (blue/dark-solid line) presents behaviors and values that are similar to the ones obtained by the GPS/IMU sensors (green/light-solid line). The state recovered by the KLT algorithm (red/dark-dashed line) does not allow a good reconstruction of the position and orientation of the UAV.

In the images shown in Figure 5.18, it can be seen that the KLT algorithm was unable to track the template in all the frames of the sequence, and that as a consequence of this,

the $\mathbf{H_w}$ homography recovered by this algorithm does not allow a good reconstruction of the position and orientation of the UAV. As can be seen in Figure 5.19, the position estimated with the KLT algorithm (red/dark-dashed line) diverges from the ground truth data (green/light-solid line).

5.5.3. Discussion

Tests have been conducted to analyze the performance of the pose estimation algorithm based on the $\mathbf{H_w}$ homography. Different mechanisms have been used to evaluate the results: real image data from flights, GPS/IMU data, and accurate position estimations using a VICON system. Additionally, the performance of the HMPMR-ICIA recovering the pose of the UAV has been compared with the one obtained with the KLT feature-based algorithm.

The results show a good correlation between the ground truth data and the position data estimated using the homography based on the HMPMR-ICIA algorithm. Additionally, the results show that the position and orientation data obtained with the HMPMR-ICIA was better than the ones obtained with the well known KLT feature-based algorithm.

From these tests, it can seen that not only the HMPMR-ICIA algorithm was able to track the template, but also that the homography estimated based on the results of the HMPMR-ICIA algorithm can be used for obtaining robust important information (position information) for vision-in-the-loop tasks at real-time frame rates. Therefore, it validated the use of the tracking and pose estimation strategies to provide valid vision-based data for UAV applications.

5.6. Results: Position Estimation for Autonomous Air-to-Air Refuelling

In this test, the pose estimation strategy based on the $\mathbf{H_w}$ homography is used to solve the position estimation problem for autonomous aerial refuelling tasks. In Section 4.7, the drogue and probe method for aerial refuelling was introduced. There, it was mentioned that for autonomous air-to-air refuelling (AAAR) capabilities, the receiver aircraft must be able to detect, track, and estimate the relative position of the drogue, taking into account that the drogue position in the image plane is affected by the effects of the tanker and the receiver aircrafts and by environmental conditions.

The HMPMR-ICIA algorithm was adapted to solve the drogue tracking problem in AAAR tasks (as was shown in Section 4.7). The proposed vision-based strategy contains four stages: detection, initialization, tracking, and 3D position estimation, as shown in Figure 5.20. The algorithm is initiated with a lost status $\mathrm{L} = 1$ (i.e. no drogue has been detected). The detection stage is then used to find the region of interest (ROI) or image template ($\mathbf{T}_{(0)}$) corresponding to where the drogue is located in the first image $\mathbf{I}_{(0)}$ (it is found automatically using a template matching algorithm, see [126]).

Once image $\mathbf{T}$ is found, the tracking algorithm is initialized (different components of the image registration algorithm are calculated). This step is carried out every time the detection stage is activated (i.e. when $\mathrm{L} = 1$).

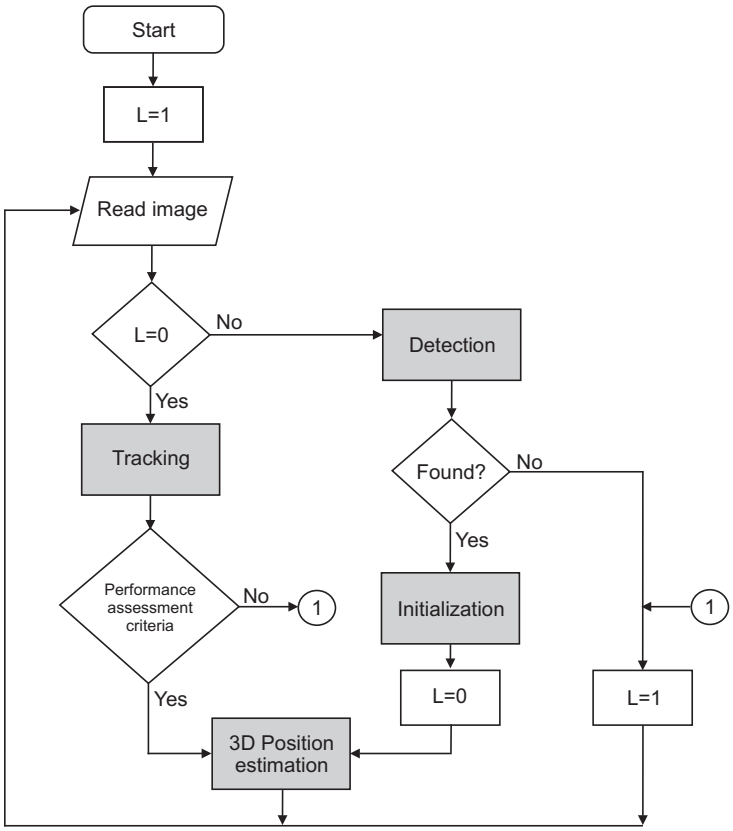


Figure 5.20: Proposed visual tracking system for AAAR tasks.

When a new image is analyzed (e.g. $\mathbf{I}_{(1)}$), if L = 0, the adapted HMPMR-ICIA algorithm is used to track the drogue. Most of the time the tracking stage operates in isolation, if the tracking stage is unable to determine the position of the drogue with a high degree of confidence, then the algorithm switches to the detection stage. To switch between the detection and tracking stages, performance assessment criteria were proposed in [126] to monitor the behavior of the tracking algorithm. If one of those criteria is not satisfied, the lost status is activated (L = 1); and as shown in Figure 5.20, the detection stage will then be used until the drogue is found again.

Once the 2D position of the drogue in the image plane is known either by using the detection or via the tracking algorithm, a 3D position estimation stage is used to calculate the three-dimensional position of the drogue with respect to the probe coordinate system. This estimation is found assuming that the camera is calibrated and that the dimension of the drogue is known.

The laboratory testbed shown in Figure 4.31 is used to reproduce the relative motion of the probe and the drogue. Actual refuelling hardware (probe and drogue) and simulated aircraft motion data that recreate the refuelling procedure are used in the tests. The camera on-board robot R2, at the base of the probe, as shown in Figure 4.31, is in charge of capturing image data of the drogue.

Figure 5.21 shows the configuration of the pose estimation algorithm for aerial refuelling. In order to apply the pose estimation algorithm based on $\mathbf{H}_{\mathbf{w}}$ (see Section 5.3), it

is required to assume that the camera is calibrated, and that the 3D points of the drogue, used by the pose estimation algorithm, lie on a plane. Additionally, it is assumed that the diameter of the drogue is known. This known diameter is used to define a plane that inscribe the drogue, as shown in Figure 5.21, where four 3D points that lie on this plane are defined ($^{w}\mathbf{x}^1$, $^{w}\mathbf{x}^2$, $^{w}\mathbf{x}^3$, $^{w}\mathbf{x}^4$).

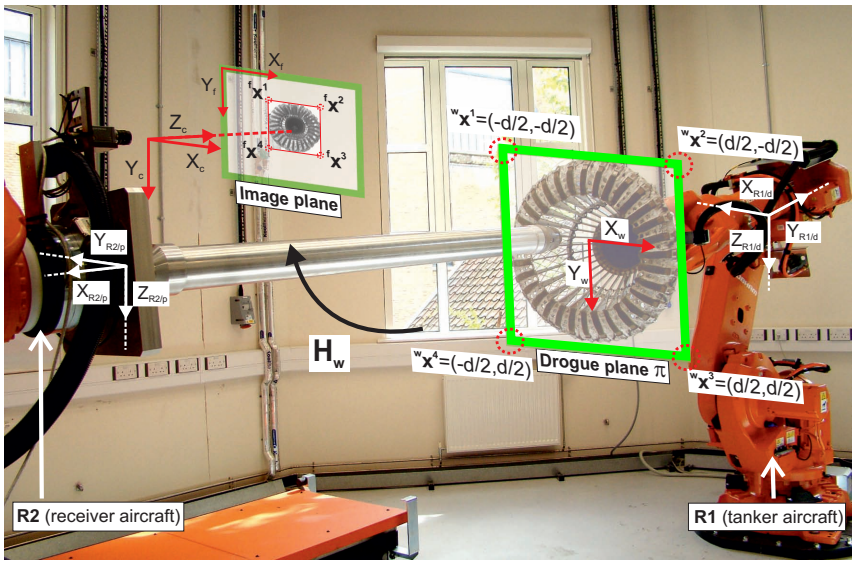


Figure 5.21: Position estimation strategy for aerial refuelling. The 2D positions of the drogue in the image plane are transformed into 3D positions assuming that the diameter of the drogue is known, the known 3D points lie on a plane, and the camera calibration parameters are known.

On the other hand, assuming that the tracking algorithm is able to robustly provide the position of the drogue in the image plane, then the position of the four corners that inscribe the drogue can be defined, as shown in Figure 5.21, as follows: $^{f}\mathbf{x}^1$, $^{f}\mathbf{x}^2$, $^{f}\mathbf{x}^3$, $^{f}\mathbf{x}^4$. Then, with these four 3D points and the four 2D points in the image plane, $\mathbf{H}_w$ can be obtained using Equation (5.15); and from this, the translation vector can be estimated as was shown in Equation (5.16).

Therefore, the 3D position of the drogue (subscript d), or the position of R1 (subscript R1) with respect to the camera coordinate system (superscript c), is found $\mathbf{t} = {}^{c}\mathbf{t}_{\mathbf{v}d} = {}^{c}\mathbf{t}_{\mathbf{v}R1}$, where $\mathbf{t}_\mathbf{v}$ refers to a vision-based estimation.

In this section, two tests are conducted. In the first one, the pose estimation algorithm is used to estimate either the motion of the drogue or the motion of the probe. Basic motions of the probe and the drogue are used. When the drogue moves, the probe (R2/receiver aircraft in Figure 5.21) is kept stationary; and when the probe moves, the drogue (R1/receiver aircraft in Figure 5.21) is kept stationary. These basic motions are conducted at different speeds.

On the other hand, in the second test, simulated aircraft motion data is sent to one of the robots (the one that simulates the receiver aircraft, shown in Figure 5.21), in order to recreate a refuelling procedure. The data corresponds to motions of the probe towards the drogue (the drogue is kept stationary), simulating the aerial refuelling task under two scenarios: light turbulence and moderate turbulence. During the test, the drogue is kept

stationary, and motion of the probe is used exclusively to reproduce the relative motion of the two bodies.

The evaluation of the vision-based position estimation is based on a comparison of the motion estimated by the tracking algorithm with the position data recorded from the robots, which is used as ground truth data. This comparison is evaluated analyzing the RMSE (root mean square error) between the data, and analyzing the behavior of the estimation during the tests.

5.6.1. Basic Motions

The image sequence used in this test contains basic motions of the drogue (moving left, right, up and down) where the background information changes, and also contains basic motions of the probe in relation to the drogue (forwards and backwards), that in some situations cause occlusions of the drogue by the probe, or make the drogue go out of the FOV (field of view) of the camera. In Section 4.7, the HMPMR-ICIA algorithm was adapted in order to track the drogue. In this test, that algorithm is used to solve the drogue tracking problem, as required by the pose estimation algorithm. Therefore, the HMPMR-ICIA algorithm is used to track the drogue, and the pose estimation algorithm, presented in Section 5.3, based on the $\mathbf{H_w}$ homography, is used to estimate the motion of either the probe or the drogue.

Figure 5.22 presents a collection of images that illustrate the results of the HMPMR-ICIA algorithm tracking the drogue. In these images, it is also possible to see the different motions used in the tests: left and right (first row), up and down (second row), and scale changes (third row).

On the other hand, Figure 5.23 shows the results of the estimated position of the drogue when it moves in the Y_p axis (the drogue moves to the left and to the right). This motion is conducted at three different speeds: 400 mm/s, 1000 mm/s, and 2000 mm/s. This figure shows the comparison of the motion estimated by the robots (green/light-solid line) with the motion estimated by the HMPMR-ICIA algorithm (red/dark-dashed line), in the Y_p axis. As can be seen, the motion estimated by the HMPMR-ICIA algorithm (red/dark-dashed line) shows a behavior that is similar to the motion performed by the robots (green/light-solid line). The vision-based estimation is also evaluated analyzing the RMSE (root mean square error) between the robots data and the vision data. For this axis, the RMSE was < 11 cm.

Additionally, in Figure 5.23 the thumbnail images located in the upper and lower parts of the plot show that the drogue was tracked in all the frames of the sequence (the red/dark box covers the drogue), and that the motion that was estimated corresponds to the one conducted by the robots, e.g. robot on the left, in Figure 5.23, Frames 160 and 1617; robot on the right, in Figure 5.23, Frames 275 and 1500; and robot in the center, in Figure 5.23, Frame 858.

Figure 5.24 shows the results of the pose estimation of the drogue when it moves up and down in the Z_d axis, at three different speeds: 400 mm/s, 1000 mm/s, and 2000 mm/s. The thumbnail images located in the upper and lower parts of the plot show that the drogue was tracked in all the frames of the sequence (the red/dark box covers the drogue), and that the motion estimated by the HMPMR-ICIA algorithm (red/dark-

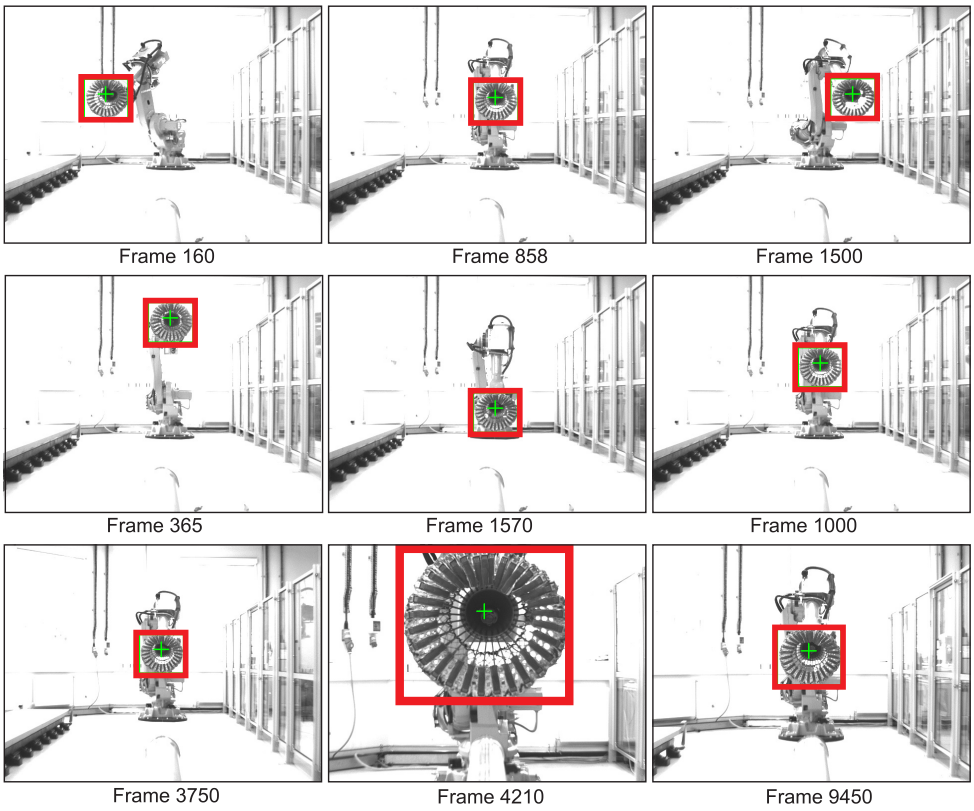


Figure 5.22: Tracking results during basic motions. Left and right (first row), up and down (second row), and scale changes (third row) The green/light crosshair and red/dark box indicate the results of the tracking task.

dashed line) corresponds to the one conducted by the robots (green/light-solid line). The RMSE obtained for this axis was < 34 cm.

Finally, Figure 5.25 shows the results of the estimated motion in the X_d axis (scale changes). In this sequence, the probe is kept stationary in Frames 0-1750, 1900-3632, 6920-8710, and 8860-10000; and from Frame 3720 to Frame 6900 the Robot 2 is commanded to move towards the stationary drogue and backwards, at two different speeds: 400 mm/s and 1000 mm/s. The motions in this test are likely to be experienced in an areal refuelling exercise. As can be seen in the thumbnail images shown in Figure 5.25, the HMPMR-ICIA algorithm tracks the drogue throughout the sequence in spite of the significant changes in scale (from 0 m to 6 m). Additionally, it can be seen that the motion recovered by the vision algorithm (red/dark-dashed line) has behavior and values that are similar to the motion conducted by the robot (green/light-solid line). The RMSE obtained for this axis is < 7 cm. This error, as well as the ones obtained in the other axes, are low considering that the estimation is based on a monocular system.

5.6.2. Aerial Refuelling Tests

In this test, the pose estimation algorithm based on the HMPMR-ICIA tracking algorithm is tested using motions representative of an aerial refuelling task. The AAAR

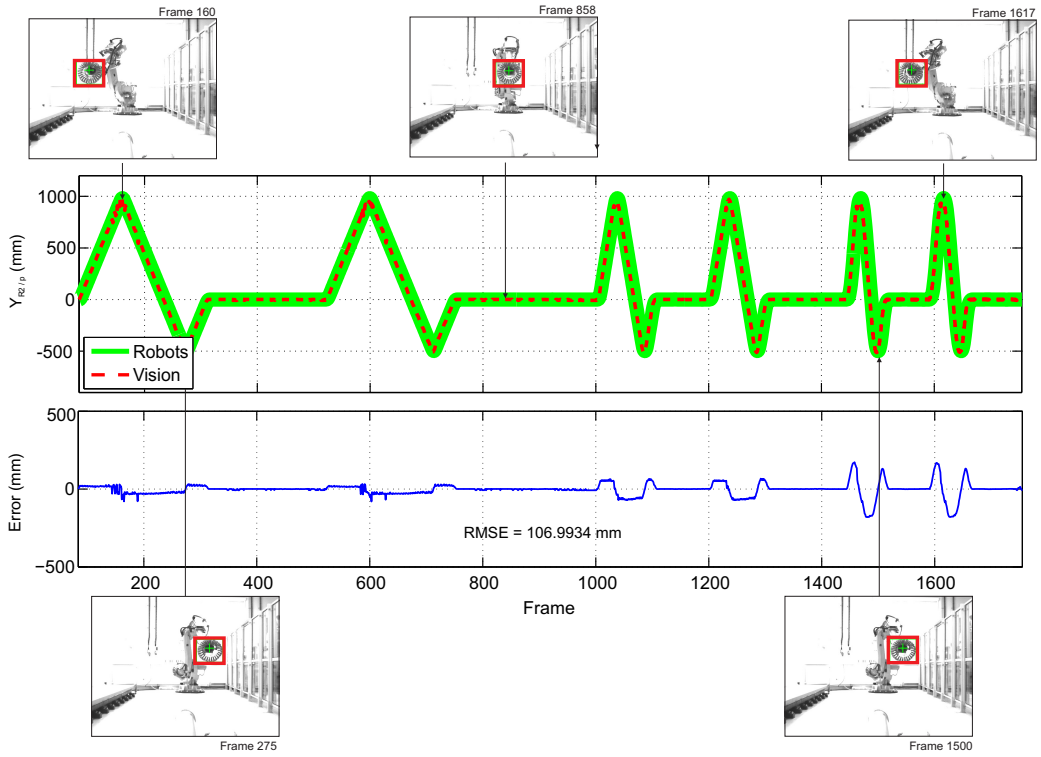


Figure 5.23: Motion estimation in the Y_p axis. Comparison of the motion estimated by the robots (green/light-solid line) with the motion estimated by the HMPMR-ICIA algorithm (red/dark-dashed line). Thumbnail images show the results of the tracking task.

testbed shown in Figure 4.31 provides the motion data for the task, using two scenarios: light turbulence and moderate turbulence. The drogue is kept stationary for these tests, and motion of the probe is used exclusively to reproduce the relative motion of the two bodies.

In this test, simulated aircraft motion data is used. This data is sent to one of the robots (R2, the one that simulates the receiver aircraft, shown in Figure 4.31), in order to recreate a refuelling procedure. The data corresponds to motions of the probe towards the drogue (the drogue is kept stationary), simulating the aerial refuelling task (with both light turbulence and no turbulence effects being considered). During the tests, adverse conditions are present in the image sequences, including the drogue being out of the FOV; changes in appearance (e.g. pitch and roll effects make the aspect of the drogue to change in the image plane); occlusions (drogue occluded by the probe); and sudden, rapid motions. When the drogue goes out of the FOV of the camera due to turbulence effects, a basic template matching algorithm was included in order to detect the drogue when it reappears and to restart the tracking algorithm.

On-line and off-line tests of the visual system during AAAR tasks have been conducted using the AAAR testbed. Nonetheless, for the analysis of the visual system presented in the following paragraphs, the image data and the robots' data were recorded and processed off-line. Positions of the probe tip relative to the drogue (${}^{\mathrm{d}}\mathbf{t}_{\mathbf{r}_\mathrm{p}}$) are recorded in off-line runs of the simulation and used as input to the robot controller to replicate the 6 meter pre-contact approach.

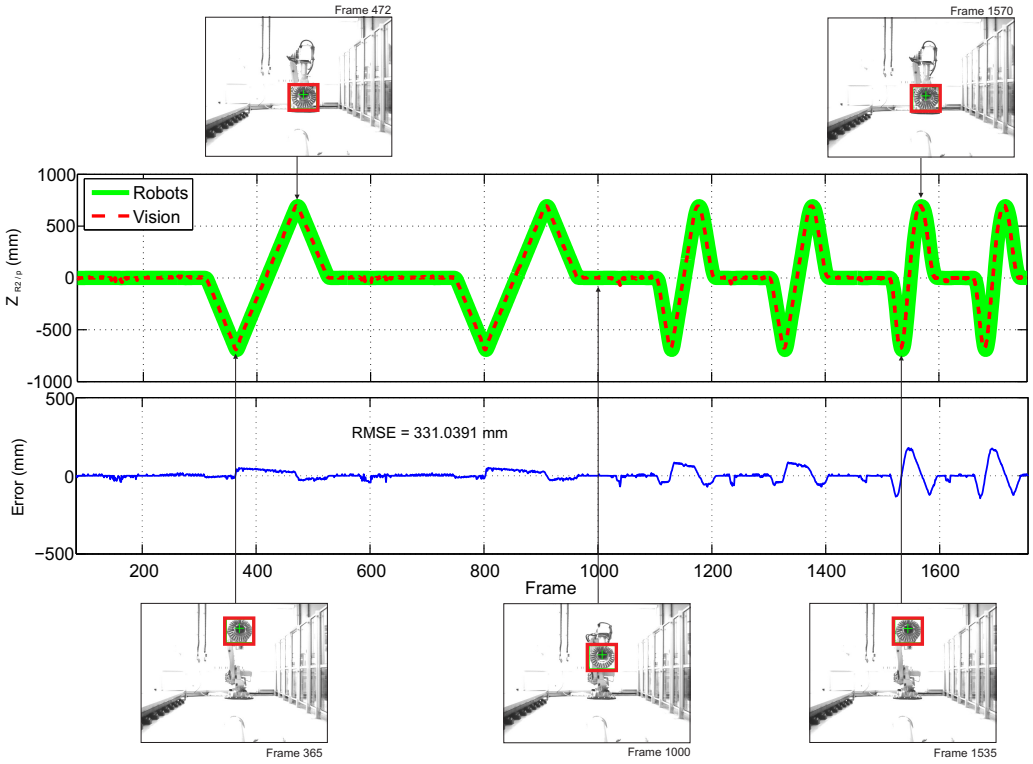


Figure 5.24: Motion estimation in the Z_p axis. Comparison of the motion estimated by the robots (green/light-solid line) with the motion estimated by the HMPMR-ICIA algorithm (red/dark-dashed line). Thumbnail images show the results of the tracking task.

The evaluation of the performance of both algorithms is based on a visual examination of the tracking results: analyzing whether the drogue is tracked in the sequence (the red/dark box must be covering the drogue), and also analyzing the motion model recovered by each algorithm.

5.6.2.1. Test 1: Light Turbulence Conditions

The HMPMR-ICIA algorithm is tested in a refuelling exercise with light turbulence conditions. Figure 5.26 shows the position and orientation data recorded from the robots which is used as ground truth data. Light turbulence conditions represent a more challenging scenario to the tracking algorithm than those exhibited in the no turbulence case, because they generate changes in orientation, including roll, pitch and yaw (Figure 5.26); occlusion of the drogue by the probe; large changes in scale; and sudden motions.

Figure 5.27 presents a collection of images illustrating the performance of the tracking task under light turbulence effects. The red/dark box indicates the results of the visual system. The drogue was detected automatically by the detection algorithm in Frame 1, as shown in Figure 5.27; and was tracked during the entire refuelling task, in spite of the changes in scale (see Figure 5.27, Frames: 1-1632), occlusions (see Figure 5.27, Frames: 665, 903, 931, etc), and periods where part of the drogue was out of the FOV of the camera (see Figure 5.27, Frames: 931, 971, 1613, and 1632).

Figure 5.28 compares the measured positions of the probe computed from the robot

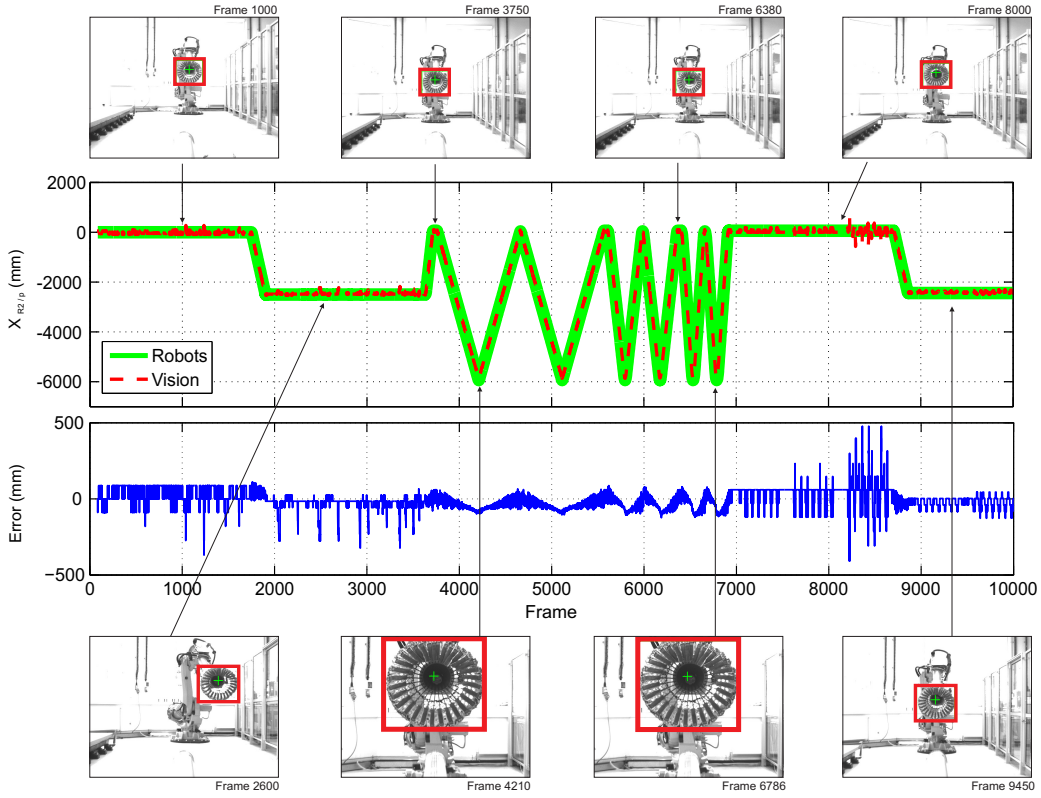


Figure 5.25: Motion estimation in the X_p axis. Comparison of the motion estimated by the robots (green/light-solid line) with the motion estimated by the HMPMR-ICIA algorithm (red/dark-dashed line). Thumbnail images show the results of the tracking task.

joint positions and the positions of the probe estimated by the visual system. The visual system estimates ${}^{c}\mathbf{t}_{vd}/{}^{c}\mathbf{t}_{vR1}$: the position of the drogue (or robot R1) with respect to the camera coordinate system (Equation (5.16)). Nonetheless, because during the task the drogue (robot R1) is kept stationary, the changes in position of the drogue in the image plane are due to the motion of the probe. Therefore, in order to compare the data, the estimated motion ${}^{c}\mathbf{t}_{vd}/{}^{c}\mathbf{t}_{vR1}$ can be used to obtain the relative motion of the probe. For the comparison, the position of the probe with respect to the drogue coordinate system $({}^{d}\mathbf{t}_{rp}/{}^{R1}\mathbf{t}_{rR2})$ recorded by the robot controller $(\mathbf{t}_r)$ is transformed into relative positions of the probe ${}^{p}\mathbf{t}_r$.

Because the visual information does not recover orientations, the image points must be compensated for rotation before the position estimation algorithm acts. This compensation is made using the known orientation data of the probe recorded by the robot controller: relative rotation angles of the probe from the starting point of the tests. Therefore, the visual data ${}^{c}\mathbf{t}_{vd}/{}^{c}\mathbf{t}_{vR1}$ found with Equation 5.16 is transformed into the probe coordinate system ${}^{p}\mathbf{t}_{rd}/{}^{R2}\mathbf{t}_{rR1}$ using the known fixed rotation between both coordinate systems $({}^{p}\mathbf{R}_c)$. With these data, the relative motion of the probe ${}^{p}\mathbf{t}_v$ estimated by the visual system is determined. ${}^{p}\mathbf{R}_c$ is defined as a rotation of 90° in the Y_c axis, followed by a rotation of 90° in the rotated Z_c axis.

Figure 5.29 shows the errors obtained when the position estimated by the tracking algorithm is compared with the position of robot R2 (the receiver aircraft). As can be

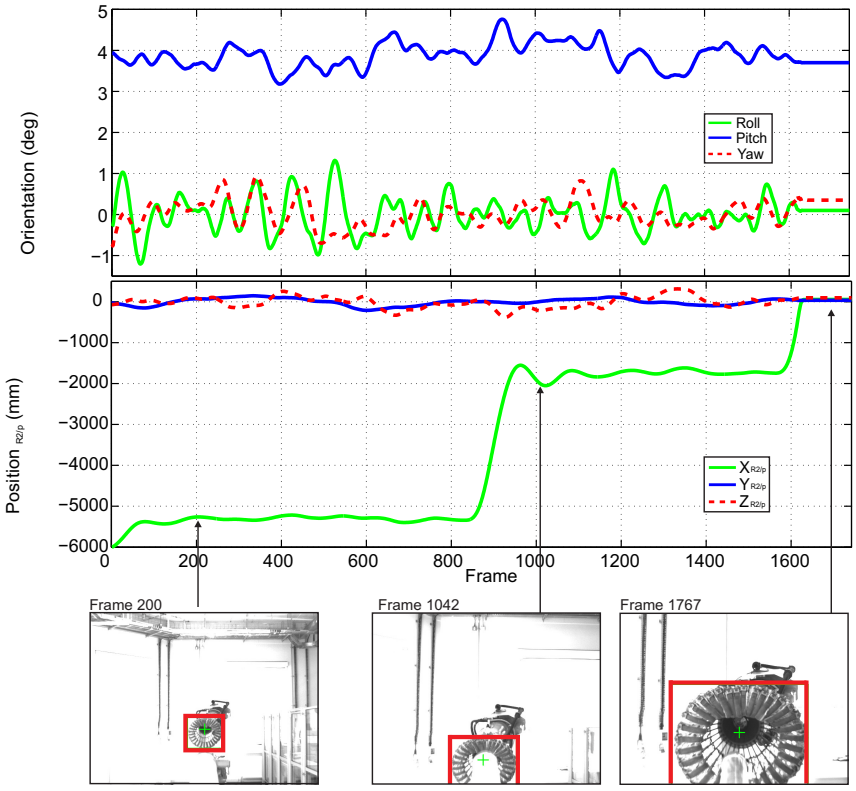


Figure 5.26: Motions of a refuelling task under light turbulence conditions. The AAAR testbed reproduces simulated motions of the receiver aircraft in a refuelling procedure. The data recorded by the robots is used as ground truth data. Thumbnail images show the results of the tracking algorithm.

seen, the obtained vision-based position data have values that are similar to the ones of the GT data. The RMSEs obtained in the three axes are < 55 mm. These errors are low, considering that the estimation is based on a monocular system.

As can be seen Figure 5.28, the motion of the receiver aircraft (R2/p), inferred using the vision-based pose estimation algorithm (red/dashed line), presents a behavior that is similar to the motion of the robots (green/solid line). The RMSE(s) reached by the visual system estimating the position of the probe were in the range of 5.5 cm in the X_p axis, 1.3 cm in the Y_p axis, and 1.6 cm in the Z_p axis. As can be seen, errors in position are low, considering that the estimation is based on a monocular system.

5.6.2.2. Test 2: Moderate Turbulence Conditions

The second test was conducted sending simulated data to the robots under moderate turbulence conditions. Figure 5.30 shows a collection of images illustrating the performance of the tracking task. The drogue was again detected automatically in the first image by the template matching algorithm, and was tracked during much of the refuelling task. The moderate turbulence conditions represent a more challenging scenario to the tracking algorithm, with motions that are more sudden and faster than those exhibited in the light turbulence case. Specific difficulties are posed by: the occlusion of

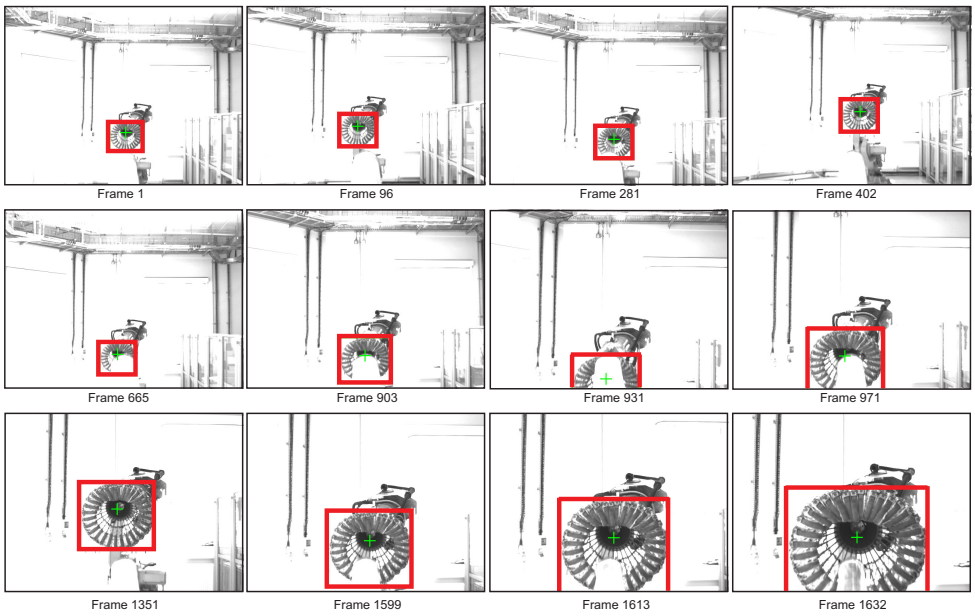


Figure 5.27: Tracking results under light turbulence conditions. The green crosshair and red box indicate the estimated position and extent of the drogue.

the drogue by the probe (Figure 5.30, Frames 670, 694, and 1445, among others); large changes in scale (Frames 1582-1608); changes in orientation, including roll, pitch and yaw (Frame: 670); and periods when the drogue is outside the FOV of the camera (Frame 931).

A significant feature of this test is the disappearance of the drogue from the FOV for approximately 200 frames. When the drogue goes out of the FOV of the camera at Frame 891, the detection algorithm is activated in order to search the drogue. The drogue remains outside the FOV until Frame 1105, where it begins to reappear in the image. At this stage, its position is not immediately recovered: the segmentation scheme used to find and verify the location of the drogue relies on an unobstructed view of the central region of the drogue (see [126] for details). When this region is partially occluded, as seen in Figure 5.30, Frame 1111, the drogue position and size can be misidentified. When the drogue moves further from the occluded zone, the detection algorithm is able to recover the position and size effectively, as in Figure 5.30, Frame 1175. Throughout this period, the tracking algorithm performs well, but it is clear that there is room for a more timely recovery of the drogue position through the implementation of detection methods, which are more robust to occlusions.

Promisingly, the tracking was robust with respect to the large changes in scale and orientation of the drogue. The latter is particularly important, as changes in orientation are not modeled by the transformation applied in the algorithm. The success of the scheme in these circumstances is attributed to the nature of the drogue's appearance: symmetry means that roll does not have a significant effect on the patterns being searched, and that small pitch and yaw motions do not greatly alter the visual characteristics.

On the other hand, Figure 5.31 shows the comparison of the vision-based position data (${}^{\mathrm{P}}\mathbf{t_v}/{}^{\mathrm{R2}}\mathbf{t_v}$) with the robot joint measurements (${}^{\mathrm{P}}\mathbf{t_r}/{}^{\mathrm{R2}}\mathbf{t_r}$). As mentioned in the light

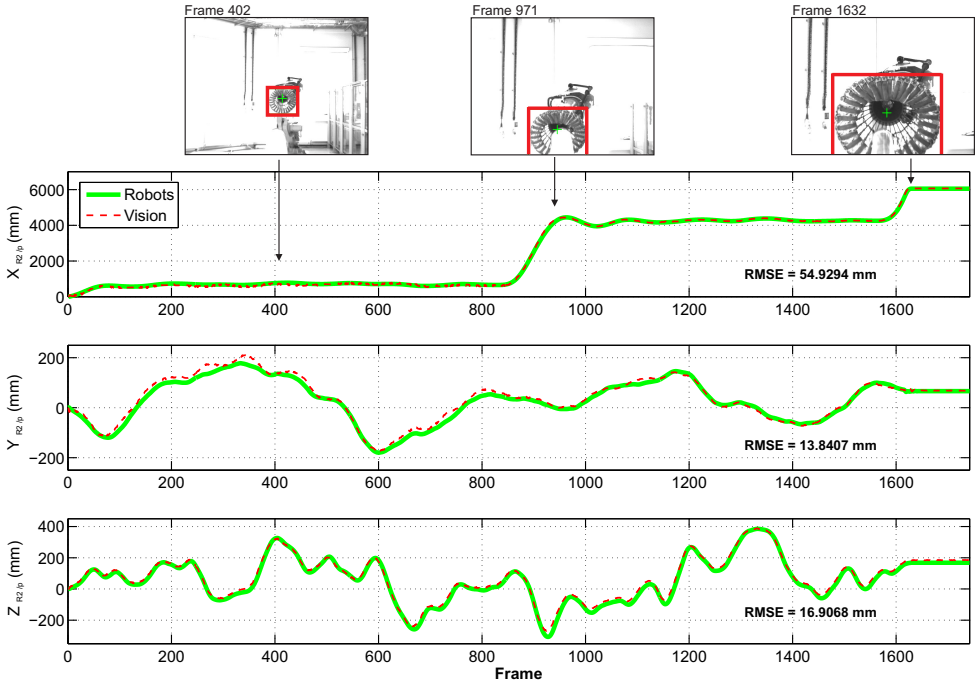


Figure 5.28: Position estimation under light turbulence conditions. The motion estimated by the tracking algorithm (red/dashed line) is compared with the motion of the R2 robot (green/solid line).

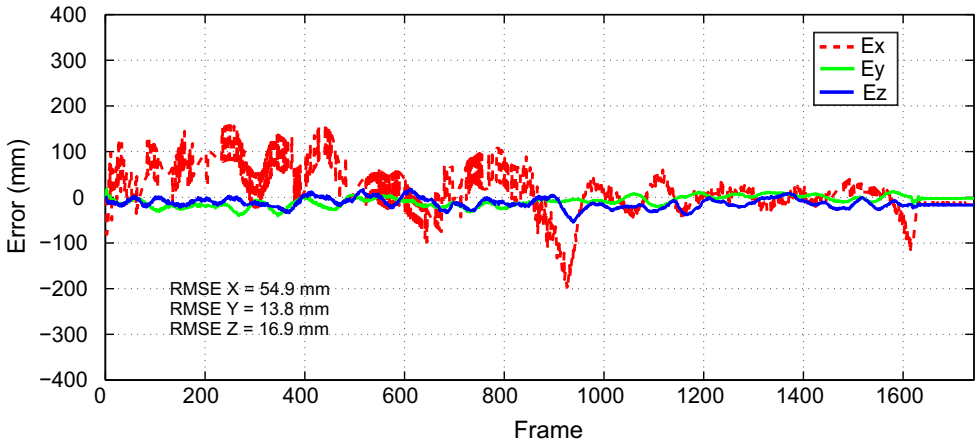


Figure 5.29: Position errors under light turbulence conditions. Errors obtained from the comparison of the position estimated by the tracking algorithm with the position of robot R2 (the receiver aircraft).

turbulence test, the image coordinates of the drogue found by the tracking algorithm are compensated for rotation, and then position data is estimated based on the pose estimation algorithm explained in Section 5.3. This algorithm is used to determine the relative motion of the probe based on visual information ${}^{p}\mathbf{t}_{\mathbf{v}}$.

In this sequence, due to the different motions, it was required to update the template in different frames. Nonetheless, the template matching algorithm was in charge of detecting the drogue when it was so required, allowing the tracking task to continue. In Figure 5.30,

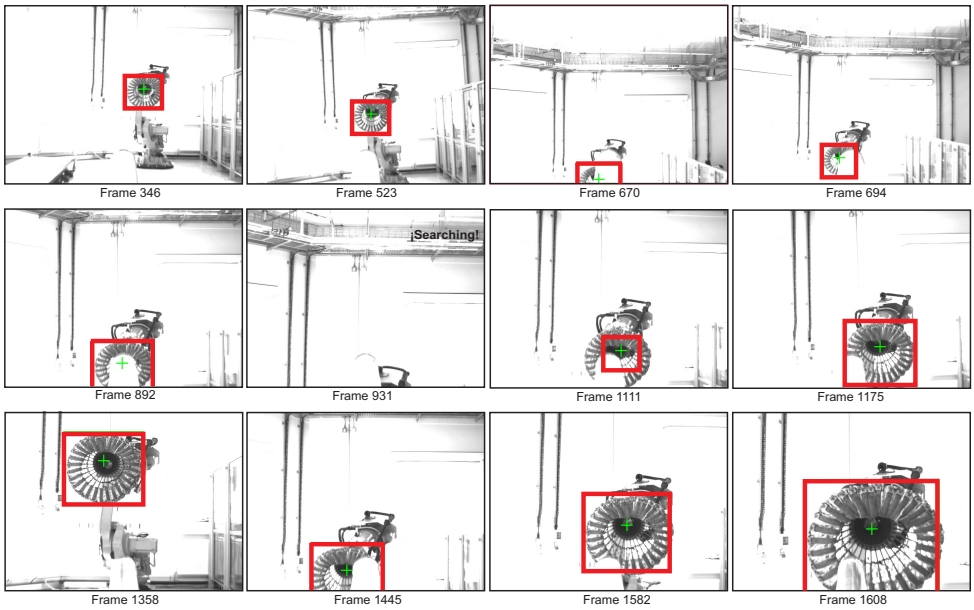


Figure 5.30: Tracking results under medium turbulence conditions. The green/light crosshair and the red/dark box indicate the estimated position and extent of the drogue.

it is possible to see the different times the template was updated by the detection stage. In Frames 891-1105 (the first shaded area), the drogue left the FOV of the camera and the template matching algorithm remained operating until the drogue appeared in the FOV of the camera. When this happened, the detection algorithm recovered the position of the drogue (Figure 5.30, Frame 1105). Nonetheless, when the drogue reappeared its center remained obstructed, as can be seen in Frame 1105, and this caused the position of the drogue to be misidentified (e.g. Frame 1105). In those frames, the tracking algorithm failed to track the drogue due to the low gradient information in the new template (the new template occupied the center of the drogue), and the HMPMR-ICIA algorithm was thus not able to find a good transformation of the parameters. Therefore, the template matching algorithm was activated to find a new drogue, which was correctly detected after Frame 1172, as can be seen in Figure 5.30, Frame 1175. Then, with this new drogue, the tracking task continued.

The shaded areas in the plot represent the moments when the detection algorithm was operating. In those frames (Frames 891-1105, and Frames 1161-1172), position data was not estimated. Nonetheless, for analyzing the results, the RMSE(s) are calculated separately for the initial and final parts of the test. In the first half of the test, from Frames 1-891, the RMSE(s) are 29 cm in the X_p axis, 4.3 cm in the Y_p axis, and 5.1 cm in the Z_p axis. Although the error in the X_p axis is not high for a monocular-based position estimation, its value is higher than in the other axes. This value is obtained basically due to a small misbehavior of the tracking algorithm in Frames 650-690 (see Figure 5.31). This discrepancy is attributed to a small error of the tracking algorithm, when due to the effects of turbulence, a big portion of the drogue goes outside the FOV of the camera and the drogue is also occluded by the probe (see Figure 5.31, Frame 665). In those frames, the percentage of pixels that the tracking algorithm can use to track the template

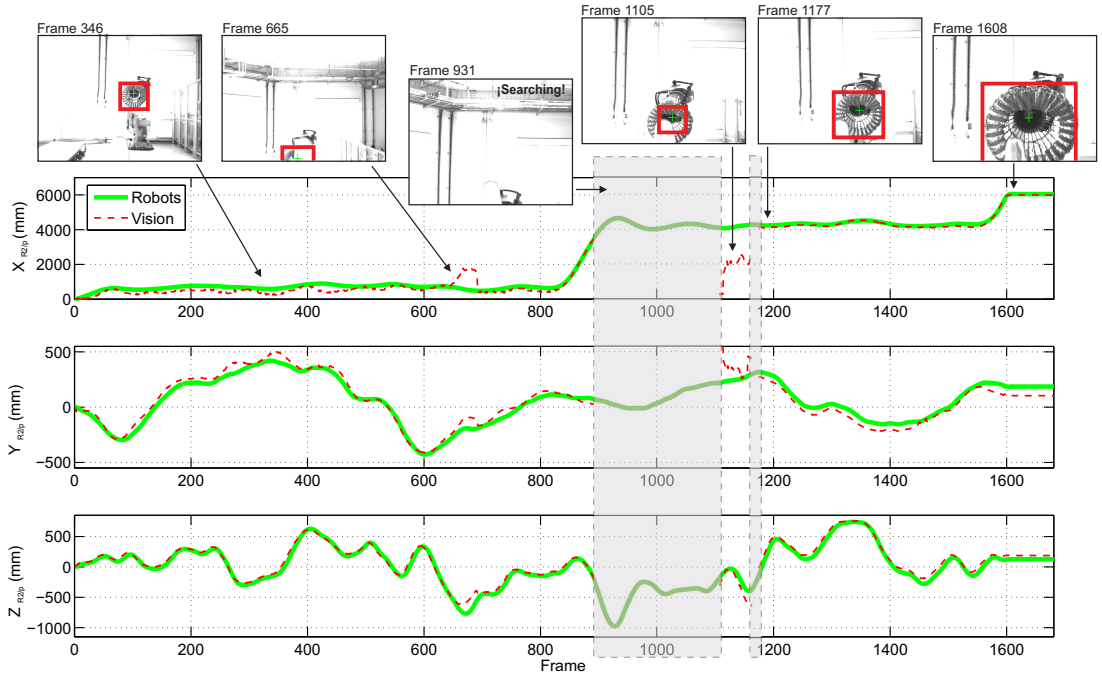


Figure 5.31: Position estimation under medium turbulence conditions. The motion estimated by the tracking algorithm (red/dashed line) is compared with the motion of the R2 robot (green/solid line). The shaded areas represent the moments when the lost status was activated, and the detection algorithm was operating.

is very low (due to the occlusion and the drogue being out of the FOV), which produces an unstable behavior of the tracking algorithm. Nonetheless, the tracking algorithm does not lose the drogue.

From Frame 1105 to Frame 1161, the template matching algorithm was not able to detect the whole extension of the drogue, i.e. in those frames the drogue was misidentified producing a big discrepancy in the estimation of the X_p and Y_p axes, as can be seen in Figure 5.31. In those frames, the size of the misidentified drogue (red/dark box in Frame 1105) is smaller than the real size of the drogue. As a consequence of this, the estimation in the X_p axis (the one related to scale) locates the drogue in a farther position than the one it really is in. Nonetheless, the estimation in the Z axis does not look affected by this misidentification of the drogue. In the final part of the test, from Frame 1172 until Frame1680, the obtained RMSE(s) are 8.6 cm in the X_p axis, 5.7 cm in the Y_p axis, and 6.1 cm in the Z_p axis.

In this test, it can be seen that the vision-based position estimations closely match the positions computed in the robot controller. The basic template matching algorithm operated as intended, successfully locating the drogue when it reentered the field of view of the camera. The scheme relies heavily on the visibility of the central portion of the drogue. Improvements could be made concerning both the accuracy and the recovery time in the presence of partial occlusions by the implementation of a more robust approach.

5.6.3. Discussion

The AAAR testbed shown in Figure 4.31 was used to test the pose estimation algorithm based on the $\mathbf{H_w}$ homography. This algorithm was used to estimate the position of either the probe or the drogue. The accuracy of the estimation was tested by comparing the position data of the robots with the vision-based position data.

The results have shown that the tracking strategy that was adopted for the aerial refuelling tasks enabled the tracking of the drogue in the different tests that include basic motions the two bodies (the receiver and the tanker, i.e. robots R1 and R2) and aerial refuelling procedures under light and moderate turbulence conditions. Additionally, it has been shown that the results of the tracking task have been robust enough to permit a vision-based position estimation of the motion. On these tests, the proposed visual system proved to be robust under adverse conditions, including rapid motions, large changes in proximity and scale, varying orientations of the drogue, and significant occlusions of the drogue by the probe. In these tests, the drogue's position was only lost when it left the FOV of the camera.

From the tests, it was possible to see that by using a tracking strategy based on the information of the drogue (using direct methods) it is possible to continue tracking the drogue when part of it is out of the FOV of the camera, or when it is occluded by the probe. The aforementioned situation was commonly found under light and medium turbulence effects. Therefore, the performance of the presented vision strategy has been shown to be of a standard appropriate to the probe and drogue autonomous aerial refuelling problem. It has low computational overheads for a vision system, and requires no modification of the target. It is particularly relevant to the final approach and contact stage, where changes in orientation are small and the algorithm's strengths in the presence of occlusions and restricted FOV can be exploited. Additional improvements focus on augmenting this algorithm by other sensing technologies.

The average accuracy of the position estimation during the refuelling tests was within 2 cm for the light turbulence conditions and 10 cm for the moderate turbulence test, making it a suitable sensor candidate for probe and drogue refuelling. Additionally, without optimizing the code in any way, real-time frame rates were obtained (> 30 fps), proving that the technique is fast enough concerning the requirements of automated aerial refuelling sensing capabilities.

5.7. Summary

This chapter has introduced two pose estimation algorithms, which depending on the available information can be used to estimate the state of the vehicle or to obtain relative position information. The algorithms are based on decomposing a homography. Concerning the state estimation algorithm, a frame-to-fame homography is estimated and decomposed to obtain frame-to-frame motions. On the other hand, for the relative pose estimation algorithm, the homography that relates points of a plane in the world reference frame with points in the image plane is estimated and then decomposed in order to estimate the relative motion.

The algorithms have been applied in challenging scenarios where robust and real-time

motion estimations are required: state estimation for aerial vehicles, and relative pose estimation for VTOL UAV landing and for autonomous aerial refuelling. In all these applications, the tests show that by using the HMPMR-ICIA tracking algorithm real-time frame rates where achieved in all the applications, and that by using this tracking algorithm with the pose estimation algorithms presented in this chapter it is possible to obtain robust vision-based position and orientation information that was comparable to the one obtained with other sensors (GPS/IMU and robots' data).

Chapter 6

Ground Multi-Camera Pose Estimation

6.1. Introduction

This chapter[1] presents a vision-based algorithm for estimating the position and heading of the UAV using the information recovered by a multi-camera system located on the ground. The algorithm is based on the detection, matching, and 3D reconstruction of key features (color landmarks) placed on-board a UAV.

The multi-camera system is a redundant system composed of three cameras (a trinocular system), that are strategically distributed to take advantage of the cameras' FOV. The system is in charge of recovering the relative 3D position and heading of the UAV with respect to the trinocular system at real-time frame rates. The 3D estimations can be used for conducting vision-based landing tasks, as explained in Chapter 7.

In this thesis, the trinocular system is proposed as a low-cost vision-based platform for autonomous landing tasks. It focuses on obtaining position estimations of the UAV that are more accurate and precise than the ones obtained by GPS information (GPS horizontal and vertical accuracy of 2 m and ± 0.5 m, respectively). Therefore, the proposed system is suitable to complement or replace the GPS-based position estimation in situations where

[1] Publications of the author related to this chapter:
- Trinocular ground system to control UAVs [121]
- On-board and Ground Visual Pose Estimation Techniques for UAV Control [124]

GPS information is unavailable or where its information is inaccurate, allowing the vehicle to develop tasks at low heights.

In many applications, multi-camera systems are considered attractive because of the considerable amount of information that they can recover and due to the increase of the camera FOV. These characteristics can help to solve common vision problems such as occlusions, and can offer more tools for control, tracking, surveillance, and navigation of mobile vehicles, among other tasks. In some cases, the hardware and computational requirements prevent these systems from offering adequate solutions; because the larger the number of cameras used is, the grater the complexity of the system is. Nonetheless, the use of a multiple-camera system becomes necessary when it is required to accurately detect and track multiple objects and to accurately compute their 3D locations.

The pose estimation algorithm presented in this chapter requires the integration of algorithms in areas such as image segmentation, tracking, and 3D-reconstruction techniques. In this chapter, the different techniques that were used are introduced, and results of the proposed multi-camera pose estimation algorithm for UAVs are presented. Real flight tests demonstrate how the multi-camera system improves the position estimation, especially at low heights.

6.2. Trinocular Pose Estimation

The multi-camera system is composed of three cameras physically connected together with overlapping FOVs that simultaneously capture images from different viewpoints. Figure 6.1 depicts the configuration of the trinocular system and the different coordinate systems involved.

State of the art techniques have been implemented to conduct the feature extraction, feature tracking, and 3D reconstruction stages of the pose estimation algorithm. The algorithms have been selected taking into account the real-time requirements of the UAV pose estimation (because it will be used for vision-in-the-loop tasks).

A color-based 3D position estimation algorithm has been implemented to detect on-board landmarks. The algorithm fuses multiple-views information by using a color-based feature extraction algorithm and the triangulation principle to recover the 3D location of on-board color landmarks. Then, the 3D information of each color landmark is used to estimate the position and orientation of the UAV with respect to the world coordinate system, as shown in Figure 6.1.

Different coordinate systems are used to map the extracted visual information and convert it from 2D into 3D: the image coordinate systems that include the lateral (X_f) and the central coordinate system (X_u) in the image plane, the camera coordinate system (X_c), the UAV coordinate system (X_v), and the trinocular world coordinate system (X_w) (see Figure 6.1). In Figure 6.1, the different frames involved are presented, and in Appendix A the camera and the image coordinate systems and their respective transformations are explained in detail.

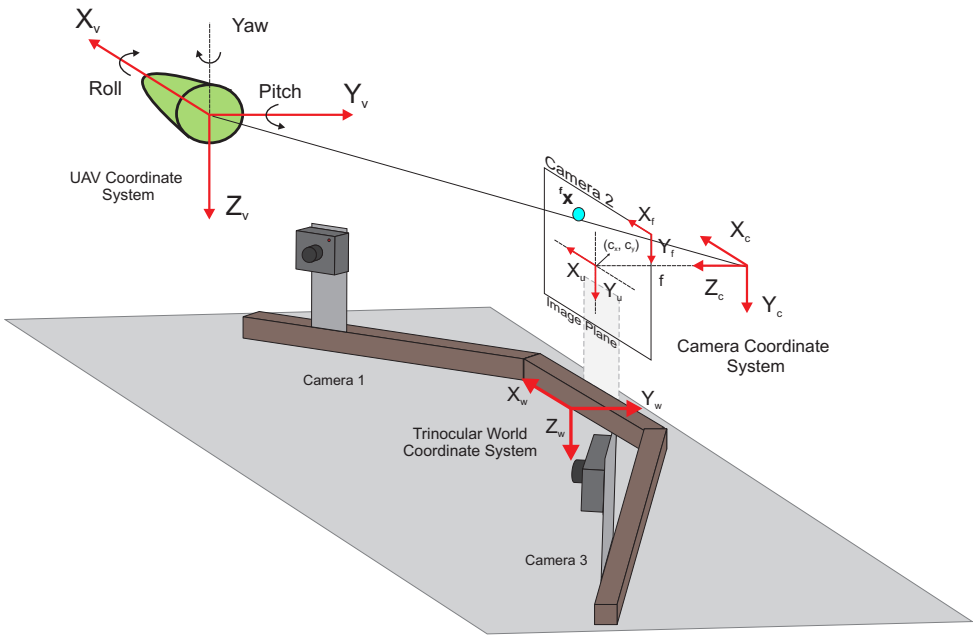


Figure 6.1: Reference frames of the trinocular system. Different coordinate systems are involved in the conversion of the 2D information (in the image plane) found in each camera into 3D information.

6.2.1. Visual Tracking

The color-based tracking algorithm presented in Section 3.2 is applied, as described in this chapter, to extract and track four different color landmarks, which are placed on-board the UAV, as can be seen in Figure 6.2. These landmarks will facilitate the detection of the UAV and also the matching of the features among the different cameras.

As explained in Section 3.2, the algorithm for color detection is based on color histograms. In each camera, the images are converted from the RGB to the HSV color space. The Hue channel of the HSV color space is used to create model histograms of the colors to be tracked and also to create the histogram of each frame of the image sequence. The relation between the histogram of an image and the model histograms (of each color) is found calculating a ratio histogram: $\mathbf{Rh}_i^k$. This ratio histogram is backprojected onto the image, and the pixels' values of the resulting image (i.e. the backprojected image) represent the probability that each pixel belongs to the color we are looking for. This backprojection algorithm (explained in Section 3.2) is used to segment the different landmarks that are on-board.

Figure 6.2 shows the results of the backprojection algorithm applied to images of the UAV recovered by the trinocular system. In that figure, it can be seen that the histogram of the current image (left histogram) is compared with a model histogram (right image), e.g. the one with the blue color. The result of this comparison is the backprojected image (lower image) with the segmented object. This backprojected image contains all the objects that have the same color as the one of the model histogram (i.e. blue objects in the example being discussed).

The location of the landmarks in the different frames is found by using the previously

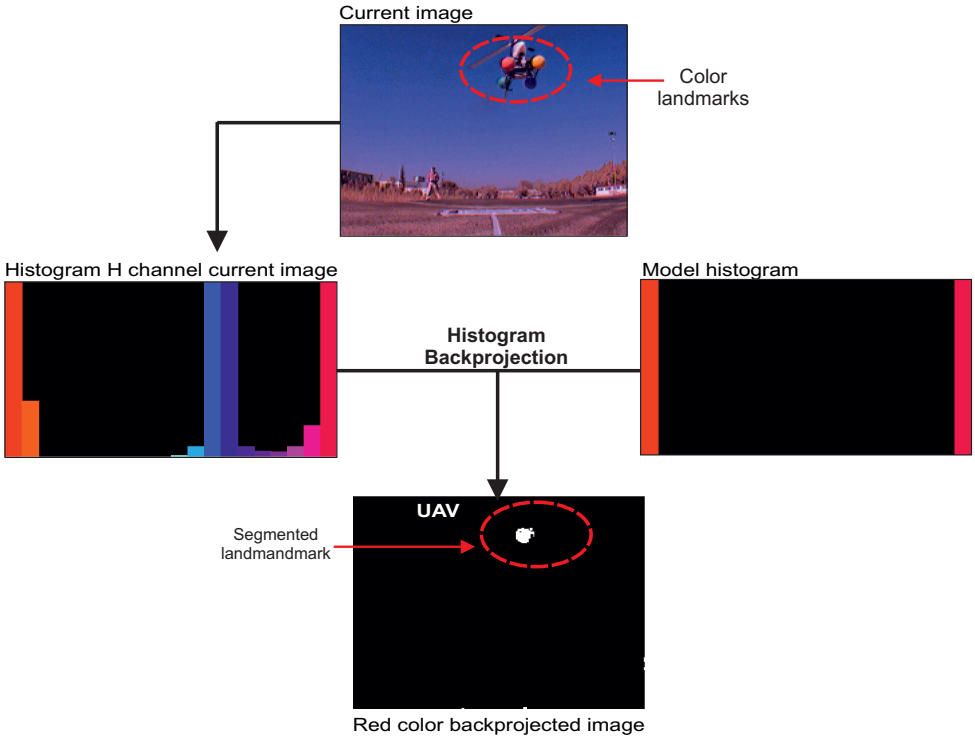


Figure 6.2: Backprojection algorithm for detecting on-board landmarks. Landmarks located on-board the UAV are detected by comparing histograms of the Hue channel. The lower image is the result of the backprojection algorithm.

mentioned algorithm; and the CAMSHIFT algorithm, explained in Section 3.2.2, is then used to track the landmarks in the different frames. The CAMSHIFT takes the back-projected image $\mathbf{B}_i^k$ created for each landmark i in each camera k, and moves a search window (previously initialized) iteratively in order to find the densest region (the peak), which will correspond to the landmark i that is looked for.

The centroid $\mathbf{x_c}_i^k$ of each feature in the different images are then used as features for the 3D reconstruction stage. These centroids are found as shown in Equation (3.3), using the information contained inside the search window found by the CAMSHIFT algorithm. Figure 6.3 shows some results of the algorithms tracking the color landmarks located on-board the UAV, and their respective centroids.

The 3D reconstruction process requires the relation among the information found by the different cameras to be found. When working with different cameras with overlapping FOVs, the matching process requires not only the differentiation of features in the same image, but also the definition of a metric, which tells whether the feature i found in the image of camera $\mathbf{I}^1$ is the same feature i found in $\mathbf{I}^2$ (where the superscript represents the number of the camera). Although this is a critical process, in this application the feature matching process among the cameras has been facilitated taking into account the color information of the different landmarks. Therefore, the features (i.e. the central moments of each landmark) are matched by grouping the characteristics with the same color that are seen simultaneously by more than two cameras. These matched centroids are then used for the 3D reconstruction stage.

Figure 6.3: Feature extraction results. The CAMSHIFT algorithm is in charge of tracking color-landmarks on-board the UAVs.

6.2.2. Pose Estimation

Once the features in each camera are extracted and matched, the information of the features in the image plane of each camera has to be transformed into a common coordinate system in order to find the 3D position of each landmark. To achieve this, the trinocular system must be calibrated, i.e. intrinsic and extrinsic parameters must be known.

The extrinsic parameters of the trinocular system (rotations and translations) permit to relate the information found in the different cameras, as can be seen in Figure 6.4. These parameters ($[{}^{c_2}\mathbf{R}_{c_3}, {}^{c_2}\mathbf{t}_{c_3}]$,$[{}^{c_2}\mathbf{R}_{c_1}, {}^{c_2}\mathbf{t}_{c_1}]$) and the intrinsic parameters $\mathbf{K}^k$ of each camera (where $k = \{1, 2, 3\}$) are found through an off-line calibration process. With this information, the 3D position of the matched landmarks can be recovered by intersecting the backprojection of the rays from the different cameras that have seen the same landmark in the 3D space. In Figure 6.4, it can be seen that by intersecting the backprojected rays of the points ${}^{f}\mathbf{p}_i^k$ seen in the three cameras k, it is possible to obtain the 3D position of the point (${}^{w}\mathbf{p}_i$). The 3D reconstruction process is explained in the following paragraphs.

Considering that the cameras are fixed on the ground, and assuming that a world coordinate system (different from the trinocular world coordinate system) is located at the camera 2 coordinate system, as shown in Figure 6.4 (cyan/light line), then the cameras and the world coordinate systems can be related by rigid transformations, which are estimated using the camera extrinsic parameters of the trinocular system. Therefore, the 3D coordinates of each landmark defined in the world coordinate system can be defined in each camera frame k, as follows:

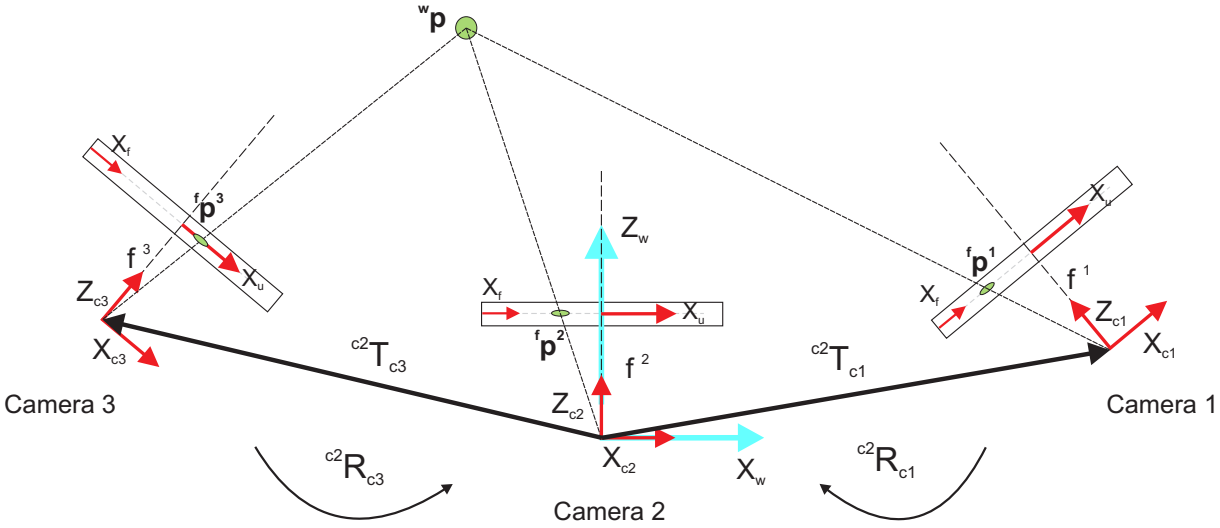


Figure 6.4: Trinocular 3D reconstruction. The extrinsic ($[{}^{c2}\mathbf{R}_{c_3}, {}^{c2}\mathbf{t}_{c_3}]$,$[{}^{c2}\mathbf{R}_{c_1}, {}^{c2}\mathbf{t}_{c_1}]$) and intrinsic parameters $\mathbf{K}^k$ are used to calculate the 3D position of the landmarks with respect to a world coordinate system located at the camera 2 coordinate system.

$$ {}^{c}\mathbf{x}_i^k = \begin{bmatrix} {}^{c_k}\mathbf{R}_w & {}^{c_k}\mathbf{t}_w \\ \mathbf{0}^T & 1 \end{bmatrix} {}^{w}\mathbf{x}_i \tag{6.1} $$

Where ${}^{c}\mathbf{x}_i^k$ are the coordinates of landmark i expressed in the camera coordinate system of camera k; ${}^{w}\mathbf{x}_i$, are the coordinates of landmark i expressed in the world coordinate system; and $[{}^{c_k}\mathbf{R}_w|{}^{c_k}\mathbf{t}_w]$ are the extrinsic parameters that relate the world and the camera coordinate systems. Because it is assumed that for the 3D reconstruction stage the world coordinate system is located in the camera 2 coordinate system, those extrinsic parameters can be related to the extrinsic parameters of the trinocular system shown in Figure 6.4.

The centroids of each landmark found by the visual tracking algorithm in each image $\mathbf{x}_{\mathbf{c}i}^k$ are defined in the lateral coordinate system (i.e. in the image plane) of each camera, i.e. ${}^{f}\mathbf{x}_{\mathbf{c}i}^{k}$ (where the left upper script f represents the lateral coordinate system in the image plane), and taking from the pinhole camera model the following relations (Appendix A):

$$ \begin{gathered} {}^{u}x_i^k = {}^{f}x_{c_i}^k - c_x^k \;\;,\;\; {}^{u}y_i^k = {}^{f}y_{c_i}^k - c_y^k \\ \frac{{}^{u}x_i^k}{f_x^k} = \frac{{}^{c}x_i^k}{{}^{c}z_i^k} \;\;,\;\; \frac{{}^{u}y_i^k}{f_y^k} = \frac{{}^{c}y_i^k}{{}^{c}z_i^k} \end{gathered} \tag{6.2} $$

where ${}^{u}x_i^k$ and ${}^{u}x_i^k$ represent the coordinates of landmark i expressed in the central camera coordinate system of each camera k; ${}^{f}x_{c_i}^k$ and ${}^{f}y_{c_i}^k$ are the coordinates of the center of gravity of each landmark in the image plane (i.e. the lateral coordinate system); c_x^k and c_y^k are the coordinates of the center of projection in pixel units; and f_x^k and f_y^k are the focal lengths in terms of pixel dimensions.

Then, a 3D point can be related with its projection in each camera using Equations 6.1 and 6.2, as follows, assuming that $\mathbf{R}^i = {}^{c_k}\mathbf{R}_w$ and that $\mathbf{t}^i = {}^{c_k}\mathbf{t}_w$:

$$
\begin{aligned}
{}^{\mathrm{u}}x_i^1 &= f_{\mathrm{x}}^1 \frac{r_{11}^1 {}^{\mathrm{w}}x_i + r_{12}^1 {}^{\mathrm{w}}y_i + r_{13}^1 {}^{\mathrm{w}}z_i + t_x^1}{r_{31}^1 {}^{\mathrm{w}}x_i + r_{32}^1 {}^{\mathrm{w}}y_i + r_{33}^1 {}^{\mathrm{w}}z_i + t_z^1} & {}^{\mathrm{u}}y_i^1 &= f_{\mathrm{y}}^1 \frac{r_{21}^1 {}^{\mathrm{w}}x_i + r_{22}^1 {}^{\mathrm{w}}y_i + r_{23}^1 {}^{\mathrm{w}}z_i + t_y^1}{r_{31}^1 {}^{\mathrm{w}}x_i + r_{32}^1 {}^{\mathrm{w}}y_i + r_{33}^1 {}^{\mathrm{w}}z_i + t_z^1} \\
{}^{\mathrm{u}}x_i^2 &= f_{\mathrm{x}}^2 \frac{r_{11}^2 {}^{\mathrm{w}}x_i + r_{12}^2 {}^{\mathrm{w}}y_i + r_{13}^2 {}^{\mathrm{w}}z_i + t_x^2}{r_{31}^2 {}^{\mathrm{w}}x_i + r_{32}^2 {}^{\mathrm{w}}y_i + r_{33}^2 {}^{\mathrm{w}}z_i + t_z^2} & {}^{\mathrm{u}}y_i^2 &= f_{\mathrm{y}}^2 \frac{r_{21}^2 {}^{\mathrm{w}}x_i + r_{22}^2 {}^{\mathrm{w}}y_i + r_{23}^2 {}^{\mathrm{w}}z_i + t_y^2}{r_{31}^2 {}^{\mathrm{w}}x_i + r_{32}^2 {}^{\mathrm{w}}y_i + r_{33}^2 {}^{\mathrm{w}}z_i + t_z^2} \\
{}^{\mathrm{u}}x_i^3 &= f_{\mathrm{x}}^3 \frac{r_{11}^3 {}^{\mathrm{w}}x_i + r_{12}^3 {}^{\mathrm{w}}y_i + r_{13}^3 {}^{\mathrm{w}}z_i + t_x^3}{r_{31}^3 {}^{\mathrm{w}}x_i + r_{32}^3 {}^{\mathrm{w}}y_i + r_{33}^3 {}^{\mathrm{w}}z_i + t_z^3} & {}^{\mathrm{u}}y_i^3 &= f_{\mathrm{y}}^3 \frac{r_{21}^3 {}^{\mathrm{w}}x_i + r_{22}^3 {}^{\mathrm{w}}y_i + r_{23}^3 {}^{\mathrm{w}}z_i + t_y^3}{r_{31}^3 {}^{\mathrm{w}}x_i + r_{32}^3 {}^{\mathrm{w}}y_i + r_{33}^3 {}^{\mathrm{w}}z_i + t_z^3}
\end{aligned}
\tag{6.3}
$$

Reorganizing Equation 6.3, for each each camera k that sees a landmark i, we have two equations and three unknowns (the 3D position of landmark i, ${}^{\mathrm{w}}\mathbf{x}_i$) with the following form:

$$
\begin{aligned}
({}^{\mathrm{u}}x_i^k r_{31}^k - f_{\mathrm{x}}^k r_{11}^k){}^{\mathrm{w}}x_i + ({}^{\mathrm{u}}x_i^k r_{32}^k - f_{\mathrm{x}}^k r_{12}^k){}^{\mathrm{w}}y_i + ({}^{\mathrm{u}}x_i^k r_{33}^k - f_{\mathrm{x}}^k r_{13}^k){}^{\mathrm{w}}z_i = \\
(f_{\mathrm{x}}^k t_x^k - {}^{\mathrm{u}}x_i^k t_z^k) \\
({}^{\mathrm{u}}y_i^k r_{31}^k - f_{\mathrm{y}}^k r_{21}^k){}^{\mathrm{w}}x_i + ({}^{\mathrm{u}}y_i^k r_{32}^k - f_{\mathrm{y}}^k r_{22}^k){}^{\mathrm{w}}y_i + ({}^{\mathrm{u}}y_i^k r_{33}^k - f_{\mathrm{y}}^k r_{23}^k){}^{\mathrm{w}}z_i = \\
(f_{\mathrm{y}}^k t_y^k - {}^{\mathrm{u}}y_i^k t_z^k)
\end{aligned}
\tag{6.4}
$$

Applying some restrictions regarding the minimum number of cameras that see a specific landmark, if there are at least two cameras seeing the same landmark, then it is possible to obtain a system of equations in the form $\mathbf{A}_i\mathbf{c}_i = \mathbf{b}_i$. Using least squares method, this over-determined system of equations can be solved as follows:

$$
\mathbf{c}_i \approx \mathbf{A}_i^{+}\mathbf{b}_i = (\mathbf{A}_i^{\mathrm{T}}\mathbf{A}_i)^{-1}\mathbf{A}_i^{\mathrm{T}}\mathbf{b}_i \tag{6.5}
$$

where:

$$
\mathbf{A}_i = \begin{bmatrix}
({}^{\mathrm{u}}x_i^1 r_{31}^1 - f_{\mathrm{x}}^1 r_{11}^1) & ({}^{\mathrm{u}}x_i^1 r_{32}^1 - f_{\mathrm{x}}^1 r_{12}^1) & ({}^{\mathrm{u}}x_i^1 r_{33}^1 - f_{\mathrm{x}}^1 r_{13}^1) \\
({}^{\mathrm{u}}y_i^1 r_{31}^1 - f_{\mathrm{y}}^1 r_{21}^1) & ({}^{\mathrm{u}}y_i^1 r_{32}^1 - f_{\mathrm{y}}^1 r_{22}^1) & ({}^{\mathrm{u}}y_i^1 r_{33}^1 - f_{\mathrm{y}}^1 r_{23}^1) \\
\vdots & \vdots & \vdots \\
({}^{\mathrm{u}}x_i^k r_{31}^k - f_{\mathrm{x}}^k r_{11}^k) & ({}^{\mathrm{u}}x_i^k r_{32}^k - f_{\mathrm{x}}^k r_{12}^k) & ({}^{\mathrm{u}}x_i^k r_{33}^k - f_{\mathrm{x}}^k r_{13}^k) \\
({}^{\mathrm{u}}y_i^k r_{31}^k - f_{\mathrm{y}}^k r_{21}^k) & {}^{\mathrm{u}}y_i^k r_{32}^k - f_{\mathrm{y}}^k r_{22}^k) & ({}^{\mathrm{u}}y_i^k r_{33}^k - f_{\mathrm{y}}^k r_{23}^k)
\end{bmatrix}
$$

$$
\mathbf{b}_i = \begin{bmatrix}
(f_{\mathrm{x}}^1 t_x^1 - {}^{\mathrm{u}}x_i^1 t_z^1) \\
(f_{\mathrm{y}}^1 t_y^1 - {}^{\mathrm{u}}y_i^1 t_z^1) \\
\vdots \\
(f_{\mathrm{x}}^k t_x^k - {}^{\mathrm{u}}x_i^k t_z^k) \\
(f_{\mathrm{x}}^k t_y^k - {}^{\mathrm{u}}y_i^k t_z^k)
\end{bmatrix}
$$

$$
\mathbf{c}_i = \begin{bmatrix}
{}^{\mathrm{w}}x_i \\
{}^{\mathrm{w}}y_i \\
{}^{\mathrm{w}}z_i
\end{bmatrix}
$$

The solution $\mathbf{c}_i$ corresponds to the 3D coordinates $({}^{\mathrm{w}}x_i, {}^{\mathrm{w}}y_i, {}^{\mathrm{w}}z_i)$ of landmark i found in the world reference frame that was assumed to be the camera 2 coordinate system.

However, taking into account that the trinocular world coordinate system is located on the ground, as can be seen in Figure 6.5, then the estimated 3D position of each landmark has to be transformed to the trinocular world coordinate system. Because the cameras are mounted on a fixed-tilted platform in order to take advantage of the cameras' FOV (see Figure 6.5), the transformation is conducted in two steps. The first transformation transforms the 3D points from the camera 2 coordinate system (the red/dark coordinate system that was selected as world coordinate system in the 3D reconstruction stage) to the cyan/light coordinate system shown in Figure 6.5, as follows:

$$ {}^{w'}\mathbf{x}_i = {}^{w'}\mathbf{R}_{c_2}\,{}^{c_2}\mathbf{x}_i \tag{6.6} $$

Where the rotation matrix ${}^{w'}\mathbf{R}_{c_2}$ has been found using the camera calibration toolbox [23], as follows: different images of a calibration pattern located in front of the camera have been captured and used to find the extrinsic parameters of each image, i.e. the orientation of the camera coordinate 2 system with respect to the cyan/light coordinate system shown in Figure 6.5.

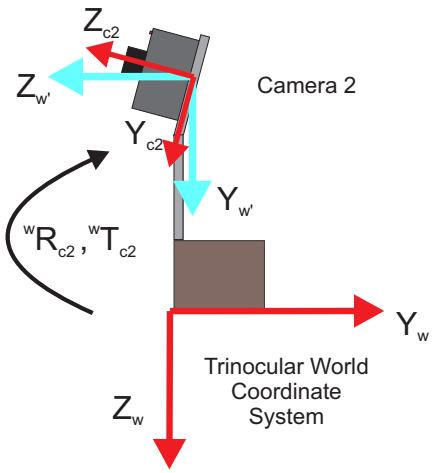


Figure 6.5: The trinocular world coordinate system. The transformation from the camera 2 coordinate system to the trinocular world coordinate system is conducted in two steps.

The second transformation, as can be seen in Figure 6.5, relates the cyan/light coordinate system with the trinocular world coordinate system (red/dark solid line). The transformation is defined by a rotation of 90° around the $X_{w'}$ axis, followed by a translation along the $Z_{w'}$ axis. The translation vector ${}^{w}\mathbf{t}_{w'}$ is found experimentally, based on the 3D coordinates of one of the landmarks extracted by the trinocular system (without including this last transformation), and taking into account that the diameter of the landmark is known. Therefore, knowing the position of the center of gravity of the landmark with respect to the floor (the radio), and knowing the position of the center of gravity with respect to the cyan/light coordinate system shown in Figure 6.5, it is possible to estimate ${}^{w}\mathbf{t}_{w'}$.

All the previous transformations let us express the 3D coordinates of each landmark with respect to the trinocular world coordinate system located on the ground. With this

information, the UAV's position and orientation (θ, i.e. the yaw angle) are determined taking into account the landmarks' distribution around the UAV coordinate system, assuming that the positions of the landmarks ${}^{\mathrm{v}}\mathbf{x}_i$ with respect to the UAV coordinate system are known, as shown in Figure 6.6.

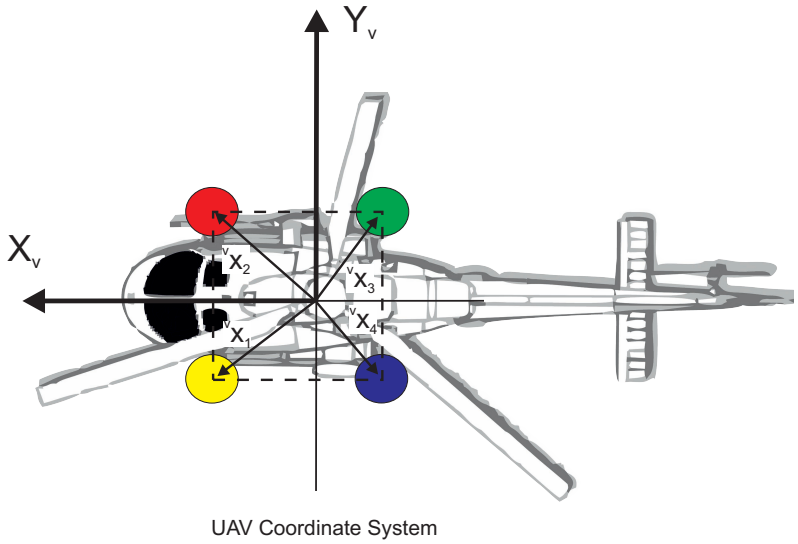


Figure 6.6: Distribution of landmarks in the UAV reference frame. The distribution of the landmarks in the UAV coordinate system is a known parameter used to extract the UAV position and orientation with respect to the trinocular world coordinate system.

The relation between the UAV coordinate system and the trinocular world coordinate system depends on a translation vector that defines the UAV's position, and on a rotation matrix that defines the orientation of the UAV (pitch, roll and yaw angles). For practical purposes and assuming that the UAV will move at low velocities, pitch and roll angles are considered equal to 0, and only the yaw angle (θ) is taken into account in order to send the adequate commands to the UAV.

Therefore, the UAV and the trinocular coordinate systems are related as follows:

$$ {}^{\mathrm{w}}\mathbf{x}_i = \begin{bmatrix} {}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}} & {}^{\mathrm{w}}\mathbf{t}_{\mathrm{v}} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} {}^{\mathrm{v}}\mathbf{x}_i \tag{6.7} $$

Where ${}^{\mathrm{w}}\mathbf{x}_i$ are the 3D coordinates of each landmark i with respect to the trinocular world coordinate system; ${}^{\mathrm{v}}\mathbf{x}_i$, are the coordinates of landmark i expressed in the world coordinate system, ${}^{\mathrm{w}}\mathbf{R}_{\mathrm{v}}$ is the rotation matrix defined around the $\mathrm{Z_v}$ axis (only yaw is estimated) and ${}^{\mathrm{w}}\mathbf{t}_{\mathrm{v}}$ is the translation vector that represents the position of the UAV with respect to the trinocular world coordinate system.

Formulating Equation (6.7) for each reconstructed landmark, we have three equations and four unknowns (the translation vector ${}^{\mathrm{w}}\mathbf{t}_{\mathrm{v}}$, and the yaw angle θ). Additionally, applying the restriction that the 3D reconstruction of at least two features is required, then the position and orientation of the UAV can be determined using Equation (6.7).

Therefore, considering that $\mathrm{c}\theta = \cos(\theta)$, $\mathrm{s}\theta = \sin(\theta)$, and formulating Equation 6.7 for all the landmarks detected, it is possible to create a system of equations with five unknowns: $\mathrm{c}\theta, \mathrm{s}\theta, {}^{\mathrm{w}}x, {}^{\mathrm{w}}y, {}^{\mathrm{w}}z$, as follows:

$$\mathbf{Ac} = \mathbf{b}$$

$$\begin{bmatrix} {}^{\mathrm{v}}x_1 & -{}^{\mathrm{v}}y_1 & 1 & 0 & 0 \\ {}^{\mathrm{v}}y_1 & {}^{\mathrm{v}}x_1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ {}^{\mathrm{v}}x_i & -{}^{\mathrm{v}}y_i & 1 & 0 & 0 \\ {}^{\mathrm{v}}y_i & {}^{\mathrm{v}}x_i & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta \\ \mathrm{s}\theta \\ {}^{\mathrm{w}}t_{x\mathrm{v}} \\ {}^{\mathrm{w}}t_{y\mathrm{v}} \\ {}^{\mathrm{w}}t_{z\mathrm{v}} \end{bmatrix} = \begin{bmatrix} {}^{\mathrm{w}}x_1 \\ {}^{\mathrm{v}}y_1 \\ {}^{\mathrm{w}}z_1 - {}^{\mathrm{v}}z_1 \\ \vdots \\ {}^{\mathrm{w}}x_i \\ {}^{\mathrm{w}}y_i \\ {}^{\mathrm{w}}z_i - {}^{\mathrm{v}}z_i \end{bmatrix} \tag{6.8}$$

This system of equations can be solved as shown in Equation 6.5, and the solution ($\mathbf{c}$) is a 5×1 vector whose components define the orientation (θ angle) and the position of the UAV, both expressed with respect to the trinocular world coordinate system.

6.3. Results: Helicopter's Pose Estimation

In this thesis, the multi-camera system is proposed to deal with the position estimation problem at low heights. In this scenario, tasks such as landing require more accurate and precise position estimations of the UAV than the ones obtained with a GPS (GPS horizontal and vertical accuracy of 2 m and ± 0.5 m respectively).

Different experiments were conducted with the aim of validating the proposed algorithm regarding feature extraction, tracking, and position and orientation estimation. The trinocular system is tested outdoors in real flight tests. In order to evaluate the results, the behavior of the vision-based estimations is compared with manually-measured data and with the UAV's state information estimated with other on-board sensors (the GPS/IMU estimation).

6.3.1. Experimental Setup

To conduct the tests, color landmarks were placed on-board the Rotomotion SR-20 electric helicopter shown in Figure 6.7. The helicopter's state data is sent to the ground station to compare the results obtained with the vision-based pose estimation algorithm.

On the other hand, the trinocular system, as shown in Figure 6.7, is composed of three FireWire cameras with 3.4 mm lenses (horizontal FOV $\approx 87°$, and vertical FOV $\approx 71°$). Each camera captures images of 320×240 size at 30 FPS. As can be seen in Figure 6.7, the cameras are strategically located on an aluminum platform in such a way that it is possible to take advantage of the cameras' field of view. The relative position and orientation of the cameras are fixed, and their intrinsic and extrinsic parameters are known due to a calibration process [22]. The cameras are connected to a laptop running Linux OS, where the vision algorithm operates. The program was developed in C++, and the OpenCV libraries were used for image processing.

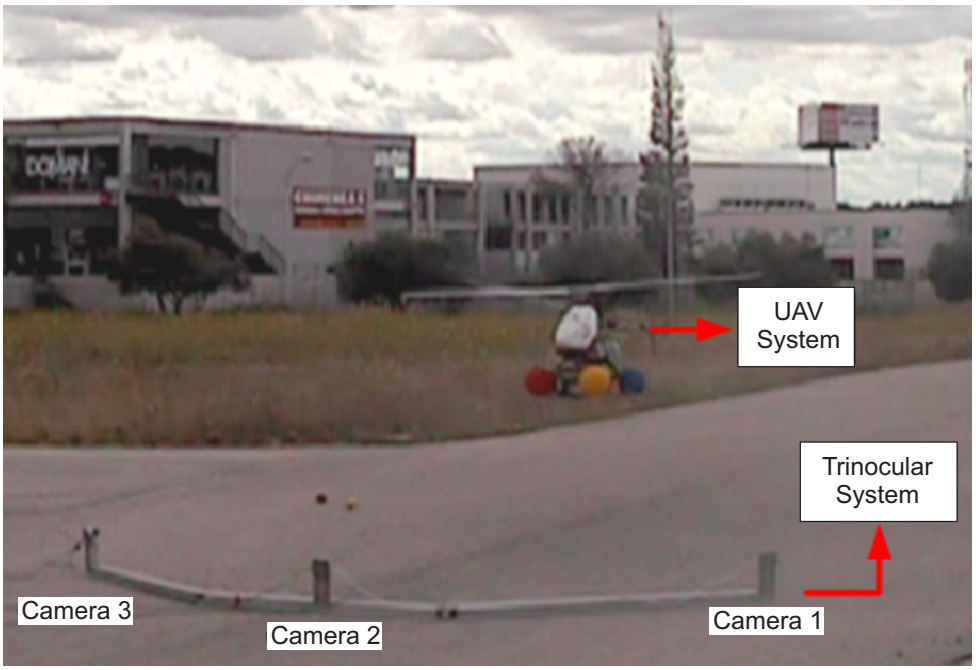


Figure 6.7: Trinocular system and helicopter testbed (Colibri III) during a flight test. Color landmarks on-board the UAV are used as key features for the pose estimation algorithm. The trinocular system is composed of three FireWire cameras that capture images of 320×240 size at 30 FPS.

6.3.2. Static Tests

The experiments described in this section consist in positioning the helicopter at different known 3D coordinates with respect to the trinocular world coordinate system. Approximated real values were obtained by manually measuring the 3D positions of the UAV. The Root Mean Square Error (RMSE) of the helicopter's positions is calculated by comparing the pose estimation results with the known 3D positions. Figure 6.8 presents the RMSE obtained when the helicopter was placed in front of the trinocular system, at different positions. The results show a typical behavior of a 3D reconstruction based on triangulation. As the helicopter moves farther away from the trinocular system, the errors in the Y_w axis increase. This is so because, at those positions, the parallax angle within the rays becomes small. On the other hand, in the figure it can be seen that the errors in the Z_w and X_w axes are satisfactory and constant within a small range, compared with the ones obtained in the Y_w direction. This happens because changes in the X_w and Y_w axes can be better perceived in the image plane.

The errors found in this test give an idea of the precision of the system, which for the X_w axis is ≈ 5 cm, for the Z_w axis is ≈ 5 cm, and for the Y_w axis the error depends on the depth. For a landing case, the depth will be ≈ 3 m, which corresponds to a Y_w precision of ≈ 10 cm. This obtained precision satisfies the requirements of the task to accomplish (landing and positioning).

The algorithm was also compared with the information obtained by the helicopter's state estimator (the output after a Kalman Filter) when the helicopter was in a specific

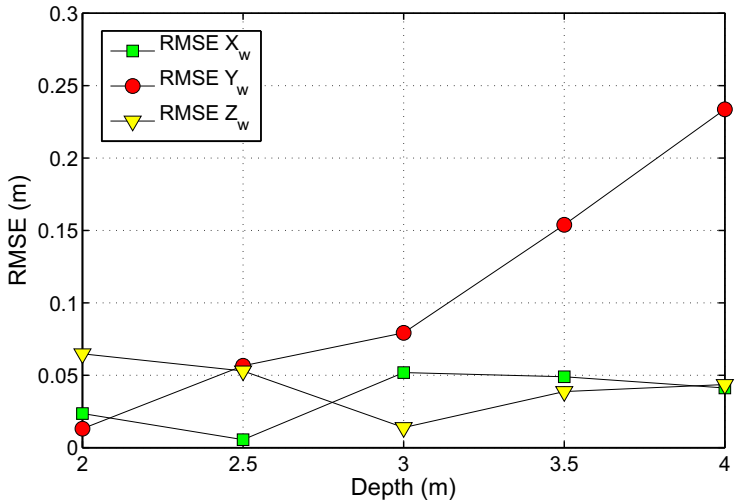


Figure 6.8: Pose estimation evaluation. RMSEs obtained by comparing the pose estimation results with known 3D positions at different depths.

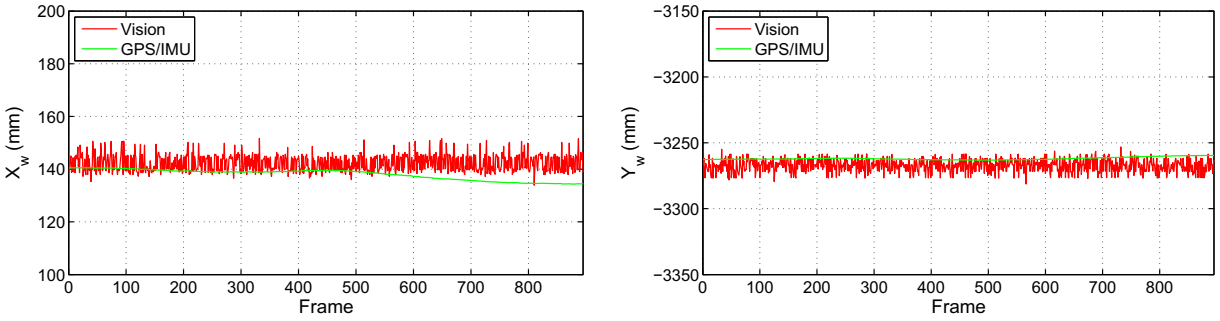
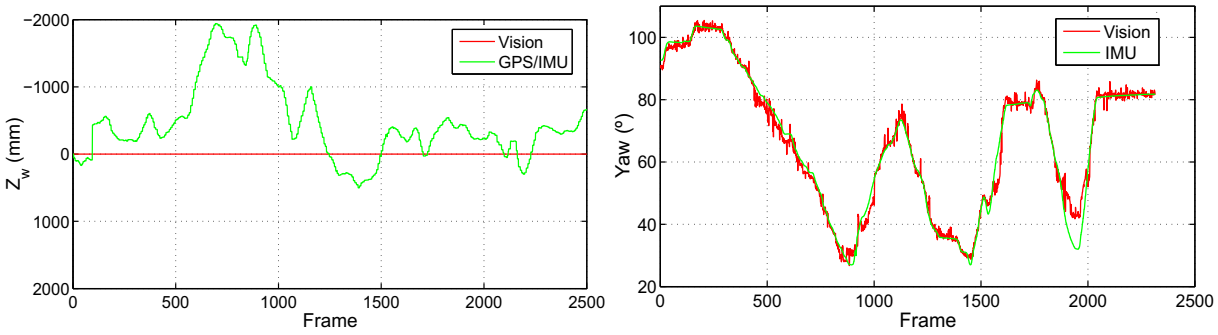


Figure 6.9: Comparison of the X_w and Y_w axes with the GPS/IMU data. The state values given by the helicopter state estimator (green/light line) are compared with the raw data estimated by the trinocular system (red/dark line).

position on the floor. Figures 6.9 and 6.10 show the comparison of the vision-based estimation (red/dark solid line) with the GPS/IMU data (green/light solid line). As can be seen in the results shown in Figure 6.9, the vision-based position estimations in the X_w and Y_w axes correspond with the ones measured by the helicopter's state estimation. It is important to notice that the vision data has not been filtered.

Figure 6.10(a) shows the comparison of the estimated altitudes (Z_w axis). In this figure, it is possible to see how the estimation of the helicopter's height can be improved by the vision system, taking into account that the estimation of the vehicle's height based on GPS information has an accuracy of ± 0.5 m (or a lower accuracy as it approaches the ground). In Figure 6.10(a), it can be seen that when the helicopter is on the ground, the GPS/IMU altitude estimation shows the UAV flying.

On the other hand, Figure 6.10(b) shows the comparison of the estimated yaw values (rotation around the Z_w axis) when the helicopter rotates clockwise and counterclockwise around the Z_w axis. In this figure, it is possible to see the similar behavior and values of the data. The obtained error was RMSE $= 9.12°$, taking the measurement given by the helicopter's state estimator as ground truth.



(a) UAV's altitude. Close to the ground GPS accuracy decreases

(b) Yaw angle. The data have similar behavior and values. The RMSE = 9.12°.

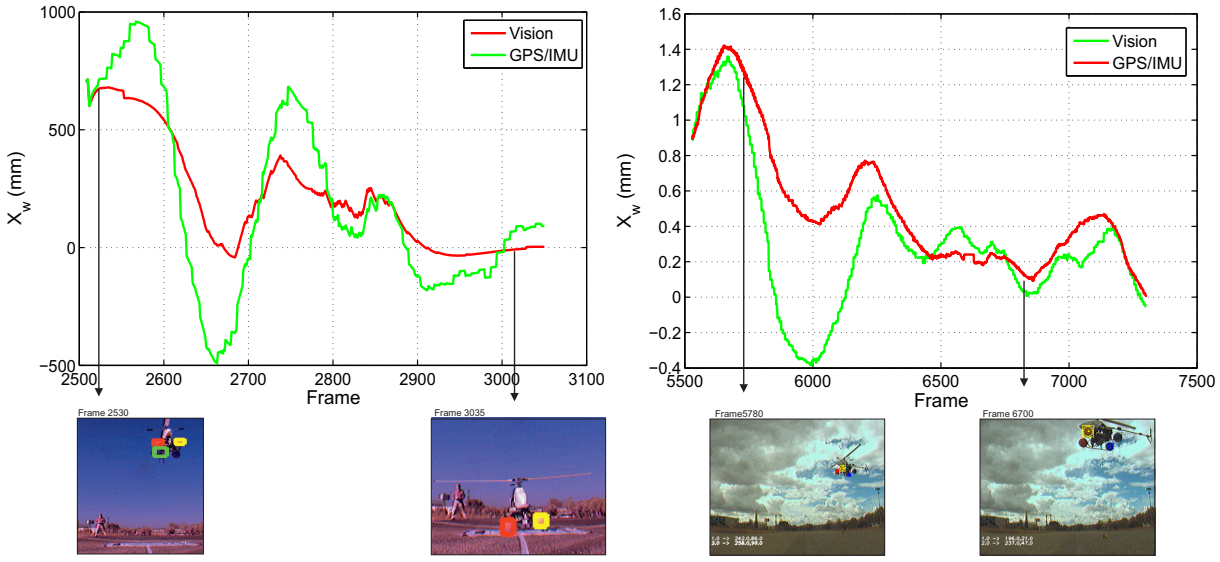
Figure 6.10: Comparison of the Z_w axis and yaw angle estimations with the GPS/IMU data. In Figure 6.10(a), it can be seen that the height estimation can be improved by the vision system (red/dark line).

6.3.3. Flight Tests

The algorithm has been tested during different flights (the video sequences can be found in [52]). Images have been captured and processed off-line. In this section, results from two different flights are used to analyzed the performance of the trinocular system for estimating the position of the UAV. The first test corresponds to a landing task which has been performed in manual-mode (flown by the pilot), and the second test corresponds to the UAV moving in front of the trinocular system. Results from an additional test are used to analyze the performance of the trinocular system estimating the yaw angle.

Figure 6.11 shows the estimated values for the X_w axis in the two tests that were conducted: Figure 6.11(a) shows the results for the landing test and Figure 6.11(b) shows the results of the second flight test. In both figures, it can be seen that for the X_w axis the vision based estimations (red/dark solid lines) and the GPS/IMU estimations (green/light solid lines) have similar behaviors in almost all the flight tests, i.e. when the helicopter moves to the left (e.g. see first thumbnail image of Figure 6.11(a)). Both estimations reflect this motion. However, the magnitude of the motion is different. The position accuracy of the GPS/IMU estimations is ± 1 m, and close to the ground this accuracy decreases. For this reason, these estimations are not considered as ground truth and are only used for analyzing the behavior of the estimations.

In order to obtain an idea of the performance of the algorithm, the images that were captured are analyzed. Analyzing the first thumbnail image shown in Figure 6.11(a), it can be seen that at the beginning the helicopter was located at the right side (positive side) of the trinocular world coordinate system (the center of the image approximately coincides with the origin of that coordinate system). During this test, the UAV was always moving between the center and the right side of the image, and when the helicopter landed, its position was at the center of the image, which means that the position estimated at that point should be ≈ 0, as can be seen in the thumbnail image on the right. Therefore, analyzing Figure 6.11(a), it can be seen that the motions described above coincide with the the motion estimated by the vision system. The vision-based estimations show that



(a) Test1 manual landing.

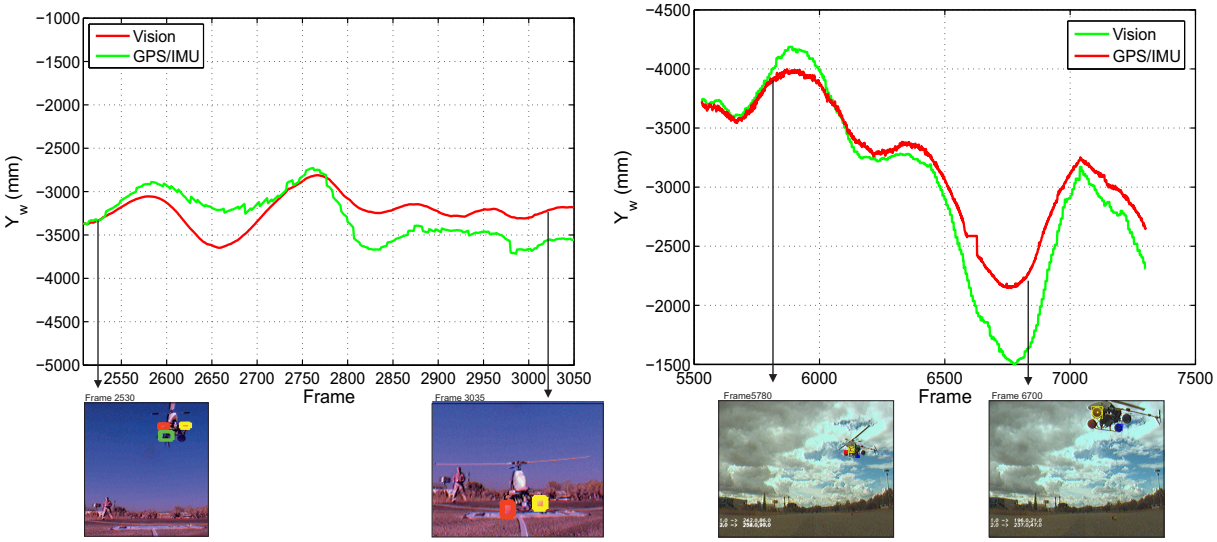
(b) Test 2 manual flight.

Figure 6.11: Position estimations in the X_w axis. The vision-based estimations (red/dark line) are compared with the ones estimated by the UAV's state estimator (based on GPS/IMU, green/light line). Thumbnail images permit to analyze the performance of the vision system.

the UAV was moving at the right side of the trinocular world coordinate system. The same analysis conducted in the second test also shows that the vision-based estimations coincide with the motions the UAV conducted.

Figure 6.12 shows the estimation in the Y_w axis for both tests. In the figures, it can be seen that the vision-based estimations have a behavior that is similar to the one of the GPS/IMU estimations. Additionally, the thumbnail images permit to confirm that the behavior of the vision-based estimated motion coincides with the motion the UAV conducted. In Figure 6.12(a), when the UAV landed (see Frames 300-3050), it landed over an object that was in front of the trinocular system. The distance between the object an the trinocular system was manually measured ≈ 3 m. Comparing this value with the one estimated by the vision system (red/dark solid line), it can be seen that the estimated position is close to the real-one. On the other hand, analyzing the thumbnail images of Figure 6.12(b), it can be seen that at the beginning of the test the UAV was far away from the trinocular system, and at the end it got closer. This behavior also corresponds with the one described by the vision-based estimations (red/dark solid line).

On the other hand, Figure 6.13 shows the results of the altitude estimations for the two tests that are being analyzed. In none of the tests the altitudes estimated by the GPS/IMU have values that are similar to the ones estimated by the trinocular system. However, the behavior of the estimated motions seems to be similar, i.e. if the UAV goes up, the estimated values increase. Analyzing the captured images, especially for the landing test, it can be seen that the behavior of the vision-based estimations coincides with the one conducted by the UAV. For example, in Figure 6.13(a), when the helicopter has landed (Frames 3000-3050), the vision system shows an estimation of the altitude ≈ 0 (red/dark solid line), whereas according to the GPS/IMU estimations (green/light solid



(a) Test1 manual landing.

(b) Test 2 manual flight.

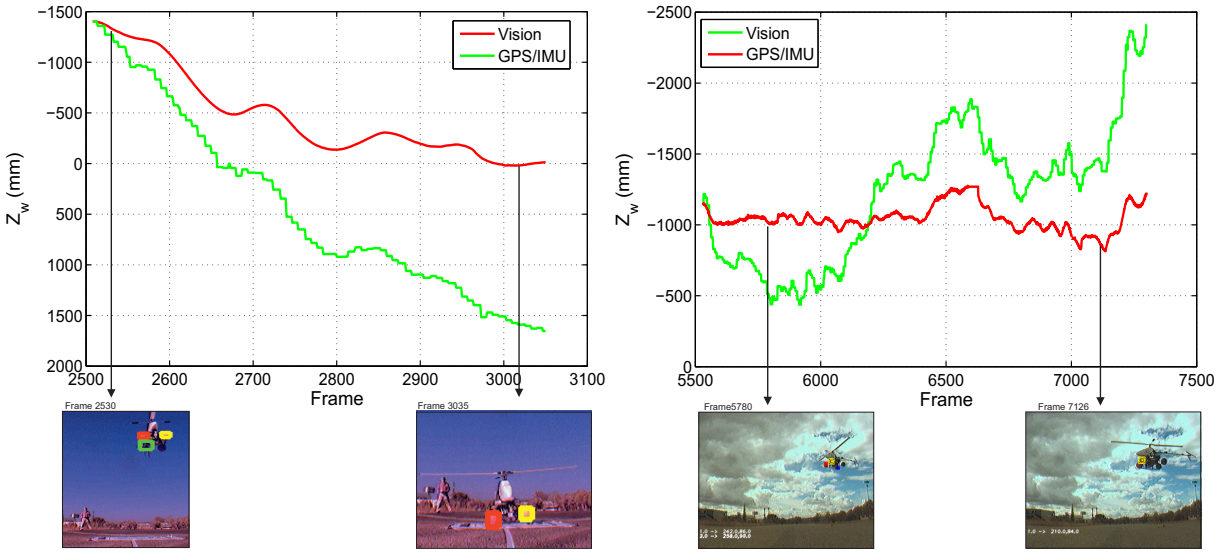
Figure 6.12: Position estimations of the Y_w axis. The vision-based estimations (red/dark line) are compared with the ones estimated by the UAV's state estimator (based on GPS/IMU, green/light line). Thumbnail images permit to analyze the performance of the vision system.

line) the UAV would be underground. Conducting the same analysis for the images of the second test (see Figure 6.13(b)), the images reveal that the behavior of the motion estimated by the trinocular system coincides with the motion conducted by the UAV.

Finally, Figure 6.14 presents the results of the estimation of the yaw angle for the landing test (test 1, Figure 6.14(a)), and also for a specific part of a third test where the UAV was manually commanded to change its heading (test 3, Figure 6.14(b)). In the figures, it can be seen that the vision-based estimations have behavior and values that are similar to the ones estimated by the IMU. The thumbnail images also show the correlation of the estimated data with the UAV motions. The obtained RMSEs were 6.25° for the test shown in Figure 6.14(a), and 4.76° for the test shown in Figure 6.14(b). In some parts of the tests, the results of the feature extraction algorithm were very noisy, and the number of detected features fluctuated from 2 to 3, making the estimation of the yaw angle noisy in some parts of the tests, as can be seen in Frames 2900-3050 of Figure (6.14(a)). However, in spite of this, the vision-based estimations prove to be coherent with the motions conducted by the UAV. Additional improvements in the feature extraction stage, such as those obtained by using leds, and also by filtering the data, help to further improve the estimations.

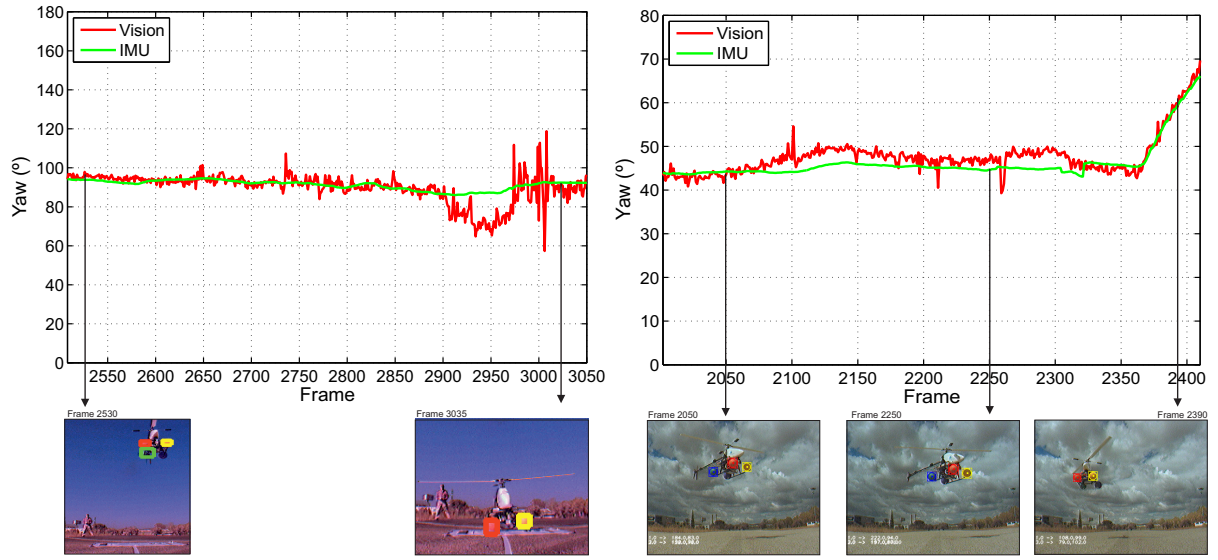
6.3.4. Discussion

Different tests have been conducted to analyze the performance of the trinocular system when estimating the position and orientation of the UAV. In these tests, it has been found that the reconstructed values are consistent with the real movements experienced by the helicopter (analyzing the position of the UAV in the images), and also that these



(a) Test1 manual landing. (b) Test 3 manual flight.

Figure 6.13: Position estimations of the Z_w axis. The vision-based estimations (red/dark line) are compared with the ones estimated by the UAV's state estimator (based on GPS/IMU, green/light line). Thumbnail images permit to analyze the performance of the vision system.



(a) Test1 manual landing. (b) Test 3 manual flight.

Figure 6.14: Estimations of the yaw (θ) angle. The vision-based estimations (red/dark line) are compared with the ones estimated by the UAV's state estimator (based on GPS/IMU, green/light line). Thumbnail images permit to analyze the performance of the vision system.

values have a behavior that is similar to the one estimated by other on-board sensors (GPS/IMU). We have analyzed the precision of the external visual system by comparing the visual estimation with 3D known positions. The precision that was achieved, ±10 cm, in the three axes, allow us to conclude that the position estimation obtained by the ground multi-camera system is more accurate than the one estimated by other on-board sensors, taking into account that the errors in position using the on-board sensors, which is based on GPS information, are around ±0.5 m for the X_w and Y_w axes, and ±1 m for the Z_w axis (height estimation). This is why the position estimation from the GPS/IMU was used only to compare the behavior of the vision-based estimations, and not to compare absolute values.

Additionally, the tests have shown that the trinocular system can be used for estimating the 3D position and orientation of the UAV, and that the different algorithms that have been selected permit to obtain estimations at real-time frame rates 15 FPS. However, additional improvements in the feature extraction stage, especially for dealing with illumination changes when operating outdoors, will help to improve the quality of the vision-based estimation.

Videos of the different tests described in this section can be found in [52].

6.4. Summary

In this chapter, a real-time pose estimation algorithm based on a ground multi-camera system has been presented and validated. A trinocular system located on the ground is in charge of detecting and tracking color landmarks on-board the UAV. The pose estimation algorithm is based on the 3D reconstruction of these landmarks. The algorithm has been tested indoors and outdoors, the latter corresponding to real flight tests conducted with a Rotomotion SR20 VTOL UAV.

The experimental results show that the estimation is consistent regarding 3D real data and has approximate errors of 5 cm in the X_w and Z_w axes, and of 10 cm in the Y_w axis (at 3 meters depth). These values satisfy the requirements of the tasks we expect to accomplish with the system (low height tasks: landing, takeoff, and positioning). Additionally, the comparison of the information provided by the vision system with that of the UAV's state estimator shows that the vision system improves the position estimation, especially the helicopter's height estimation, whose accuracy based on GPS is 0.5 m or lower when the vehicle approaches the ground.

The pose estimation results obtained with the ground trinocular system make the proposed system suitable for maneuvers at low heights (lower than 5 m) and close to structures, where precise movements are required, and also for situations where the GPS signal is inaccurate or unavailable.

Chapter 7

Visual Servoing for UAVs

7.1. Introduction

In this chapter[1], the vision-based algorithms presented in previous chapters are used to conduct vision-based tasks for UAVs. The main objective of this chapter is to test how good (robust and fast) the vision-based estimations are to be included in the UAV's control loop, and how including vision-based estimation in the control loop can be used to improve the capabilities of UAVs, e.g. to locate or move towards a specific object.

Visual servoing or visual sevo control consists in using visual information to control a robot combining image processing and control techniques. The bottleneck of visual servoing is to obtain important robust visual information of the environment that can be used for controlling purposes, e.g in real-time. In this sense, UAVs can be considered a challenging testbed for visual servoing techniques. It combines the difficulties of constant changes in the image information, due to constant vibrations, with the difficulties of outdoors operations (e.g. non-structured environments, illumination changes, etc) and 3D motion. In previous chapters, importance was attached to the obtention of robust visual information for the visual servoing tasks.

Depending on how the different components of the visual servoing task are and how they interact, different visual servoing schemes are defined. In [92], a visual servoing

[1]Publications of the author related to this chapter:
- A Hierarchical Tracking Strategy for Vision-Based Applications On-Board UAVs [123]
- Visual Servoing for UAVs [37]

taxonomy was introduced answering two questions: does the visual system provide set-points or directly joint-level inputs? is the error signal defined in 3D space or in the image plane in terms of image features? If the visual information is used to directly stabilize the robot, i.e. the visual system directly computes the joint inputs, it is called in [92] as direct visual servoing. In this scheme, the visual controller is responsible of the stability of the system, and high frequency rates ($\approx$ less than 10 ms) are required to close the loop. Conversely, if the visual system provides set-points to a low-level controller that is the one in charge of the stability of the robot, then it is called a dynamic look-and-move system which uses a hierarchical architecture, in which the kinematic singularities of the robot are separated from the visual controller. Currently, many robots already have an interface for accepting cartesian velocity or incremental position commands, reason why this scheme is the most commonly adopted in visual servoing.

On the other hand, by answering the second question, a second classification differentiates the visual servoing approaches according to the error function: position-based or image-based visual servoing. If the visual information is used to estimate the pose of the target with respect to the camera, and the error is calculated in the pose space, this is called position-based visual servoing (PBVS). If, on the other hand, control commands are computed from image features, this is called image-based visual servoing (IBVS). The image based scheme reduces delays caused by the pose estimation and also by errors due to camera calibration.

Visual servoing schemes are also classified according to the camera configuration (the way the camera is mounted on the robot) [92]: eye-in-hand, when the camera is mounted on the robot; or eye-to-hand, when the camera is fixed in the workspace. In the first one, there is a known relationship between the camera and the robot. In the second one, the image of the target can be independent of the robot motion, unless the target is the same robot. In both configurations, calibration of the intrinsic and extrinsic camera parameters is required either to determine the position of the camera with respect to the world coordinate system, or to determine the relative pose between the camera and the robot, which is required for the eye-in-hand configuration. The latter is known as the hand/eye calibration problem [92].

In [43] the basic components of the visual servoing techniques are explained. In this section, we make use of them to define the basis of a visual servoing task which is going to be used in the following sections to define the different visual servoing strategies adopted in this thesis. As was presented in [43] and using the same notation, the objective of a vision-based control task is to minimize:

$$\mathbf{e}(t) = \mathbf{s}[\mathbf{m}(t), \mathbf{a}] - \mathbf{s}^* \tag{7.1}$$

where $\mathbf{m}(t)$ corresponds to the image coordinates of features; $\mathbf{a}$ is a set of parameters such as camera calibration parameters or 3D models of objects, that can be used (depending on the kind of visual servoing strategy adopted) to compute the visual features; and $\mathbf{s}^*$ are the desired values of the features (the setpoints).

As was mentioned previously, the different visual servoing architectures differ in the way $\mathbf{s}$ is chosen. $\mathbf{s}$ can be a set of features in the image plane or a set of 3D parameters that are estimated from image measurements. In order to design a velocity controller, the relationship between the time variation of $\mathbf{s}$ and the camera velocity

$\mathbf{v_c} = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^\top$ is required, where $\boldsymbol{v}_c$ represents the linear velocity of the origin of the camera frame and $\boldsymbol{\omega}_c$ the angular velocity of the camera frame. The relationship between these terms is given by:

$$\dot{\mathbf{s}} = \mathbf{L}_s \mathbf{v}_c \tag{7.2}$$

where $\mathbf{L}_s$ is called the interaction matrix related to $\mathbf{s}$. Calculating the time variation of Equation (7.1) and replacing it in Equation (7.2), the relationship between the camera velocity and the time variation of the error is obtained:

$$\dot{\mathbf{e}} = \mathbf{L}_e \mathbf{v}_c \tag{7.3}$$

Where $\mathbf{L}_s = \mathbf{L}_e$. In a classical visual servoing strategy, [43] in order to ensure an exponential decoupled decrease of the error $\dot{\mathbf{e}} = -\lambda \mathbf{e}$, the input to the robot controller is defined as:

$$\mathbf{v}_c = -\lambda \mathbf{L_e}^{+} \mathbf{e} \tag{7.4}$$

where $\mathbf{L_e}^{+}$ is the pseudo-inverse of $\mathbf{L_e}$, being $\mathbf{L_e^{+}} = (\mathbf{L_e}^\top \mathbf{L_e})^{-1} \mathbf{L_e}^\top$.

However, due to noise of image measurements, instead of an exact value of $\mathbf{L_e}$ it is only possible to obtain an estimated value of the interaction matrix $\widehat{\mathbf{L_e^{+}}}$. The control law in (7.4) is defined as:

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L_e^{+}}} \mathbf{e} \tag{7.5}$$

Equation (7.5) is the basic design of a visual servo controller. In this chapter, this general design has been implemented for visual servoing tasks on-board UAVs. Different strategies have been adopted to choose $\mathbf{s}$, according to the way the camera or cameras are located in the system, leading to different estimations of $\widehat{\mathbf{L_e^{+}}}$.

In this thesis, the dynamic look-and-move architecture using both eye-to-hand and eye-in-hand camera configurations are used. The eye-to-hand configuration is used with the trinocular system to control the UAV based on the pose estimation algorithm explained in Section 6; and the eye-in-hand configuration is used when the control task is conducted with the camera on-board the UAV, based on visual features or on the pose estimation algorithm described in Section 5. In this thesis, both position-based and image-based visual servoing techniques are applied to control the UAV.

A detailed explanation of the different visual servoing schemes can be found in [43, 92, 44, 119].

7.2. Image-Based Visual Servoing (IBVS)

Unmanned Aerial Vehicles are usually equipped with on-board cameras to visually monitor scenes. In this section, it is described how the information provided by an on-board camera is used to track interest objects using the visual tracking algorithm presented in Section 4, and the positions of the objects are used as features to conduct vision-based control tasks. The vision-based control commands are integrated into the UAV control system following a dynamic look-and-move architecture. As shown in Figure 7.1, in this

architecture a fast internal control loop (±120 HZ) is in charge of the UAV stability; and an external loop (the vision system), which operates at low frequencies (usually < 30 HZ), is in charge of sending references to the UAV's internal loop.

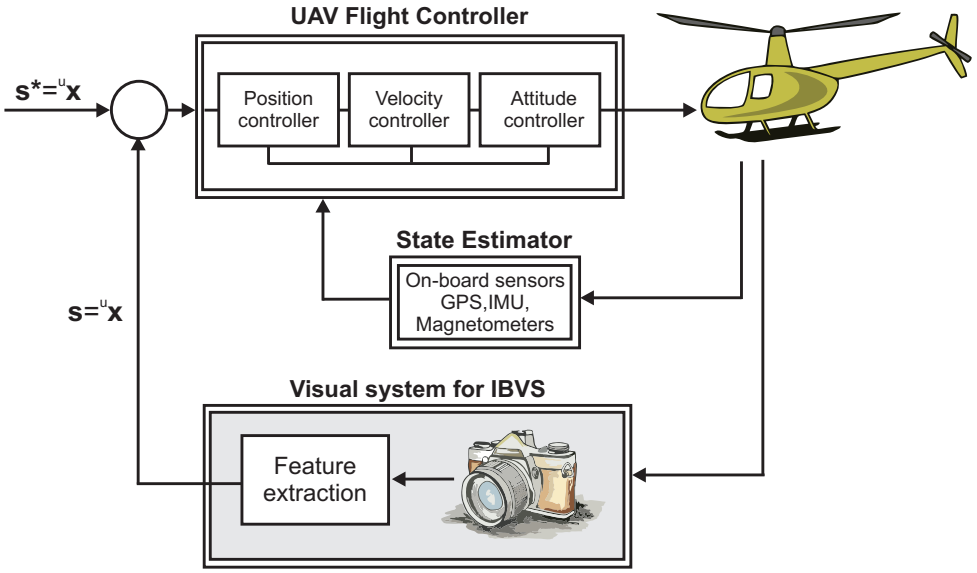


Figure 7.1: Dynamic look-and-move architecture for IBVS. A fast internal loop (flight controller) is in charge of the helicopter's stability. The vision loop is the external loop which operates at lower frequencies (< 30 Hz) and uses the visual information as a feedback to the helicopter's flight control. In the IBVS, the references are based on image features. Therefore, the control tasks are defined in the image plane.

In an IBVS scheme, these references are based on image features. Therefore, the visual servoing task based on IBVS consists in moving the UAV in order to locate the object of interest in a specific position inside the image plane (e.g. the center of the image). According to the general definition of the control task presented in Equation (7.1), in the IBVS strategy that is used to control UAVs $\mathbf{m}$ are the pixel coordinates of the center of the object, and $\mathbf{a}$ are the intrinsic camera parameters (see Appendix A). The features $\mathbf{s}$ correspond to the position of the object of interest (e.g. the center of the object, or the coordinates of a square that inscribes the object), and $\mathbf{s}^*$ is the desired position of the object. Both $\mathbf{s}^*$ and $\mathbf{s}$ are defined in the image plane (expressed in the central coordinate system, see Appendix A). We consider the case where $\mathbf{s}^*$ is constant (e.g. the center of the image plane) and where changes in $\mathbf{s}$, i.e. changes of the visual features are due to motion of the camera, and so to motion of the UAV (the system uses an eye-in-hand configuration, which means that the camera is on-board the UAV). Therefore, the control task consists in centering the object of interest in the image plane by sending vision-based control commands to the UAV.

If the center of the object of interest with 3D coordinates ${}^{c}\mathbf{x} = ({}^{c}x, {}^{c}y, {}^{c}z)$ with respect to the camera frame projects in the image plane as ${}^{u}\mathbf{x} = ({}^{u}x, {}^{u}y)$ in the central camera coordinate system, then using the pinhole camera model (see Appendix A), the following relation between the 3D point and the 2D point in the image plane can be found:

$$
\begin{aligned}
{}^{\mathrm{u}}x &= ({}^{\mathrm{f}}x - c_x)/f_x = {}^{\mathrm{c}}x/{}^{\mathrm{c}}z \\
{}^{\mathrm{u}}y &= ({}^{\mathrm{f}}y - c_y)/f_y = {}^{\mathrm{c}}y/{}^{\mathrm{c}}z
\end{aligned}
\tag{7.6}
$$

where according to Equation (7.1), $\mathbf{m} = ({}^{\mathrm{f}}x, {}^{\mathrm{f}}y)$ are the coordinates of the center of the object with respect to the lateral camera coordinate system in the image plane (in pixel units), and $\mathbf{a} = (c_x, c_y, f_x, f_y)$ are the camera intrinsic parameters. Therefore, the features to control are defined as $\mathbf{s} = {}^{\mathrm{u}}\mathbf{x} = ({}^{\mathrm{u}}x, {}^{\mathrm{u}}y)$, e.g. the center of the object.

Taking time derivatives of Equation (7.6):

$$
\begin{aligned}
{}^{\mathrm{u}}\dot{x} &= {}^{\mathrm{c}}\dot{x}/{}^{\mathrm{c}}z - {}^{\mathrm{c}}x{}^{\mathrm{c}}\dot{z}/{}^{\mathrm{c}}z^2 = ({}^{\mathrm{c}}\dot{x} - {}^{\mathrm{u}}x{}^{\mathrm{c}}\dot{z})/{}^{\mathrm{c}}z \\
{}^{\mathrm{u}}\dot{y} &= {}^{\mathrm{c}}\dot{y}/{}^{\mathrm{c}}z - {}^{\mathrm{c}}y{}^{\mathrm{c}}\dot{z}/{}^{\mathrm{c}}z^2 = ({}^{\mathrm{c}}\dot{y} - {}^{\mathrm{u}}y{}^{\mathrm{c}}\dot{z})/{}^{\mathrm{c}}z
\end{aligned}
\tag{7.7}
$$

and relating the velocity of a 3D point with the spacial velocity of the camera:

$$
{}^{\mathrm{c}}\dot{\mathbf{x}} = -\boldsymbol{v}_{\mathbf{c}} - \boldsymbol{\omega}_c \times {}^{\mathrm{c}}\mathbf{x}
$$

$$
\begin{aligned}
{}^{\mathrm{c}}\dot{x} &= -v_x - \omega_y{}^{\mathrm{c}}z + \omega_z{}^{\mathrm{c}}y \\
{}^{\mathrm{c}}\dot{y} &= -v_y - \omega_z{}^{\mathrm{c}}x + \omega_x{}^{\mathrm{c}}z \\
{}^{\mathrm{c}}\dot{z} &= -v_z - \omega_x{}^{\mathrm{c}}y + \omega_y{}^{\mathrm{c}}x
\end{aligned}
\tag{7.8}
$$

Then, using Equations (7.7) and (7.8), the velocity of the projected 3D point in the image plane is related with the camera motion (i.e. the relation between the motion of features and the motion of the camera), as follows:

$$
\begin{aligned}
{}^{\mathrm{u}}\dot{x} &= -v_x/{}^{\mathrm{c}}z + {}^{\mathrm{u}}xv_z/{}^{\mathrm{c}}z + {}^{\mathrm{u}}x{}^{\mathrm{u}}y\omega_x - (1 + {}^{\mathrm{u}}x^2)\omega_y + {}^{\mathrm{u}}y\omega_z \\
{}^{\mathrm{u}}\dot{y} &= -v_y/{}^{\mathrm{c}}z + {}^{\mathrm{u}}yv_z/{}^{\mathrm{c}}z + (1 + y^2)\omega_x - {}^{\mathrm{u}}x{}^{\mathrm{u}}y\omega_y - {}^{\mathrm{u}}x\omega_z
\end{aligned}
\tag{7.9}
$$

which can be expressed as:

$$
{}^{\mathrm{u}}\dot{\mathbf{x}} = \mathbf{L}_{\mathbf{x}}\mathbf{v}_{\mathbf{c}} \tag{7.10}
$$

where $\mathbf{L}_{\mathbf{x}}$ is called the interaction matrix:

$$
\mathbf{L}_{\mathbf{x}} = \begin{bmatrix}
-\dfrac{1}{{}^{\mathrm{c}}z} & 0 & \dfrac{{}^{\mathrm{u}}x}{{}^{\mathrm{c}}z} & {}^{\mathrm{u}}x{}^{\mathrm{u}}y & -(1 + {}^{\mathrm{u}}x^2) & {}^{\mathrm{f}}y \\
0 & -\dfrac{1}{{}^{\mathrm{c}}z} & \dfrac{{}^{\mathrm{u}}y}{{}^{\mathrm{c}}z} & (1 + {}^{\mathrm{u}}y^2) & -{}^{\mathrm{u}}x{}^{\mathrm{u}}y & -{}^{\mathrm{u}}x
\end{bmatrix}
\tag{7.11}
$$

As can be seen in Equation (7.11), $\mathbf{L}_{\mathbf{x}}$ depends on the value ${}^{\mathrm{c}}z$, which is the depth of the point relative to the camera frame, and also depends on the intrinsic camera parameters that permit to estimate ${}^{\mathrm{u}}\mathbf{x}$ from the coordinates of the point in pixel units ${}^{\mathrm{f}}\mathbf{x}$. However, in practice, the values that are used to estimate the interaction matrix correspond to an estimation of the real ones. This is why the real $\mathbf{L}_{\mathbf{x}}$ is not used, but what is used instead is an approximation of the real one: $\widehat{\mathbf{L}_{\mathbf{x}}}$.

Equation (7.11) relates the motion of the camera with the motion of the object perceived in the camera image. However, visual servo control applications typically require

the reverse: computation of $\mathbf{v_c}$ given ${}^{f}\dot{\mathbf{x}}$. As explained in [43], there are different ways of selecting $\widehat{\mathbf{L}_\mathbf{e}^+}$. If the current depth of each feature is known, one way is to select $\mathbf{L_e} = \mathbf{L_x}$ and $\widehat{\mathbf{L}_\mathbf{e}^+} = \mathbf{L}_\mathbf{e}^+$. Therefore:

$$\mathbf{v}_c = -\lambda \mathbf{L}_\mathbf{e}^+ (\mathbf{s} - \mathbf{s}^*) \tag{7.12}$$

7.2.1. Results: IBVS with an On-Board Camera

In this section, the Image-Based Visual Servoing strategy is applied to control a UAV. As shown in Figure 7.2, the objective of the task is to position the object of interest in different places in the image plane (e.g. the center of the image, the bottom left corner, etc.). Simulated and real flight tests are conducted to achieve this.

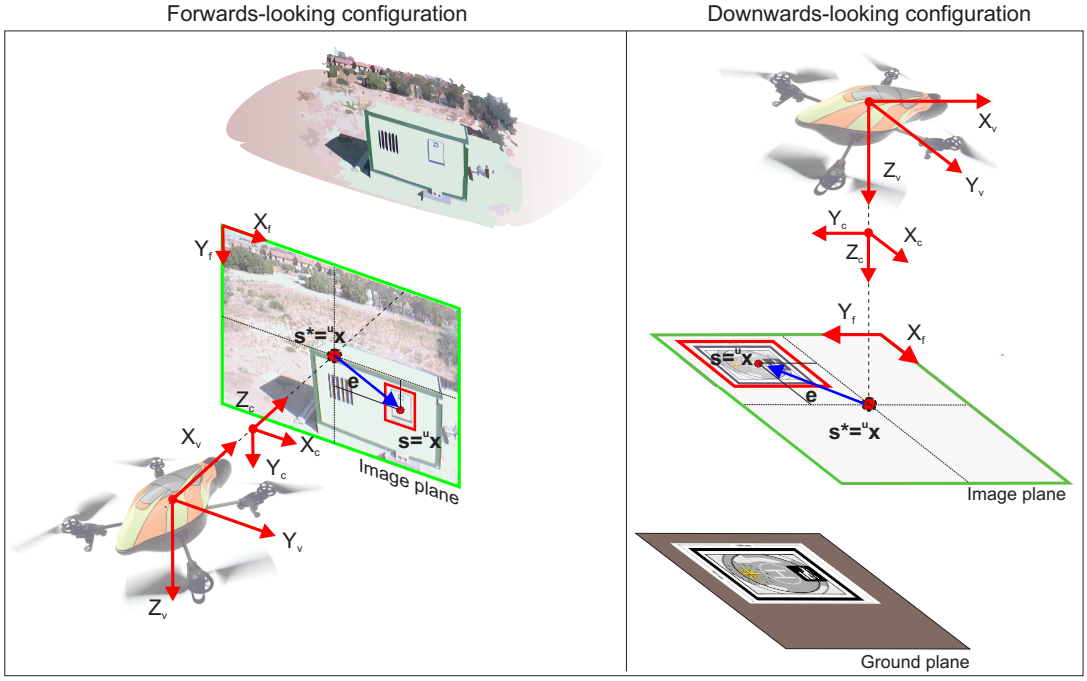


Figure 7.2: IBVS objective. The objective of the control task is to place the object of interest in a specific position in the image plane, e.g. the image center. Tests have been conducted using different camera configurations: forwards-looking and downwards-looking.

In the tests, a camera located on-board a quadrotor is used to capture images of objects of interest, and the HMPMR-ICIA algorithm explained in Section 4 is used to track the objects. Therefore, their 2D positions in the image plane are known. Depending on the configuration of the on-board camera (forwards-looking or downwards-looking), different interaction matrices are estimated and different degrees of freedom are controlled.

This section presents simulation results and also results from real flight tests. In the simulated test, a virtual environment that uses the ROS (Robot Operating System) framework [155], the 3D simulator Gazebo [71], and the Starmac aircraft model [178] are used. In the real flight tests, the AR.Drone-Parrot UAV [7] is used.

7.2.1.1. Simulation Tests

In this test, simulated data obtained from the 3D simulator Gazebo [71] and the Starmac aircraft model [178] are used to test an IBVS strategy. In this test, a planar object (a helipad) is located on the ground of the simulation environment, and a camera located on-board a quadrotor looking downwards is used to capture synthetic images of the helipad. The center of the helipad is tracked by the HMPMR-ICIA 4-3-2-2 algorithm, and is used as feature to control the UAV. The control task of this test consists in placing the helipad in different positions in the image plane, as shown in the right image of Figure 7.2. Once the first setpoint is achieved ($\mathbf{s_1^*}$), the UAV hovers over it for a few seconds and then moves towards the following setpoints: $\mathbf{s_2^*}$ and $\mathbf{s_3^*}$. The motion is conducted over the helipad at a fixed altitude (at 10 m over the ground). Image-based control commands are sent to the UAV in order to locate the helipad in the different defined positions in the image plane.

Therefore, $\mathbf{s}$ is defined as $\mathbf{s} = [\frac{({}^{\mathrm{f}}x - c_x)}{f_x}, \frac{({}^{\mathrm{f}}y - c_y)}{f_y}]$, the center of the object found by the tracking algorithm; and the different desired positions are defined as: $\mathbf{s_1^*} = [\frac{(320 - c_x)}{f_x}, \frac{(240 - c_y)}{f_y}]$, $\mathbf{s_2^*} = [\frac{(520 - c_x)}{f_x}, \frac{(440 - c_y)}{f_y}]$, and $\mathbf{s_3^*} = [\frac{(120 - c_x)}{f_x}, \frac{(140 - c_y)}{f_y}]$, where the first setpoint $\mathbf{s_1^*}$ corresponds to the center of the image plane taking into account that the image is 640×480 pixels in size. Both the current and the desired positions, $\mathbf{s}$ and $\mathbf{s}^*$, are expressed in the central coordinate system (see the camera model in Appendix A).

The vision-based commands sent to the UAV are based on Equation (7.12). In Equation (7.12), $\mathbf{L_e^+}$ relates the 6 DOF of the camera. However, because only two features are going to be used for the control task (x and y positions), only 2 DOF can be controlled with this information. Therefore, assuming that the UAVs' height will be constant during the task, then the camera velocity $v_z = 0$. Additionally, assuming that the control task will be conducted at low speeds < 1 m/s, then the attitude angles of the UAV can be assumed to be constant or can be assumed to have a minimum value, then $\omega_x \approx \omega_y \approx \omega_z \approx 0$.

With all these assumptions, then $\mathbf{L_x}$ is now written as:

$$\mathbf{L_x} = \begin{bmatrix} -\frac{1}{{}^{\mathrm{c}}z} & 0 \\ 0 & -\frac{1}{{}^{\mathrm{c}}z} \end{bmatrix} \tag{7.13}$$

Assuming that $\mathbf{L_e} = \mathbf{L_x}$ and $\widehat{\mathbf{L_e^+}} = \mathbf{L_e^+}$, then:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = -\lambda \widehat{\mathbf{L_e^+}} (\mathbf{s} - \mathbf{s}^*) \tag{7.14}$$

In Figure 7.2, it can be seen that in the downwards-looking configuration the camera and the UAV coordinate systems are related by a rotation in the Z axis, as shown in the right image of Figure 7.2. Therefore, from Equation (7.14), the vision-based commands that are sent to the UAV are defined as follows:

$$\begin{bmatrix} v_{x_{\mathrm{UAV}}} \\ v_{y_{\mathrm{UAV}}} \end{bmatrix} = \begin{bmatrix} -v_y \\ v_x \end{bmatrix} \tag{7.15}$$

In Equation (7.13), the depth of the points is required in order to estimate $\widehat{\mathbf{L}_\mathbf{e}^+}$. Taking into account that the tracked object is on the ground, and assuming that the position of the UAV coordinate system coincides with the camera coordinate system, then the depth of the points can be directly obtained from the altitude of the UAV estimated by the state estimator, or it can be estimated based on vision if the size of the object is known. Therefore, with these assumptions, $\widehat{\mathbf{L}_\mathbf{e}^+}$ can be estimated. In the tests, λ has been defined experimentally.

In Figure 7.3 it is possible to see the trajectory of the UAV during the IBVS task. The blue line corresponds to the position data obtained by the Starmac-ROS package [178]. In the figure, the three setpoints to which the UAV was commanded to move to can be seen. First, the UAV moves towards the helipad in order to place the helipad in the center of the image plane $\mathbf{s}_\mathbf{1}^*$, and then the UAV is positioned in the other defined setpoints.

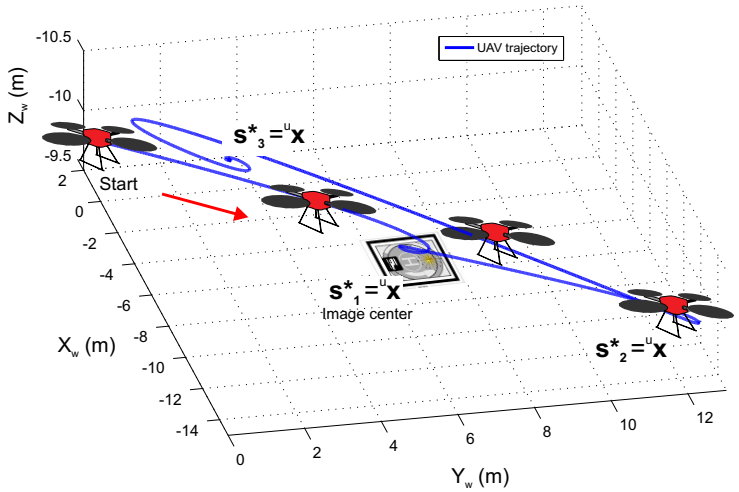


Figure 7.3: 3D Trajectory during the IBVS task. Three set points $\mathbf{s}^*$ have been defined in different parts of the image (the center, the bottom right corner, and the upper left corner). The blue line corresponds to the position data obtained by the Starmac-ros package [178].

Finally, Figure 7.4 plots the trajectory of the center of the helipad in the image plane. It can be seen that the three defined setpoints have been reached using the control law defined in Equation (7.14).

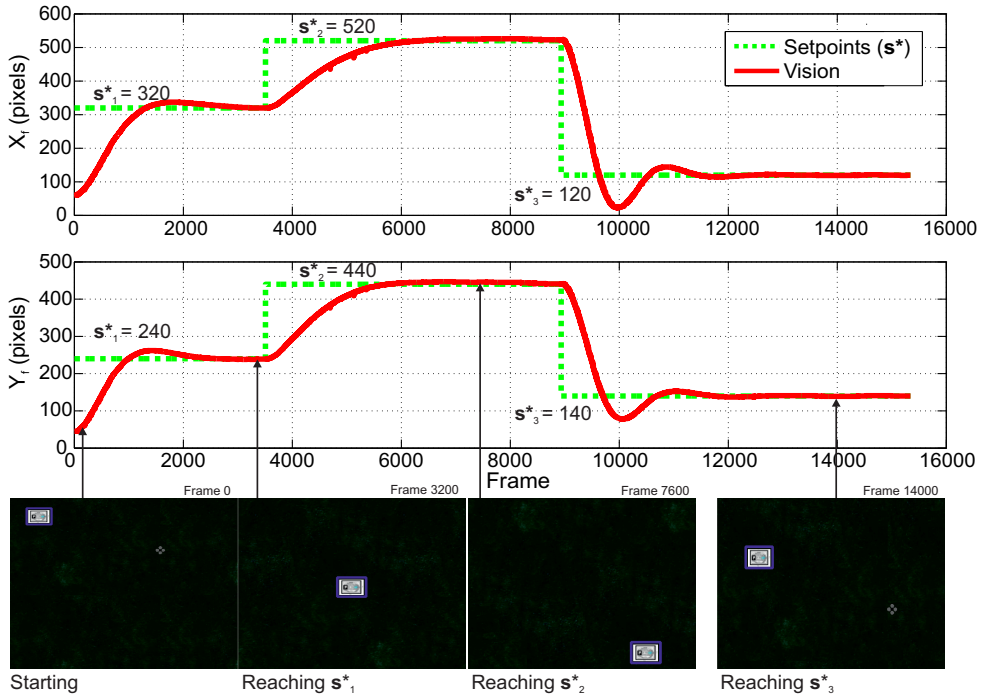


Figure 7.4: Simulation results of the IBVS task. The figures show the trajectory of the center of the helipad in the image plane (red/dark solid line). The three defined setpoints (green/light dashed line) have been reached.

7.2.1.2. Real Flight Test

The tests were conducted with the Parrot UAV shown in Figure 7.5. The Parrot has two cameras. The forwards-looking camera captures images of 320×240 pixels size and is used, in this test, to track an object located on a wall, as described in the left image of Figure 7.2. The images are processed in a laptop on the ground, and vision-based control commands are sent back to the Parrot controller through a WIFI link, using the ROS driver for the AR-Drone [10]. The on-board cameras have been calibrated using the Caltech camera calibration toolbox for Matlab [23].

Figure 7.5: Parrot-AR.Drone. UAV used for the real tests.

The object is tracked by the HMPMR-ICIA 8-4-2-2 algorithm. The features used for the visual control task $\mathbf{s}$ are defined as $\mathbf{s} = [\frac{({}^{\mathrm{f}}x - c_x)}{f_x}, \frac{({}^{\mathrm{f}}y - c_y)}{f_y}, \frac{{}^{\mathrm{u}}l}{f}]$, where the first two terms refer to the center of the object in the image plane found by the tracking algorithm,

and the third term refers to the size of the object in the image plane which is related to the distance of the UAV with respect to the wall, as follows: ${}^{\mathrm{u}}l \propto 1/{}^{\mathrm{c}}z$. To control the distance to the wall many options can be considered. However for simplicity, we control the distance to the wall using the size of the upper edge of the object to track. Therefore, ${}^{\mathrm{u}}l = ({}^{\mathrm{u}}x_2 - {}^{\mathrm{u}}x_1)$, where ${}^{\mathrm{u}}\mathbf{x_1}$ and ${}^{\mathrm{u}}\mathbf{x_2}$ are the upper left and upper right coordinates of the object that is being tracked.

Therefore, the control task of this test consists in placing the helipad in different positions in the image plane and in maintaining the UAV in a defined position with respect to the wall. The two setpoints are defined as $\mathbf{s_1^*} = [\frac{(90-c_x)}{f_x}, \frac{(230-c_y)}{f_y}, \frac{1.2\times{}^{\mathrm{u}}l_{\mathrm{init}}}{f}]$ and $\mathbf{s_2^*} = [\frac{(70-c_x)}{f_x}, \frac{(170-c_y)}{f_y}, \frac{0.6\times{}^{\mathrm{u}}l_{\mathrm{init}}}{f}]$. The first two terms refer to the position in the image plane, and the third term refers to the desired distance (depth) to the object, which is defined according to the size of the object in the image plane when the control task is initialized (l_{init}), e.g. 1.2 times the initial size. Image-based control commands are sent to the UAV in order to locate the object in the different defined positions in the image plane. Once the first setpoint is achieved ($\mathbf{s_1^*}$), the UAV moves towards the second setpoint ($\mathbf{s_2^*}$). Both the current and the desired positions, $\mathbf{s}$ and $\mathbf{s}^*$, are expressed in the central coordinate system (see the camera model in Appendix A).

Equations (7.6), (7.7), and (7.9) show how to obtain the values of the interaction matrix for the first two features ${}^{\mathrm{u}}x$ and ${}^{\mathrm{u}}y$. For the third feature ${}^{\mathrm{u}}l$ we have that:

$$ {}^{\mathrm{u}}l = ({}^{\mathrm{u}}x_2 - {}^{\mathrm{u}}x_1) = ({}^{\mathrm{c}}x_2/{}^{\mathrm{c}}z_2) - ({}^{\mathrm{c}}x_1/{}^{\mathrm{c}}z_1) \tag{7.16} $$

Taking time derivative of Equation (7.16), and assuming that ${}^{\mathrm{c}}z_1 = {}^{\mathrm{c}}z_2 = {}^{\mathrm{c}}z$, then we have that:

$$ {}^{\mathrm{u}}\dot{l} = ({}^{\mathrm{u}}\dot{x}_2 - {}^{\mathrm{u}}\dot{x}_1) = \left[\frac{{}^{\mathrm{c}}\dot{x}_2 - {}^{\mathrm{u}}x_2{}^{\mathrm{c}}\dot{z}}{{}^{\mathrm{c}}z}\right] - \left[\frac{{}^{\mathrm{c}}\dot{x}_1 - {}^{\mathrm{u}}x_1{}^{\mathrm{c}}\dot{z}}{{}^{\mathrm{c}}z}\right] \tag{7.17} $$

Then, replacing Equation (7.8) in Equation (7.17), and reorganizing the equation, the relationship between the rate change of size of the object to track with the motion of the camera is found as follows:

$$ {}^{\mathrm{u}}\dot{l} = (v_z{}^{\mathrm{u}}l)/{}^{\mathrm{c}}z + {}^{\mathrm{u}}l{}^{\mathrm{u}}x\omega_x - {}^{\mathrm{u}}l{}^{\mathrm{u}}y\omega_y \tag{7.18} $$

Therefore with Equation (7.9) and Equation (7.18) the interaction matrix is written as follows:

$$ \mathbf{L_x} = \begin{bmatrix} -\frac{1}{{}^{\mathrm{c}}z} & 0 & \frac{{}^{\mathrm{u}}x}{{}^{\mathrm{c}}z} & {}^{\mathrm{u}}x{}^{\mathrm{u}}y & -(1+{}^{\mathrm{u}}x^2) & {}^{\mathrm{f}}y \\ 0 & -\frac{1}{{}^{\mathrm{c}}z} & \frac{{}^{\mathrm{u}}y}{{}^{\mathrm{c}}z} & (1+{}^{\mathrm{u}}y^2) & -{}^{\mathrm{u}}x{}^{\mathrm{u}}y & -{}^{\mathrm{u}}x \\ 0 & 0 & \frac{{}^{\mathrm{u}}l}{{}^{\mathrm{c}}z} & {}^{\mathrm{u}}l{}^{\mathrm{u}}y & -{}^{\mathrm{u}}l{}^{\mathrm{u}}x & 0 \end{bmatrix} \tag{7.19} $$

The vision-based commands are based on Equation (7.12). Because in this test the control task is defined with only 3 features, only 3 DOF are controlled. Therefore, Equation (7.11) is defined using the following assumption: the UAV will fly at low speeds < 1 m/s, therefore the attitude angles can be assumed to be constant or can be assumed to have a minimum value, then $\omega_x \approx \omega_y \approx \omega_z \approx 0$. Thus $\mathbf{L_x}$ is now written as:

$$\mathbf{L_x} = \begin{bmatrix} -\frac{1}{{}^{\mathrm{c}}z} & 0 & \frac{{}^{\mathrm{u}}x}{{}^{\mathrm{c}}z} \\ 0 & -\frac{1}{{}^{\mathrm{c}}z} & \frac{{}^{\mathrm{u}}y}{{}^{\mathrm{c}}z} \\ 0 & 0 & \frac{{}^{\mathrm{u}}l}{{}^{\mathrm{c}}z} \end{bmatrix} \tag{7.20}$$

As in the other tests, assuming that $\mathbf{L_e} = \mathbf{L_x}$ and $\widehat{\mathbf{L_e^+}} = \mathbf{L_e^+}$, then the vision-based commands that are sent to the UAV are defined as follows:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v_{y_{\mathrm{UAV}}} \\ v_{z_{\mathrm{UAV}}} \\ v_{x_{\mathrm{UAV}}} \end{bmatrix} = -\lambda \widehat{\mathbf{L_e^+}} (\mathbf{s} - \mathbf{s}^*) \tag{7.21}$$

Figure 7.6 shows the results of the test. The thumbnail images correspond to images captured during the test. After taking-off, the object to track is selected and vision-based commands are sent to the Parrot in order to reach the different setpoints. In the figure, it can be seen that the defined control law allows to complete the control task, and it can also be seen that the tracking algorithm tracked the template throughout the task. Thumbnail images of Figure 7.6 show the different stages of the task.

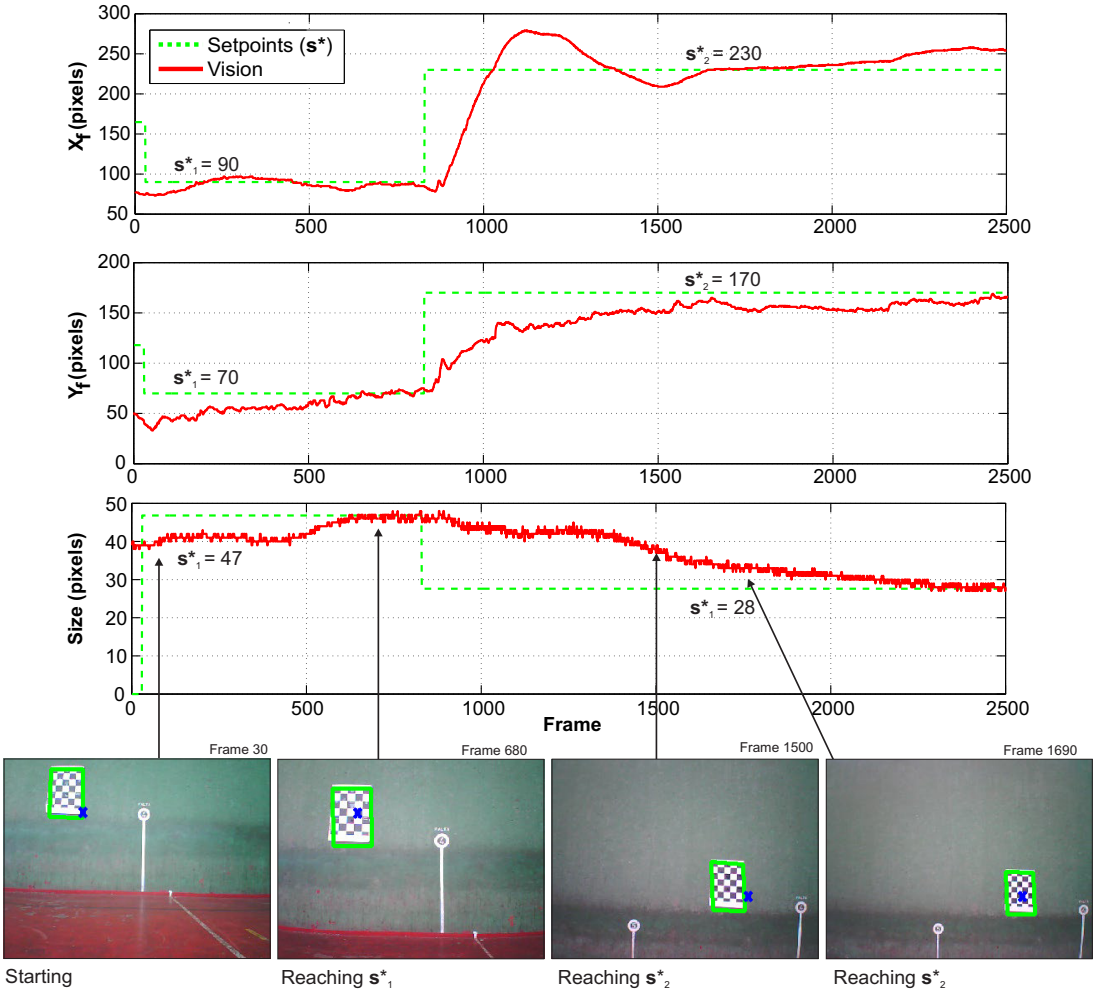


Figure 7.6: Flight test results of the IBVS task. The first two figures show the trajectory of the center of the object in the image plane (red/dark solid line). The third figure shows the evolution of the size of the template (red/dark solid line). It can be seen that the two defined setpoints (green/light dashed line) have been reached.

7.3. Position-Based Visual Servoing

The position-based or pose-based visual servoing method relies on a pose estimation algorithm (based on image features) used to estimate the relative pose between the object and the camera. The error function is defined in the 3D space. Therefore, knowledge of the intrinsic camera parameters and the 3-D model of the object are required. The problem of estimating the pose based on image features is known as the 3D localization problem [43]. In this thesis, Chapters 5 and 6 dealt with three pose estimation algorithms, two of which are used in this section to control a UAV.

As presented in Section 7.2, the vision-based control commands are integrated into the UAV control system following a dynamic look-and-move architecture. As shown in Figure 7.7, a fast (> 100 Hz) internal control loop is in charge of the UAV stability; and a slower (< 30 Hz) external loop (the vision system) is in charge of sending references to the UAV's internal loop. As can be seen in the figure, in an PBVS scheme, these references

are pose references defined in the cartesian space. Therefore, the visual servoing task based on PBVS consists in moving the UAV to a specific 3D position.

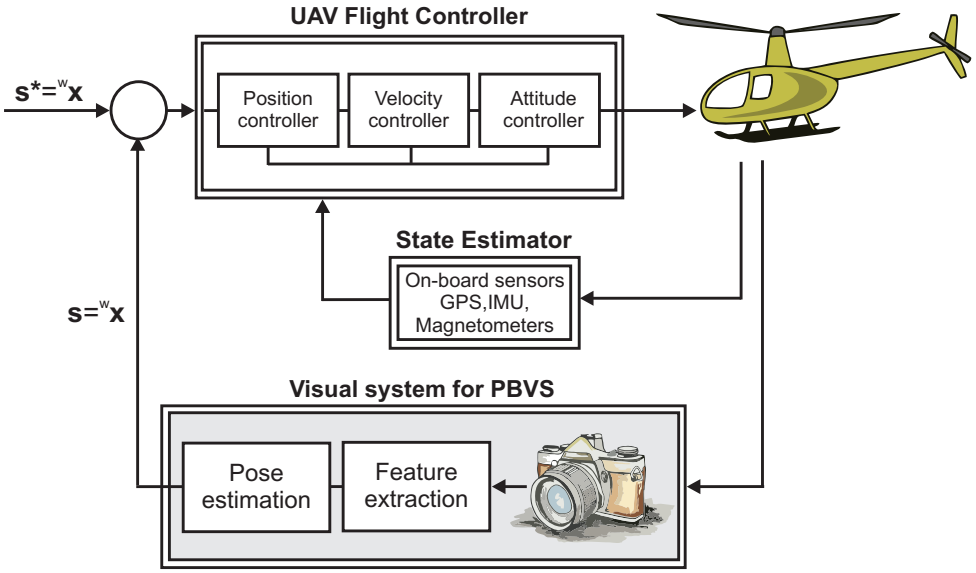


Figure 7.7: PBVS dynamic look-and-move architecture. A fast internal loop (flight controller) is in charge of the helicopter's stability. In a PVBS strategy, the error function is defined in the 3D space. Therefore, the visual system requires a pose estimation algorithm, such as the ones presented in Chapters 5 and 6.

According to the definition of the objective of a vision-based control task, which was defined in Equation (7.1), in a PBVS strategy $\mathbf{m}$ are the features used to estimate the pose; $\mathbf{a}$ are the camera intrinsic parameters (matrix $\mathbf{K}$, see Appendix A) and the object's 3D model; $\mathbf{s}$ corresponds to the parameters that define the pose of the camera (e.g. $\mathbf{R}$, $\mathbf{t}$); and $\mathbf{s}^*$ corresponds to the setpoint, i.e. the desired pose. In this thesis, we base our PVBS tasks on one of the approaches presented in [92]: Point-Features Based Motions, using both a fixed and a moving camera. In the following paragraphs, this approach is formulated for our UAV problem.

The positioning task, therefore, consists in bringing a point with UAV coordinates ${}^{\text{UAV}}\mathbf{p}$ (e.g. the center of gravity of the UAV) to a fixed point ${}^{\text{w}}\mathbf{s}^*$ that is visible in the scene and is defined in the world reference frame. This is called in [92] as point-to-point positioning. If the camera or cameras are fixed on the ground (e.g. an eye-to-hand configuration), then the error function is defined in the world coordinate system, as follows:

$$\begin{aligned} {}^{\text{w}}\mathbf{e} &= {}^{\text{w}}\mathbf{s} - {}^{\text{w}}\mathbf{s}^* \\ {}^{\text{w}}\mathbf{e} &= {}^{\text{w}}\mathbf{M}_{\text{UAV}}\, {}^{\text{UAV}}\mathbf{p} - {}^{\text{w}}\mathbf{s}^* \end{aligned} \tag{7.22}$$

were ${}^{\text{w}}\mathbf{M}_{\text{UAV}}$ is the transformation matrix that transforms the point from the UAV to the world reference frame w. In Equation (7.22), ${}^{\text{W}}\mathbf{M}_{\text{UAV}}$ corresponds to the value to be controlled. If one or more cameras (e.g. the trinocular system) provide an estimation of the position of the point with respect to the camera frame ${}^{\text{c}}\mathbf{p}$, and if the transformation matrix ${}^{\text{w}}\mathbf{M}_{\text{c}}$ is found by calibration, then ${}^{\text{w}}\mathbf{s} = {}^{\text{w}}\mathbf{M}_{\text{c}}\, {}^{\text{c}}\mathbf{p}$ is known. Therefore:

$$^{\mathrm{w}}\mathbf{e} = {}^{\mathrm{w}}\mathbf{M}_{\mathrm{c}}{}^{\mathrm{c}}\mathbf{p} - {}^{\mathrm{w}}\mathbf{s}^{*} \tag{7.23}$$

In consequence, since Equation (7.23) is linear, the control law can be written as follows, taking into account that the control input to be computed is the desired translation velocity of the UAV $\mathbf{v}_{\mathrm{UAV}}$:

$$\mathbf{v}_{\mathrm{UAV}} = -\lambda {}^{\mathrm{UAV}}\widehat{\mathbf{M}}_{\mathrm{w}}({}^{\mathrm{w}}\widehat{\mathbf{M}}_{\mathrm{c}}{}^{\mathrm{c}}\widehat{\mathbf{p}} - {}^{\mathrm{w}}\mathbf{s}^{*}) \tag{7.24}$$

where ${}^{\mathrm{w}}\widehat{\mathbf{M}}_{\mathrm{c}}$ is found by an off-line calibration; and ${}^{\mathrm{c}}\widehat{\mathbf{p}}$ and ${}^{\mathrm{UAV}}\widehat{\mathbf{M}}_{\mathrm{w}}$ are found by a pose estimation algorithm. Therefore, these values are subject to errors. In the absence of outside disturbances, Equation (7.24) will drive the system to an equilibrium state [92].

Conversely, if an on-board camera is used (e.g. eye-in-hand configuration), then the error function can be expressed in the UAV coordinate system as follows:

$$\begin{aligned} {}^{\mathrm{UAV}}\mathbf{e} &= {}^{\mathrm{UAV}}\mathbf{p} - {}^{\mathrm{UAV}}\widehat{\mathbf{M}}_{\mathrm{w}}{}^{\mathrm{w}}\mathbf{s}^{*} \\ {}^{\mathrm{UAV}}\mathbf{e} &= {}^{\mathrm{UAV}}\mathbf{p} - {}^{\mathrm{UAV}}\mathbf{M}_{\mathrm{c}}{}^{\mathrm{c}}\widehat{\mathbf{M}}_{\mathrm{w}}{}^{\mathrm{w}}\mathbf{s}^{*} \end{aligned} \tag{7.25}$$

In Equation (7.25), ${}^{\mathrm{UAV}}\mathbf{M}_{\mathrm{c}}$ is a fixed transformation. Assuming that the position of the coordinate system of the UAV coincides with the camera coordinate system, ${}^{\mathrm{UAV}}\mathbf{M}_{\mathrm{c}}$ is defined depending on the configuration of the camera on-board the UAV: downwards-looking or forwards-looking. If the camera is looking downwards, ${}^{\mathrm{UAV}}\mathbf{M}_{\mathrm{c}}$ is defined as a rotation of 90° in the Z axis. Additionally in Equation (7.25), ${}^{\mathrm{c}}\widehat{\mathbf{M}}_{\mathrm{w}}$ corresponds to a transformation matrix obtained by a pose estimation algorithm (e.g. such as the one explained in Section 5).

Therefore, the velocity commands are defined as:

$$\mathbf{v}_{\mathrm{UAV}} = -\lambda({}^{\mathrm{UAV}}\mathbf{p} - {}^{\mathrm{UAV}}\mathbf{M}_{\mathrm{c}}{}^{\mathrm{c}}\widehat{\mathbf{M}}_{\mathrm{w}}{}^{\mathrm{w}}\mathbf{s}^{*}) \tag{7.26}$$

Equations (7.24) and (7.26) describe the control laws that will be applied for controlling the UAV. The advantage of using this approach instead of an IBVS is that the control task is defined directly in cartesian coordinates, which are the coordinates used to move the UAV. However, PBVS relies on the pose estimation algorithm which depends on the camera calibration parameters, and therefore errors in the estimation of these values may cause positioning errors.

In [43], another PBVS scheme is presented. Where $\mathbf{s}$ is defined as ${}^{\mathrm{c}^{*}}\mathbf{s} = ({}^{\mathrm{c}^{*}}\mathbf{t}_{\mathrm{c}}, \theta\mathbf{u})$, and $\theta\mathbf{u}$ defines the angle/axis parameterization for the rotation. For this case, ${}^{\mathrm{c}^{*}}\mathbf{s}^{*} = 0$. Therefore, ${}^{\mathrm{c}^{*}}\mathbf{e} = {}^{\mathrm{c}^{*}}\mathbf{s}$. Formulating this PBVS scheme to the UAV case, we have that if $\mathbf{s}^{*}$ is defined in the world reference frame in such a way that it coincides with the origin of this coordinate system, ${}^{\mathrm{w}}\mathbf{s}^{*} = 0$. If $\mathbf{s}$ is the position of the UAV defined in the world reference frame ${}^{\mathrm{w}}\mathbf{s}$, then ${}^{\mathrm{w}}\mathbf{s} = ({}^{\mathrm{w}}\mathbf{t}_{\mathrm{UAV}}, {}^{\mathrm{w}}\mathbf{R}_{\mathrm{UAV}})$. In this case, the rotation is defined by the rotation matrix, although other parameterizations can be used. Therefore, the control law is defined as:

$$\begin{aligned} \boldsymbol{v}_{\mathrm{UAV}} &= -\lambda({}^{\mathrm{w}}\mathbf{R}_{\mathrm{UAV}}^{\top}{}^{\mathrm{w}}\mathbf{t}_{\mathrm{UAV}}) \\ \boldsymbol{\omega}_{\mathrm{UAV}} &= -\lambda({}^{\mathrm{w}}\mathbf{R}_{\mathrm{UAV}}^{\top}) \end{aligned} \tag{7.27}$$

In Equation (7.27), ${}^{\mathrm{w}}\mathbf{R}_{\mathrm{UAV}}$ and ${}^{\mathrm{w}}\mathbf{t}_{\mathrm{UAV}}$ are estimated using a pose estimation algorithm, as the one explained in Chapter 5. Stability analysis of these control schemes can be found in [43].

7.3.1. Results: PBVS with an On-Board Camera

In this section, the PBVS scheme has been tested using an eye-in-hand configuration, i.e. the camera is on-board the UAV. Simulation tests are conducted using a virtual environment based on the ROS (Robot Operating System) framework [155], the 3D simulator Gazebo [71], and the Starmac aircraft model [178]. Additionally, real flight tests are conducted using the AR.Drone-Parrot UAV [7] shown in Figure 7.5.

We present results of using a PBVS strategy for controlling the altitude of the UAV. As can be seen in Figure 7.8, the objective of the task when the PBVS strategy is used consists in moving the UAV to a desired 3D position. For the tests conducted in this section, we consider the cases where the setpoints $\mathbf{s}^*$ are constant, defined in the 3D space; and a vision-based pose estimation algorithm is in charge of providing the measurements of the position of the UAV, i.e. is in charge of estimating $\mathbf{s}$.

7.3.1.1. Simulation test

The test is conducted using a virtual environment that uses the ROS (Robot Operating System) framework. In Section 7.2.1, an IBVS strategy was used to center an object in the image plane sending velocity commands to a UAV in the $\mathrm{X_v}$ and $\mathrm{Y_v}$ axes. In this section, the task is complemented by adding commands in the $\mathrm{Z_v}$ axis based on position information. The altitude of the UAV is estimated based on the knowledge of the template size. The control task, therefore consists in locating the object of interest in the center of the image using the results obtained in Section 7.2.1, and seeks to bring the UAV to a defined altitude based on position-based commands, i.e. a hybrid approach that combines IBVS and PBVS. The object of interest is a flat template located on the ground (a helipad), the HMPMR-ICIA algorithm is in charge of tracking the helipad (i.e. is in charge of determining the 2D position of the helipad in the image plane), and the pose estimation algorithm explained in Chapter 5 is in charge of estimating the altitude of the UAV.

The control task of this test consists, first, in placing the quadrotor over the helipad at a fixed altitude (10 m), i.e. in locating the helipad in the center of the image plane (coordinate $(320, 240)$), using the IBVS approach explained in Section 7.2.1. When the helipad is close to the center, the vision-based altitude of the quadrotor, estimated using the method described in Section 5, is used to estimage the altitude commands that are sent to the flight controller in order to make the quadrotor descend to a defined position. In this last stage, both the control in the image plane and the altitude control operate simultaneously.

Therefore, for the PBVS task, the setpoint is defined as ${}^{\mathrm{w}}s^* = -1$ m in the world reference frame, which is located in the center of the object, as shown in Figure 7.8. ${}^{\mathrm{w}}\widehat{s}$ corresponds to the vision-based estimations of the altitude of the UAV. Thus, the velocity commands sent to the UAV are defined as follows:

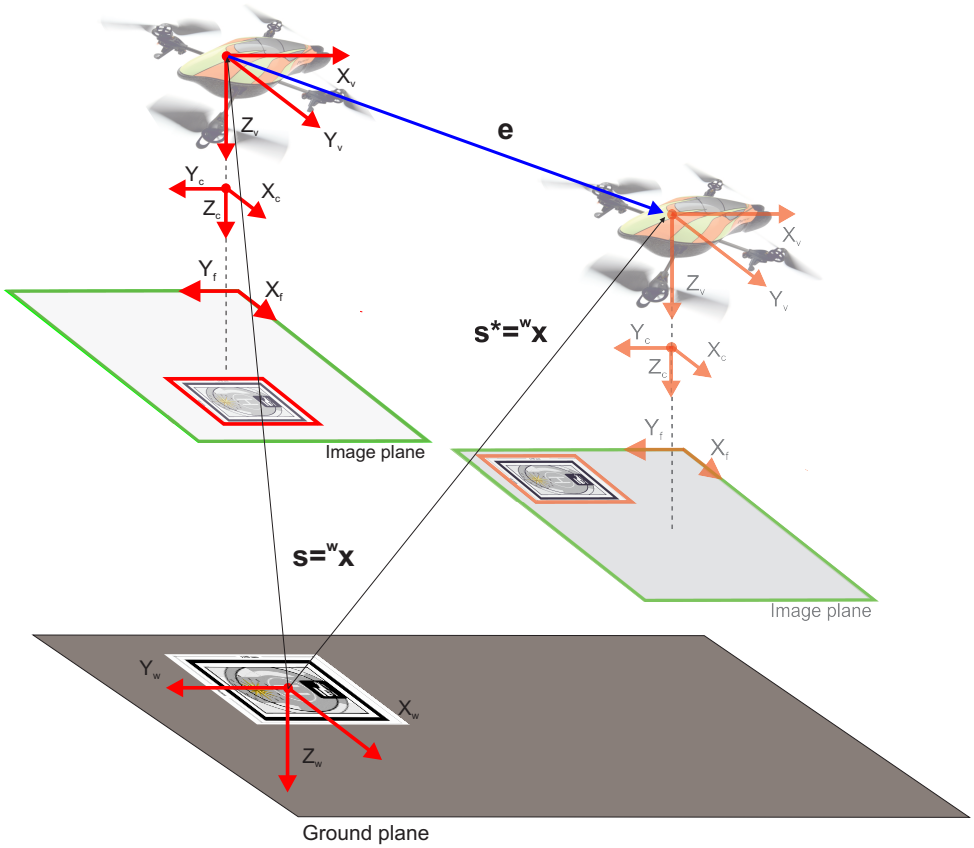


Figure 7.8: PBVS objective. The objective of the task consists in moving the UAV to a desired 3D position. We consider the case where the setpoints $\mathbf{s}^*$ are constant and defined in the 3D space; and where a vision-based pose estimation algorithm is in charge of providing the measurements of the position of the UAV, i.e. is in charge of estimating $\mathbf{s}$.

$$v_{z\,\mathrm{UAV}} = -\lambda^{\mathrm{UAV}}\widehat{M}_{\mathrm{w}}({}^{\mathrm{w}}\widehat{s} - {}^{\mathrm{w}}s^*) \tag{7.28}$$

where ${}^{\mathrm{UAV}}\widehat{M}_{\mathrm{w}} = -1$, and is in charge of converting the altitude from the world reference frame to the UAV coordinate system.

Figure 7.9 shows the trajectory of the quadrotor during the task. The blue line corresponds to the position data obtained by the Starmac-ros package [178]. In that figure, it can be seen that, first, the quadrotor moves towards the helipad in order to locate the helipad in the center of the image plane (setpoints: Sp ${}^{\mathrm{f}}x^* = 320$ pixels and Sp ${}^{\mathrm{f}}y^* = 240$ pixels), based on an IBVS strategy; and then starts to descend until it reaches the altitude setpoint (Sp ${}^{\mathrm{w}}z^* = -1$ m). When the quadrotor is descending, it continues maintaining the helipad centered in the image plane.

Figure 7.10 shows the first stage of the landing approach. As mentioned previously, this stage is conducted in the image plane based on the information of the center of the helipad recovered by the HMPMR-ICIA algorithm. In the graphic, it can be seen that the helipad reaches the setpoints in the $\mathrm{X_f}$ and $\mathrm{Y_f}$ axes (i.e. in the center of the image

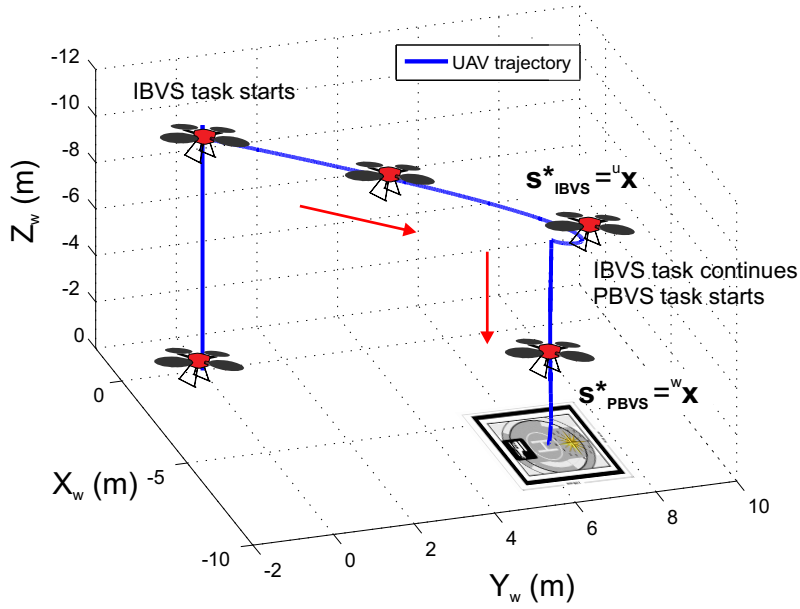


Figure 7.9: 3D Trajectory during the IBVS and PBVS tasks. First, the UAV moves using an IBVS strategy in order to center the helipad in the image plane. Then, a PBVS strategy is used to to control the altitude of the UAV. The blue line corresponds to the position data obtained by the Starmac-ros package [178].

plane) and remains centered during the task, i.e. the red/dark solid line reached the Sp ${}^{\mathrm{f}}x^* = 320$ pixels, and the magenta/light solid line reached the Sp ${}^{\mathrm{f}}y^* = 240$ pixels. In the figure, the setpoints in the $\mathrm{X_f}$ and $\mathrm{Y_f}$ axes are represented by the green/light dashed lines. The blue/dark dashed line is a control flag that indicates when the visual control in the $\mathrm{X_f}$ and $\mathrm{Y_f}$ axes (i.e. in the image plane) is operating. It can be seen that the image-based control was also operative during the descent process.

In the thumbnail images of Figure 7.10, it can be seen how the helipad was centered in the first part of the task (frames 300-4871) and then remained centered when the quadrotor was descending (frames 4872-7338).

On the other hand, Figure 7.11 shows the results of the position-based visual control task. When the helipad is centered (after frame 4871), a control flag is activated (blue/dark dashed line) and position-based control commands, based on the vision-based altitude estimation, are sent to the flight controller in order to make the quadrotor descend over the helipad to reach the ${}^{\mathrm{w}}z^* = -1$ m setpoint (frames 4872-7338). In Figure 7.11, it can be seen that the vision-based altitude estimations (red/dark solid line) have values and behaviors that are similar to the altitude estimated by the flight controller of the quadrotor (cyan/light solid line). The thumbnail images show that the HMPMR-ICIA algorithm tracked the template throughout the task, and that when the quadrotor was descending the helipad remained centered in the image plane.

7.3.2. Results: PBVS with a Ground Multi-Camera System

In this thesis, we propose the use of the trinocular system introduced in Section 6 as a landing platform for UAVs. The muti-camera system is in charge of recovering the

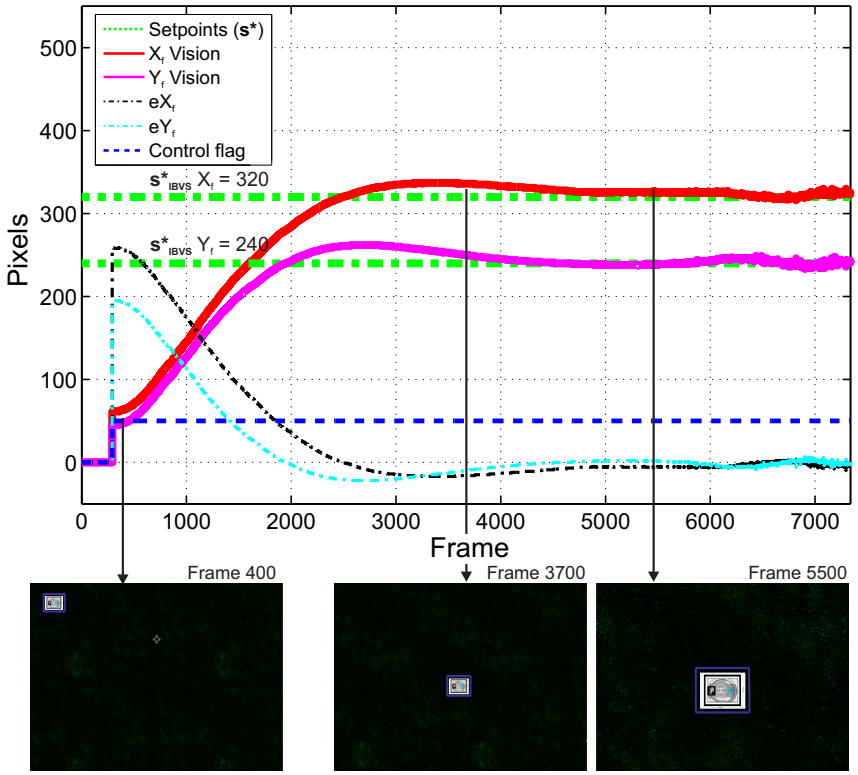


Figure 7.10: Simulation results, stage 1: IBVS. When the control flag is activated (blue/dark dashed line), the 2D position of the helipad in the image plane (red/dark solid line and the magenta/light solid line) is used to send image-based control commands to the quadrotor in order to center the helipad in the image plane (the green/light dashed lines represent the setpoints).

3D position and heading of the UAV based on the detection and tracking of color key features that are on-board. One of the advantages of using this system for landing taks is that the installation of additional on-board sensors for conducting basic tasks such as landing is not required, and therefore the limited capacity on-board can be well-exploited for other sensors needed for the mission. Additionally, as was shown in Section 6, with this system robust 3D information at real-time frame rates is obtained. This makes the system suitable to be used in vision-based control tasks.

In this section, the estimations obtained with the trinocular system are used to send vision-based commands to the UAV to conduct position-based visual servoing tasks.

7.3.2.1. Helicopter landing test

The trinocular system is used in this test to estimate the 3D position and heading of the UAV shown in Figure 4.21. The visual estimation is integrated with the UAV control loop using a PBVS strategy in a dynamic look-and-move architecture [92, 43]. The UAV testbed has a low-level controller based on PID control loops to ensure the helicopter' stability (see Appendix C for a detailed description of the UAV testbed). A client-server architecture working in a multi-client wireless network allows the integration of the vision system with the low level controller. The visual control system sends references to the

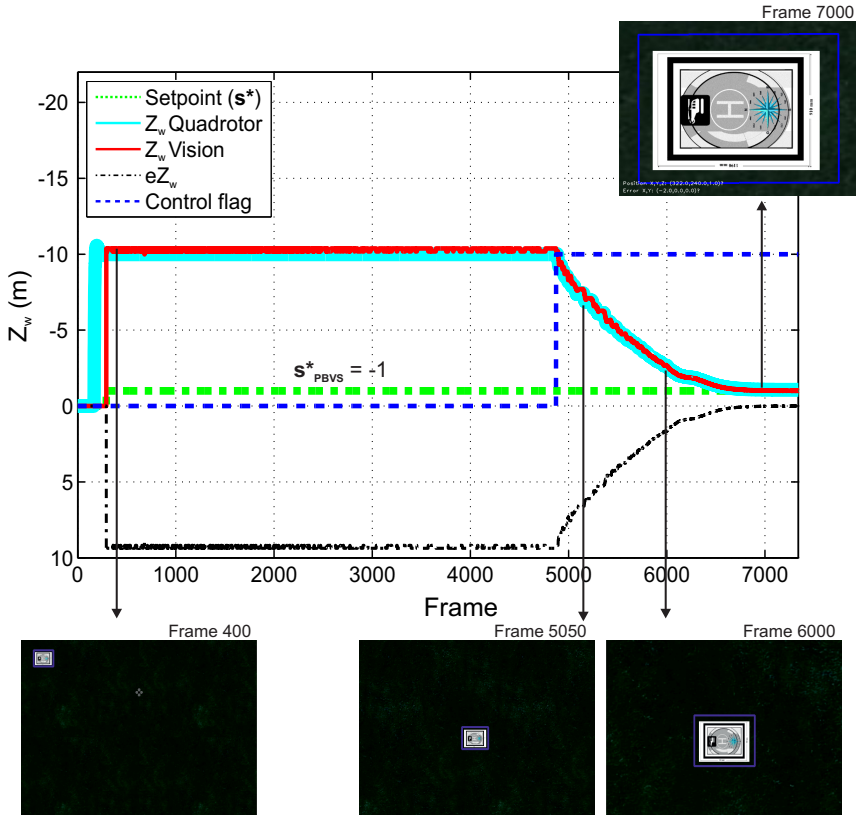


Figure 7.11: Simulation results, stage 2: PBVS. During the first stage IBVS, the altitude remains constant. Once the quadrotor is over the helipad, a control flag is activated (blue/dark dashed line), and the vision-based height estimations (red/dark solid line) are used to command the quadrotor to an altitude of 1 m (green/light dashed line) with respect to to the ground. The cyan/light solid line corresponds to the altitude estimated by the state estimator of the quadrotor.

flight controller using TCP/UDP messages [128].

As shown in Figure 7.12, the control task in this test consists in controlling the altitude of the UAV in order to achieve a vision-based landing. The desired position and the position information given by the trinocular system, both defined in the world coordinate system, will be compared to generate references to the position controller, as can be seen in Figure 7.12. These references are first transformed into commands to the helicopter by taking into account the helicopter's orientation, and then those references are sent to the helicopter's flight controller in order to move the UAV to the desired position.

The trinocular system described in Chapter 6 determines the position of the UAV in the world coordinate system. Therefore, according to the definition of the error function of the PBVS described in Equation (7.23), for this test ${}^{\mathrm{w}}\mathbf{s}^* = (0, 0, 0)$, although only the z coordinate is controlled; and ${}^{\mathrm{w}}\mathbf{s} = {}^{\mathrm{w}}\mathbf{x}$, where ${}^{\mathrm{w}}\mathbf{x} = {}^{\mathrm{w}}\mathbf{M}_{\mathrm{c}}{}^{\mathrm{c}}\mathbf{p}$, being ${}^{\mathrm{c}}\mathbf{p}$ the 3D coordinates of the center of gravity of the UAV with respect to the camera frame, and ${}^{\mathrm{w}}\mathbf{M}_{\mathrm{c}}$ the transformation from the camera to the world reference frame. All these trasnformations are explained in Chapter 6. The commands sent to the UAV are based on Equation (7.24), where ${}^{\mathrm{UAV}}\widehat{\mathbf{M}}_{\mathrm{w}}$ is defined by the rotation matrix ${}^{\mathrm{UAV}}\widehat{\mathbf{R}}_{\mathrm{w}}$, found in Chapter 6, that

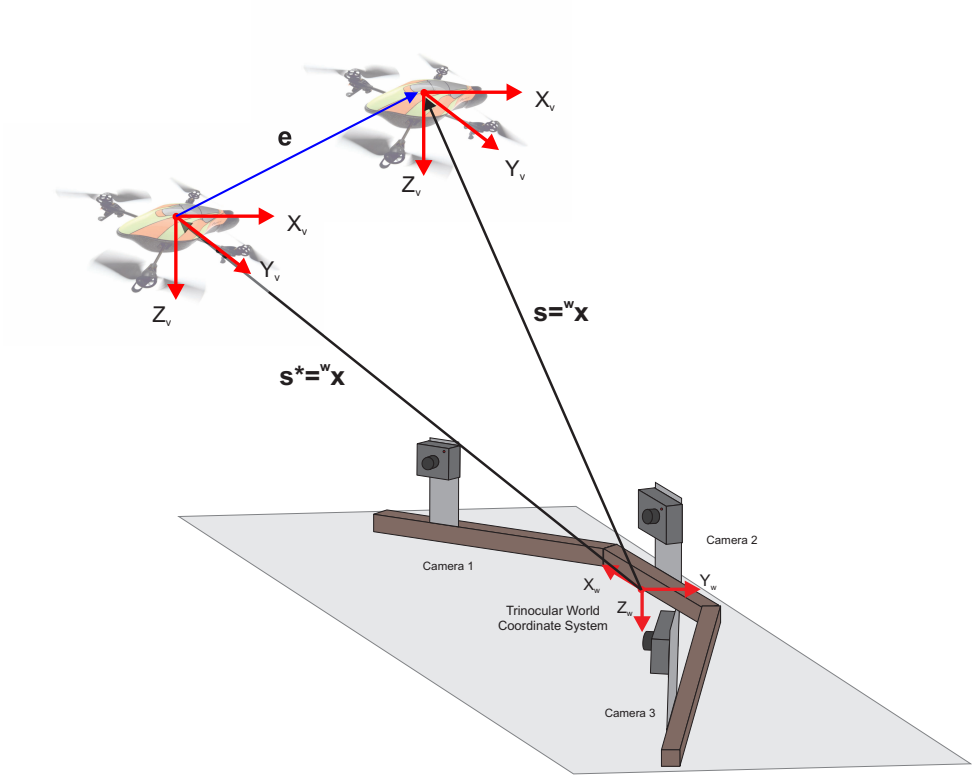


Figure 7.12: PBVS task for the ground trinocular system. The error function is defined by the 3D desired position expressed in the trinocular world coordinate system ($\mathbf{s}^*$), and the 3D position of the UAV ($\mathbf{s}$) which is estimated by a vision-based algorithm. With this, vision-based commands are sent to the UAV flight controller to make the helicopter move towards the desired position.

represents the heading of the UAV, and ${}^{w}\widehat{\mathbf{M}}_{c}{}^{c}\widehat{\mathbf{p}}$ is the position of the UAV with respect to the world reference frame found with the algorithm explained in Chapter 6.

In this section, results from two different landing tests are presented. Figure 7.13 shows the results of the first landing test. This figure compares the vision-based position estimations of the UAV for the X_w and Y_w axes (red/dark solid lines) with the position estimated by the GPS/IMU data (green/light solid lines). In the figure, it can be seen that the vision-based estimations have values and behavior that are similar to the ones obtained with the GPS/IMU data. Additionally, the thumbnail images permit to analyze the vision-based estimations. For the X_w estimation (plot on the left), it can be seen that at the beginning the UAV was aligned with the camera 2 of the trinocular system, i.e. in the origin of the X_w axis. Then, the UAV moved to the right (see Frame 2750), and then, when the UAV landed (e.g. Frame 2890), the estimated position in the X_w axis was also on the right side of the world coordinate system but farther than the initial position, as is reflected in the thumbnail image.

The same correlation among the vision-based estimations, the GPS/IMU estimations, and the thumbnail images, can be seen in the plot shown on the right side of Figure 7.13 (the Y_w estimation). The thumbnail images show that at the beginning of the test (Frame 2500) the UAV was away from the trinocular system. Then, as shown in Frame 2700,

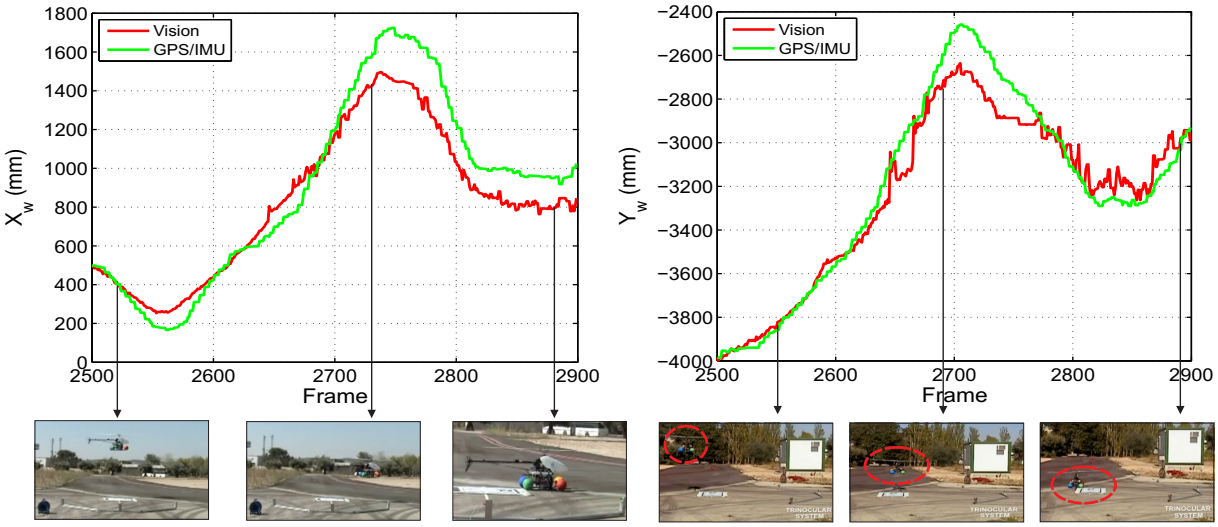


Figure 7.13: X_w and Y_w estimations for the landing test 1. The vision-based estimations obtained with the trinocular system (red/dark solid lines) are compared with the estimation from the GPS/IMU data (green/light solid lines). The thumbnail images show the correlation of the estimated motion with the real motion of the UAV.

the UAV went closer to it; and finally, when the UAV landed, the UAV was far from the trinocular system but not as far from it as at the beginning of the test. These motions are reflected not only in the plot but also in the thumbnail images.

Figure 7.14 shows the comparison of the vision-based estimation (red/dark solid line) with the GPS/IMU estimation (green/light solid line) of the altitude of the UAV. As was shown in Chapter 6, when the UAV is close to the ground, the accuracy of the GPS-based altitude estimation decreases. This can be seen in Figure 7.14, which shows that before the helicopter landed the GPS/IMU estimation shows the UAV as if it were underground (see Frame 2800). On the other hand, the blue/dark dashed line shows the evolution of the error during the test, and the black/dark solid line represents the setpoint, which for this test was defined as 0 . The thumbnail images shown in Figure 7.14 show that the vision-based estimations are consistent with the motions conducted by the UAV.

Figure 7.15 shows the results of the second landing test. In this test, the UAV was first flying in front of the trinocular system. Then, when the landing signal was activated (Frame 1120), vision-based control commands were sent to the UAV flight controller to conduct the landing task. Figure 7.15 compares the vision-based altitude data (red/dark solid line) with the GPS/IMU data (green/light solid line). As was shown in the previous test, when the UAV is close to the ground, the GPS-based altitude estimation does not coincide with the real altitude of the UAV. On the other hand the blue/dark dashed line shows the evolution of the error when the landing task was activated (Frame 1120). The thumbnail images shown in Figure 7.15 permit to analyze the vision-based altitude estimations. As can be seen, the behavior of the vision-based estimations coincides with the motion conducted by the UAV. When the helicopter was on the ground (see Frame 1550), the vision-based estimations were close to 0.

From the conducted test, it was possible to see that robust real-time vision-based estimations were obtained with the trinocular system, that permits to accomplish land-

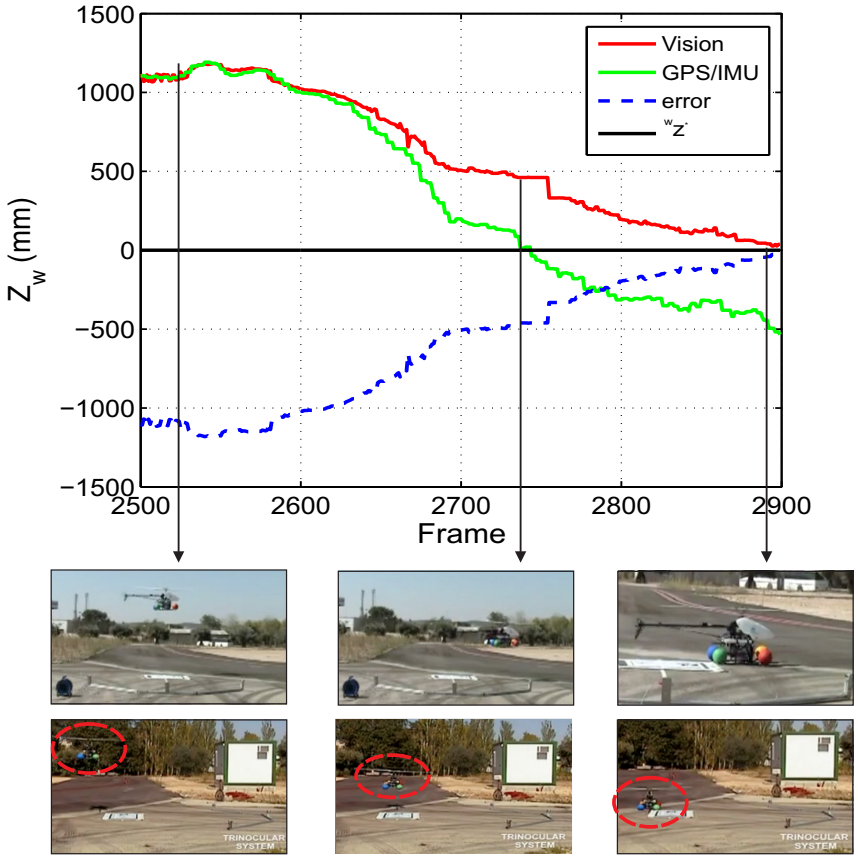


Figure 7.14: PBVS results of the landing test 1. A desired altitude ($z = 0$) is compared with the altitude of the helicopter, estimated by the trinocular system. Vision-based references are generated to make the helicopter move towards the desired position. The thumbnail images show the correlation of the vision-based estimations with the real motions of the UAV.

ing tasks successfully. Additionally, it was possible to see that the GPS-based altitude estimations are not reliable estimations for conducting those tasks, that require accurate altitude estimations.

7.4. Summary

In this chapter, visual servoing techniques have been reviewed and also formulated for the different tests conducted with the UAVs. Position-based visual servoing (PBVS) and image-based visual servoing (IBVS) approaches have been tested with both simulated data and real flight tests. IBVS tasks have the advantage of requiring only the position information in the image plane, i.e. they are based on the results of the tracking algorithm. Conversely, PBVS tasks depend not only on the tracking algorithm, but also on a pose estimation algorithm. Nonetheless, both position-based and image-based visual servoing techniques had satisfactory results in the tests that were conducted.

The main intention of this chapter was to test the capabilities of the visual algorithms, described in previous chapters, developed to sense and estimate the position of

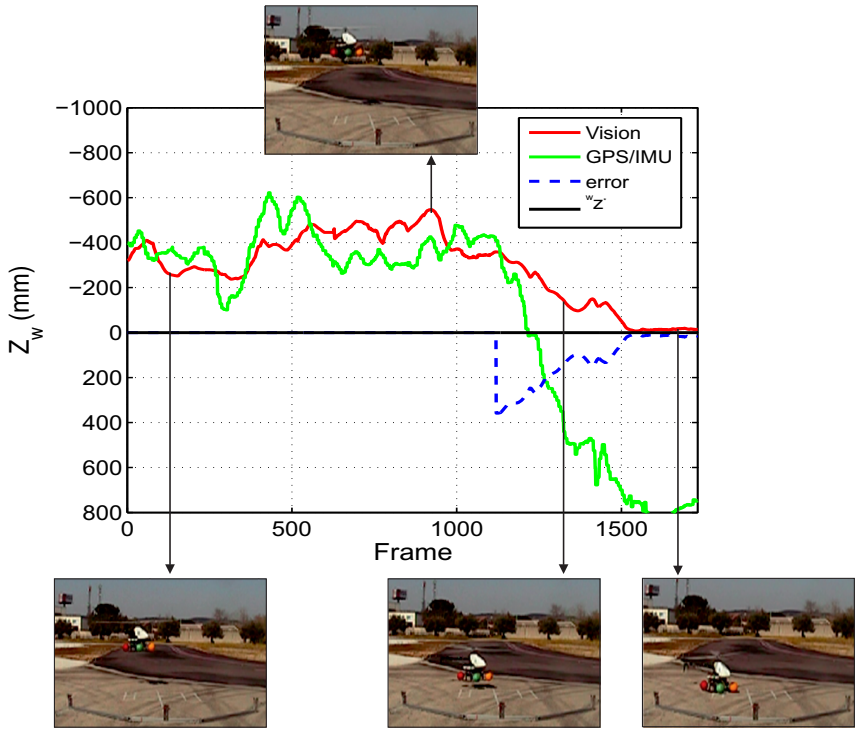


Figure 7.15: PBVS results of the landing test 2. A desired altitude ($z = 0$) is compared with the altitude of the helicopter, estimated by the trinocular system. Vision-based references are generated to make the helicopter move towards the desired position. The thumbnail images show the correlation of the vision-based estimations with the real motions of the UAV.

UAVs based on visual information. On-line tests represent a challenging scenario for any visual algorithm, and delays in communications, illumination changes, and real-time requirements are some of the problems found in this scenario. The tests that were conducted reveal that the tracking algorithm, the HMPMR-ICIA algorithm, and also the pose estimation algorithms, can be used for image-based and/or position-based control tasks. The speed reached when closing the loop with these algorithms was fast enough to successfully perform the tasks at real-time frame rates.

Chapter 8

Conclusions and Future Work

Cameras are a low-cost source of information that require the intervention of specific time-consuming techniques such as feature detection, feature tracking, 3D reconstruction, or pose estimation, among other techniques, in order to process, analyze, and understand the information contained in images. In most situations, vision-based solutions require all the previously mentioned steps, making the operation of systems in real-time a difficult task to achieve.

In this thesis, cameras have been used for augmenting UAVs' capabilities, and solutions for some of the problems that arise when a visual sensor is used to guide UAVs have been presented (i.e. real-time frame rates). This thesis has contributed to the state of the art by proposing robust and fast strategies that allow the use of vision algorithms in the field of UAVs. **Novel solutions for solving the tracking problem** of objects on-board UAVs, for **estimating the pose of the UAV** based on the visual information collected by cameras located either on the ground or on-board; and also novel applications of these techniques for solving different problems, such as **visual tracking for aerial refuelling, and vision-based landing**, among others, have been presented.

One of the objectives of this thesis was to propose a tracking strategy for tracking planar structures (or structures that can be assumed to be planar) with cameras on-board UAVs. In the literature reviewed in Chapter 2, it was found that the vast majority of approaches makes use of features to track objects. In this thesis, **we have proposed to address the tracking task with direct methods** [94], taking into account that with such methods robust motion estimations can be obtained, because the information of every pixel of the object to track is used in the motion estimation process. Direct

methods have good performances when the frame-to-frame motion is small. However, due to the amount of information that direct methods have to evaluate, these kinds of methods are mostly used in off-line applications. In this thesis, we presented solutions to the small frame-to-frame motion constraint of direct methods and also to the real-time problem of these methods, and therefore with the proposed algorithms we have been able to apply direct methods in demanding real-time applications.

In Chapter 4, **we proposed to use the ICIA algorithm** [14] **to deal with the efficiency problem of direct methods; and we also proposed to extend it with a hierarchical** multi-parametric and multi-resolution strategy that we have called and classified as **an HMPMR structure**, which is in charge of dealing with the small motion constraint of direct methods. We have shown that the resulting algorithm, the HMPMR-ICIA, can be used for obtaining robust motion estimations at real-time frame rates.

The HMPMR-ICIA algorithm has been successfully **applied to solve the tracking problem on-board UAVs and to solve the drogue tracking problem for autonomous air-to-air refuelling (AAAR) tasks** (based on the probe-and-drogue method). The proposed strategy has been tested with image data from real flight tests using a UAV, where the requirements of direct methods are easily unsatisfied due to vehicle vibrations and the problems caused by outdoors operations. For the aerial refuelling tasks, the proposed tracking algorithm has been tested in a robotic laboratory facility using real visual information from real refuelling hardware (a drogue and a probe). In both applications, the performance of the HMPMR-ICIA has been compared with state of the art algorithms based both on direct methods and on feature-based methods, using simulated data, aerial images, and publicly available datasets of planar templates, and also using different evaluation mechanisms in order to analyze the performance of the HMPMR-ICIA algorithm.

The results shown in Chapters 4 have shown that typical MR tracking approaches based on direct methods, and also approaches based on features, can easily fail in the presence of partial occlusions (when part of the template goes out of the FOV of the camera) and of strong changes in position. It has also been shown that the performance of a tracker based on direct methods is improved when the MP and MR hierarchies are fused, making it able to tolerate long and strong frame-to-frame motions. In the different tests conducted, we have found that **if the HMPMR strategy is adopted with the ICIA algorithm, it is possible to obtain robust estimations at real-time frame rates** (16 FPS, even when estimating 8 parameters), being the strategy robust under the presence of large motions, partial occlusions, and vehicle vibrations (without requiring any specific hardware or software for video stabilization), even when simple and complex motion models are estimated. Additionally, the results have shown that the tracking algorithm based on **the HMPMR-ICIA offer a potential solution for the probe and drogue autonomous aerial refuelling problem**. Although the results presented in this thesis are focuses on the application of tracking for aerial vehicles, the HMPMR strategy can be extended to other applications (e.g. mosaicing, motion detection and segmentation, pose estimation, etc), even using other image registration algorithms [115, 78, 17].

On the other hand, based on the advantages of using direct methods, and also based on the results obtained with the HMPMR-ICIA tracking algorithm, in Chapter 5 **we**

have proposed two pose estimation algorithms based on the image information captured by a camera on-board a UAV: one for recovering the state of UAVs, and a second one used for recovering the relative position between the UAV and an object of interest (the object that is tracked).

The first algorithm described in Section 5.3.1 is based on the relative motion obtained by decomposing the frame-to-frame homography estimated by the HMPMR-ICIA algorithm applied to a patch that covers around 80% of the image. The proposed strategy has been tested with real flight data in representative stages of a flight: cruise, landing, and take-off, being two of those stages considered critical: take-off and landing. The performance of the pose estimation strategy has been analyzed by comparing it with the GPS/IMU estimations. The results have shown correlation between the visual estimation and the GPS/IMU data, and it has been shown that the assumption of the planar surface to solve the homography decomposition ambiguity is valid in the three analyzed cases. This demonstrates that **the visual estimation can be used to provide a good temporal approximation of the vehicle's state when it is required (e.g. during GPS drop-outs) at real-time frame rates ($\approx$ 12 FPS), based only on visual information**.

A second pose estimation algorithm, described in Section 5.3, was proposed to estimate the relative position between the UAV and an object of interest. The algorithm is based on the results found by the HMPMR-ICIA tracking algorithm, assuming that the dimensions of the tracked object are known. The position in the image plane of the object of interest is used to calculate an euclidean homography which is then decomposed in order to obtain the 3D position of the UAV with respect to the object. In this thesis, this algorithm has been applied for estimating relative positions between the camera and the object that is being tracked as a way to compare the results of the tracking algorithm with known ground-truth data (e.g. VICON data). Additionally, this algorithm has been used for obtaining position data that is later required for position-based visual servoing tasks (positioning and landing), as was shown in Chapter 7. The results obtained in Sections 5.5 and 5.6 show that **there is correlation between the ground-truth data and the vision-based estimations**.

Therefore, based on the results obtained in Chapters 4 and 5, we conclude that the information that is recovered by **the proposed tracking and pose estimation algorithms, based on direct methods (i.e. based on the HMPMR-ICIA), can be used for obtaining important information (e.g. vehicles' state and relative position), robustly and fast enough to use it for vision-in-the-loop tasks**.

On the other hand, in Chapter 6 **we proposed a ground multi-camera system for estimating the pose of UAVs**. A novel low-cost multi-camera system was proposed to deal with the position estimation problem at low heights. In this scenario, tasks such as landing require more accurate and precise position estimations of the UAV than the ones obtained with GPS (GPS horizontal and vertical accuracies are of 2 m and $\pm$0.5 m, respectively). State of the art algorithms were implemented in order to estimate the UAV's position and orientation. In Section 6.3 the trinocular system was tested tested outdoors in real flight tests. The vision-based estimated data has been compared with the UAV's state information estimated with other on-board sensors. **The comparison has shown that the vision system improves the position estimation, especially**

the height estimation, whose current accuracy based on GPS is 0.5 m or lower when the vehicle approaches the ground. Additionally, in the tests it has been seen that **small variations in position are better perceived by the vision system, which makes this system suitable for maneuvers at low heights and close to structures** where precise movements are required.

Finally, in Chapter 7 the capabilities of the proposed tracking and pose estimation algorithms to obtain robust real-time estimations were analyzed. **On-line tests represent a challenging scenario for any visual algorithm**, because delays in communications or illumination changes can affect their performance. The tests that were conducted reveal that the tracking algorithm, the HMPMR-ICIA algorithm, and also the pose estimation algorithms, can be used for vision-based control tasks. The estimations and the speed reached when closing the control loop with these algorithms show that the algorithms were robust and fast enough to successfully perform tasks at real-time frame rates.

In sum, the different computer vision techniques that have been presented in this thesis can deal with some of the problems found when performing vision-based tasks on-board UAVs (vibrations, 3D motion, limited computational capacity on-board, etc.), and can also deal with some computer vision problems (sensibility to large motion, high processing requirements, etc.). In this thesis, different algorithms have been proposed to exploit the visual information extracted both from an on-board camera and from a novel multi-camera system. The performance of the proposed vision algorithms has shown to be of a standard that is **appropriate regarding the different explored applications: aerial refuelling (Sections 4.7 and 5.6), landing (Sections 7.3.1 and 7.3.2.1), and state estimation (Section 5.5). It is noteworthy that they have low computational overheads for a vision systems**.

8.1. Publications

The work derived from this thesis has been published in the following international journals and international conferences. Ranking information corresponds to the 2012 version of the Journal Citation Report data.

International Journals

- **HMPMR Strategy for Real-Time Tracking in Aerial Images, Using Direct Methods**. Carol Martínez, Pascual Campoy, Iván F. Mondragón, José Luis Sánchez, and Miguel Olivares-Méndez. Journal of Machine Vision and Applications. Springer. ISSN:0932-8092. **IF= 1.103, Q3**
  Under revision.

- **A Vision-Based Strategy for Autonomous Aerial Refuelling Tasks**. Carol Martínez, Thomas Richardson, Peter Thomas, Jonathan Luke du Bois, Pascual Campoy. Journal of Robotics and Autonomous Systems. 2013. Elsevier
  ISSN:0921-8890. **IF= 1.156, Q2**
  DOI = `http://dx.doi.org/10.1016/j.robot.2013.02.006`

- **A Hierarchical Tracking Strategy for Vision-Based Applications On-Board UAVs**. Carol Martínez, Iván F. Mondragón, Pascual Campoy, José Luis Sánchez, and Miguel Olivares-Méndez. Journal Intelligent and Robotic Systems. 2013. Springer. ISSN:0921-0296. **IF= 0.827, Q3**
  DOI = `http://dx.doi.org/10.1007/s10846-013-9814-x`

- **On-board and Ground Visual Pose Estimation Technique for UAV Control**. Carol Martínez, Iván F. Mondragón, Miguel A. Olivares-Méndez, and Pascual Campoy. Journal Intelligent and Robotic System. Volume 61, Issue 1-4. 2011 Springer. ISSN:0921-0296. **IF= 0.827, Q3**
  DOI = `http://dx.doi.org/10.1007/s10846-010-9505-9`

Book Chapter

- **Visual Servoing for UAVs**. Pascual Campoy, Iván F. Mondragón, Miguel A. Olivares-Méndez, and Carol Martínez. Book chapter: Visual Servoing, Intechweb.org Online Publication 2010.
  DOI = `http://dx.doi.org/10.5772/187`

International conferences

- **Towards Autonomous Air-to-Air Refuelling for UAVs Using Visual Information**. Carol Martínez, Thomas Richardson, and Pascual Campoy. IEEE International Conference on Robotics and Automation ICRA 2013, Karlsruhe, Germany. May 6 - 11

- **A Hierarchical Strategy for Real-Time Tracking On-board UAVs** Carol Martínez, Pascual Campoy, Iván Mondragón, Jose Luis Sánchez-Lopez, Miguel A. Olivares-Méndez. The 2012 International Conference on Unmanned Aircraft Systems ICUAS'12, June 12-15, 2012. Philadelphia, PA USA.

- **A Multi-resolution Image Alignment Technique Based on Direct Methods for Pose Estimation of Aerial Vehicles**. Carol Martínez, Luis Mejías, and Pascual Campoy. International Conference on Digital Image Computing DICTA. December, 6-8, 2011. Queensland Australia.
  DOI = `http://dx.doi.org/10.1109/DICTA.2011.97`

- **On-board and Ground Visual Pose Estimation Technique for UAV Control**. Carol Martínez, Iván F. Mondragón, Miguel A. Olivares-Méndez, and Pascual Campoy. 3rd International Symposium on Unmanned Aerial Vehicles UAV'10, June 2010. Dubai, Arab Emirates

- **Trinocular ground system to control UAVs**. Carol Martínez, Pascual Campoy, Iván Mondragón, Miguel Olivares. IEEE/RSJ International Conference on Intelligent Robots and Systems. October 2009. St. Louis, MO, USA
  DOI = `http://dx.doi.org/10.1109/IROS.2009.5354489`

8.2. Future Work

This thesis has shown how computer vision can play an important role in the acquisition of significant information for UAVs. The different algorithms that were presented have proved to be capable of solving some important computer vision problems and also some of the problems found when addressing vision-based tasks on-board UAVs. All the different algorithms implemented in this thesis have their own limitations. Most of the limitations correspond to those situations where the assumptions made in the algorithms are not satisfied, and all these situations are the basis of future work in the different areas explored in this thesis.

In Chapter 4 it was found that, by using the HMPMR-ICIA algorithm, a robust and a fast visual tracking algorithm for tracking planar structures can be obtained. In this thesis we have presented solutions for one of the direct methods' constraints: the HMPMR strategy was proposed to deal with the small frame-to-frame motion constraint of direct methods. However, there is another assumption that is not directly examined in this thesis: the appearance constancy constraint (the intensity values of the object must be constant), although in our tests this was not a critical factor. Nonetheless, considering the inherent appearance changes that take place in outdoors operations (illumination changes), solutions to this problem must be proposed, so that the tracking algorithm can adapt to these changes. Possible solutions to this problem include exploring a template update strategy [97], to work with normalized images, or to include an illumination model in the formulation of the tracking task [78, 120], among others.

Under large frame-to-frame motions, the HMPMR suggests using as many MR levels as possible in order to cope with this large motion. This can lead to a slow performance of the algorithm (the minimization is conducted in each level). In order to speed up the algorithm, criteria to control the performance of the direct method inside the MR structure can be explored. Therefore, based on these criteria, it wil be possible to formulate a dynamic strategy, that decides, according to the results obtained in the lower resolution levels (e.g. based on the error), which other levels of the MR and MP structure are required for estimating the motion. For instance, if we have an MR pyramid with 5 levels, and the motion of the object is small, maybe conducting the minimization only in the lowest and in the highest resolution levels would be enough to estimate the motion of the object. This dynamic strategy could help speeding up the algorithm. Another possible solution to improve the speed is changing the number of pixels used in each level to estimate the motion, e.g. by selecting an aleatory number in each level, for example to use every 5 pixels in the highest resolution level; by using only the information with high image contrast as in [31]; or by conducting selective pixel integration [54] in order to estimate the motion.

Other important characteristics for a tracking algorithm that have not been well studied in the literature are the capabilities to robustly identify when the tracking algorithm has failed [202], and to estimate when it will fail. In this sense, in [126], we have presented some criteria to control the performance of the tracking algorithm, and although satisfactory results were obtained, with those criteria robust and more general strategies have to be formulated. Visual tracking is a very difficult problem that requires robustness not just for tracking but also for dealing with situations when the algorithm fails

or when the initial conditions of the algorithms change. In this sense, object detection algorithms have to be explored and implemented to be able to start and re-initialize the tracking algorithms whenever this is required. Additionally, machine learning approaches [11, 209] can be used for dealing with the recognition problem and also for dealing with the adaptation stage that is required to tackle occlusion problems.

Drift is another problem that needs further studies. As was shown in Sections 4.4.1 and 5.3.1, direct methods are known to obtain robust estimations and to drift slowly over time. Nonetheless, the tracking and the pose estimation strategies presented in this thesis are based on the integration of previous estimations, and therefore the performance of the algorithms can be affected by drift. For the tracking strategy, a possible solution can be based on a template update strategy, where future work can be oriented to establishing criteria to define when and how the template must be updated. Other solutions can be based on machine learning, and on the use of other sensors such as GPS/IMU information [93], in order to correct the position of the object in the image plane (sensor fusion), and therefore to deal with possible problems that may emerge due to the propagation of the parameters throughout the image sequence, especially when the template's appearance changes notoriously. In the case of the pose estimation algorithm, solutions based on the use of georeferenced images to correct the drift by matching these images with the captured images can be used [48, 41].

In general, this thesis dealt with three main computer vision areas: visual tracking, pose estimation, and visual control; and each of them can be the object of possible avenues for future research. Some of those possibilities were mentioned in the previous paragraphs. The computer vision areas dealt in this thesis were explored with the purpose of applying vision for UAVs. We have presented some applications where vision can play an important role following objects, estimating the state of the UAV, and/or estimating relative positions with respect to an object of interest. Once robust estimations are obtained, they can be used in a wide range of UAVs applications: obstacle detection and avoidance, surveillance, inspection, firefighting, etc. However, one of the most difficult aspects where research efforts are still required has to do with the ability of UAVs to adapt to the different conditions of missions. Concerning this, sensor fusion and machine learning algorithms appear as key factors to find possible solutions to this problem.

.

Appendix A

Camera Model

A.1. Intrinsic Parameters

The simplest model is the pinhole camera model. It states that a 3D point expressed in the camera coordinate system ${}^{\mathrm{c}}\mathbf{p} = [{}^{\mathrm{c}}x, {}^{\mathrm{c}}y, {}^{\mathrm{c}}z]^{\mathrm{T}}$ is projected onto the image plane in the point ${}^{\mathrm{f}}\mathbf{p} = [{}^{\mathrm{f}}x, {}^{\mathrm{f}}y]^{\mathrm{T}}$ by intersecting the ray that links the 3D point ${}^{\mathrm{c}}\mathbf{p}$ with the center of projection and the image plane [81](see Figure A.1).

By similar triangles (see Figure A.1 (b)), it is possible to transform the point ${}^{\mathrm{c}}\mathbf{p}$, expressed in the camera coordinate system, into the point ${}^{\mathrm{u}}\mathbf{p}$ expressed in central coordinates, as follows:

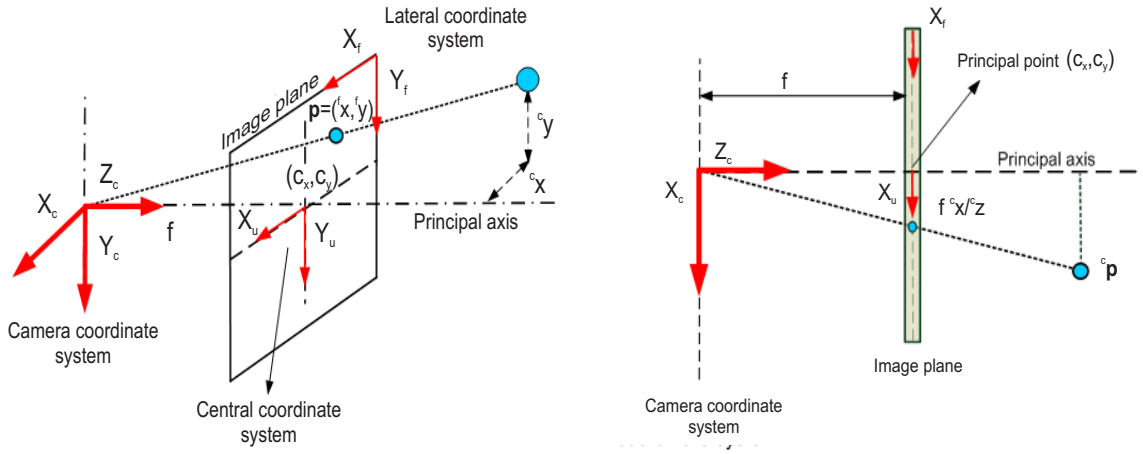
$$\frac{{}^{\mathrm{u}}x}{f} = \frac{{}^{\mathrm{c}}x}{{}^{\mathrm{c}}z}, \qquad \frac{{}^{\mathrm{u}}y}{f} = \frac{{}^{\mathrm{c}}y}{{}^{\mathrm{c}}z} \tag{A.1}$$

Using homogeneous coordinates, Equation (A.1) can be expressed in matrix notation, as follows:

$$\begin{bmatrix} n\,{}^{\mathrm{u}}x \\ n\,{}^{\mathrm{u}}y \\ n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} {}^{\mathrm{c}}x \\ {}^{\mathrm{c}}y \\ {}^{\mathrm{c}}z \\ 1 \end{bmatrix} \tag{A.2}$$

where f represents the focal length in mm. The transformation from the central coordinate system to the lateral coordinate system is achieved, as follows:

$${}^{\mathrm{f}}x = {}^{\mathrm{u}}x\, m_x + c_x, \qquad {}^{\mathrm{f}}y = {}^{\mathrm{u}}y\, m_y + c_y \tag{A.3}$$



(a) (b)

Figure A.1: Pinhole camera model. Different coordinate systems are used to find the relationship between a 3D point and its corresponding 2D projection onto the image plane.

where $({}^{\mathrm{f}}x,\ {}^{\mathrm{f}}y)$ are the coordinates of the point in the lateral coordinate system (in pixels), $(c_x,\ c_y)$ are the coordinates of the image center (in pixels), f is the focal length (mm), and $(m_x,\ m_y)$ are the number of pixels per unit distance in the X and Y directions.

Taking into account the previous transformation, the projection of a 3D point expressed in the camera coordinate system to a point in the image plane can be defined as follows:

$$\begin{bmatrix} n\,{}^{\mathrm{f}}x \\ n\,{}^{\mathrm{f}}y \\ n \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{\mathrm{c}}x \\ {}^{\mathrm{c}}y \\ {}^{\mathrm{c}}z \\ 1 \end{bmatrix} \tag{A.4}$$

$$ {}^{\mathrm{f}}\mathbf{p} = \mathbf{K}\,{}^{\mathrm{c}}\mathbf{p} $$

where $\mathbf{K}$ is the matrix that contains the intrinsic camera parameters: the coordinates of the image center $(c_x,\ c_y)$, and the focal length in pixel dimensions $({}^{\mathrm{f}}x,\ {}^{\mathrm{f}}y)$, being $f_x = f\,m_x$, and $f_y = f\,m_y$.

A.2. Extrinsic Parameters

The camera coordinate system is often unknown. However, points in space can be expressed in terms of an arbitrary system to make it possible to obtain the parameters that define the camera coordinate system with respect to a known world coordinate system, as shown in Figure A.2.

The two coordinate systems are related by a rotation matrix $\mathbf{R}$ that aligns the two coordinate frames, and by a translation vector $\mathbf{t}$ that links the origins of the coordinate systems. These parameters are called the extrinsic camera parameters, and they allow to

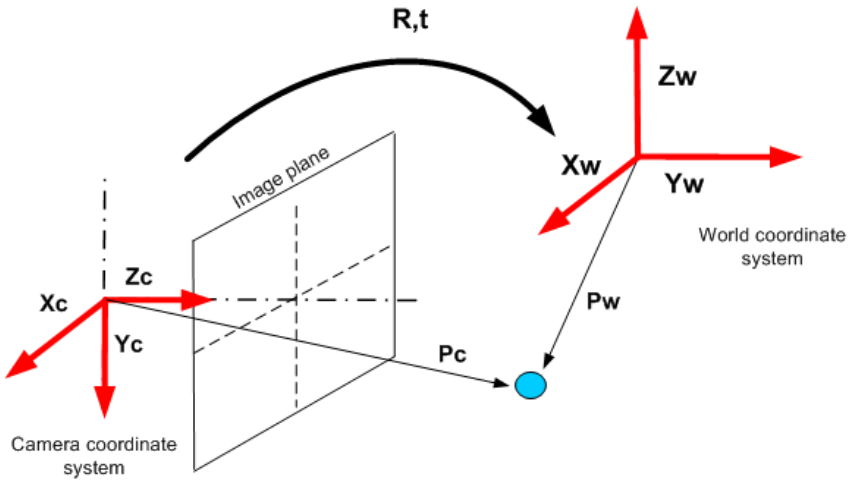


Figure A.2: Extrinsic camera parameters. The camera and the world coordinate systems are related by a rigid transformation, which is defined by a rotation matrix $\mathbf{R}$ and by a translation vector $\mathbf{t}$, that link the two coordinate systems.

define a point ${}^{w}\mathbf{p}$ in the world coordinate system in the camera coordinate system ${}^{c}\mathbf{p}$, as follows:

$$ {}^{c}\mathbf{p} = [\mathbf{R}|\mathbf{t}]{}^{w}\mathbf{p} $$

A.3. The Complete Model

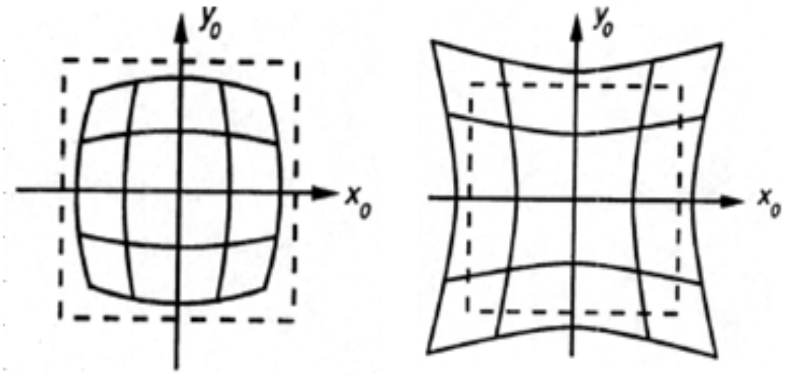
The previous transformations correspond to the pinhole camera model. This model assumed an ideal pinhole camera and does not take into consideration geometric distortions caused by the use of real lenses. In order to complete this model, geometric distortions have to be included. Distortions correspond to optic deficiencies called aberrations, which cause a degradation of the final image. Contrary to other aberrations, distortions do not affect the image quality, but they have a significant impact on the image geometry.

Therefore,

$$ \begin{aligned} {}^{u}x_d &= {}^{u}x + Dx({}^{u}x, {}^{u}y) \\ {}^{u}y_d &= {}^{u}y + Dy({}^{u}x, {}^{u}y) \end{aligned} \tag{A.5} $$

where ${}^{u}x_d$ and ${}^{u}y_d$ are the distorted coordinates; ${}^{u}x$ and ${}^{u}y$ are the undistorted coordinates, and $\mathrm{D}(x, y)$ is the distortion that includes the radial and tangential distortions.

The radial distortion is a result of the lens' imperfections. The effects of radial distortion are that straight lines are bent (curved) and that points are moved in the radial direction from their correct position. In Figure A.3, it is possible to see the types of radial distortion (negative and positive). Radial distortion can be characterized by two or more parameters (k_1, k_2, k_3) that represent the distortion coefficients and depend on the characteristics of the lenses. On the other hand, the tangential distortion is the result of assembling imperfections of the lens, and is represented by the parameters p_1, p_2. Equa-



(a) (b)

Figure A.3: RadialDistortion. (a). Barrel distortion (negative): points are moved from their correct position towards the image center. (b) Pincushion distortion (positive): points are displaced further away from the optical axis.

tion (labeleq:AddingRadialDistorion) can be expanded with the distortion parameters, as follows:

$$\begin{aligned} {}^{\mathrm{u}}x_d &= {}^{\mathrm{u}}x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + [2p_1 {}^{\mathrm{u}}x {}^{\mathrm{u}}y + p_2(r^2 + 2\,{}^{\mathrm{u}}x^2)] \\ {}^{\mathrm{u}}y_d &= {}^{\mathrm{u}}y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + [2p_2 {}^{\mathrm{u}}x {}^{\mathrm{u}}y + p_1(r^2 + 2\,{}^{\mathrm{u}}x^2)] \end{aligned} \tag{A.6}$$

where $r^2 = {}^{\mathrm{u}}x^2 + {}^{\mathrm{u}}y^2$; k_1, k_2 and k_3 are the radial distortion parameters; p_1, p_2 are the tangential distortion parameters; ${}^{\mathrm{u}}x_d$ and ${}^{\mathrm{u}}y_d$ are the distorted coordinates; and ${}^{\mathrm{u}}x$ and ${}^{\mathrm{u}}y$ are the undistorted coordinates.

Once the distortion parameters have been added to the coordinates of the pinhole model, then the distorted point is transformed into the lateral coordinate system in the image plane, as follows:

$${}^{\mathrm{f}}x = {}^{\mathrm{u}}x_d m_x + c_x, \qquad {}^{\mathrm{f}}y = {}^{\mathrm{u}}y_d m_y + c_y \tag{A.7}$$

where ${}^{\mathrm{f}}x$ and ${}^{\mathrm{f}}y$ are the coordinates of the point in the lateral coordinate sytem; ${}^{\mathrm{u}}x_d$ and ${}^{\mathrm{u}}y_d$ are the coordinates of the distorted point expressed in the central camera coordinate system; m_x and m_y are the number of pixels per unit distance in the x and y directions; and c_x and c_y are the coordinates of the image center.

Appendix B

Camera Calibration

The calibration process corresponds to the computation of the intrinsic and extrinsic camera parameters.

1. Intrinsic parameters:
   - Coordinates of the center of projection: c_x, c_y(pixels)
   - Focal length in terms of pixel dimensions: f_x, f_y ($f_x = fm_x$ and $f_y = fm_y$), being m_x and m_y the number of pixels per unit distance.
   - Radial distortion coefficients: r_1, r_2, r_3.
   - Tangential distortion coefficients: p_1, p_2.
2. Extrinsic parameters:
   - Rotation matrix $\mathbf{R}$.
   - Translation vector $\mathbf{T}$.

The calculation of the parameters is done using *Zhang's* algorithm [211]. This algorithm calculates the camera parameters using the information of multiple views of a planar object (see Figure B.1). The algorithm's steps are as follows:

1. Calculation of homographies between images $\mathbf{H}$.
2. Calculation of intrinsic parameters with multiple views.

3. Calculation of extrinsic parameters for each view.

4. Global optimization considering distortion.

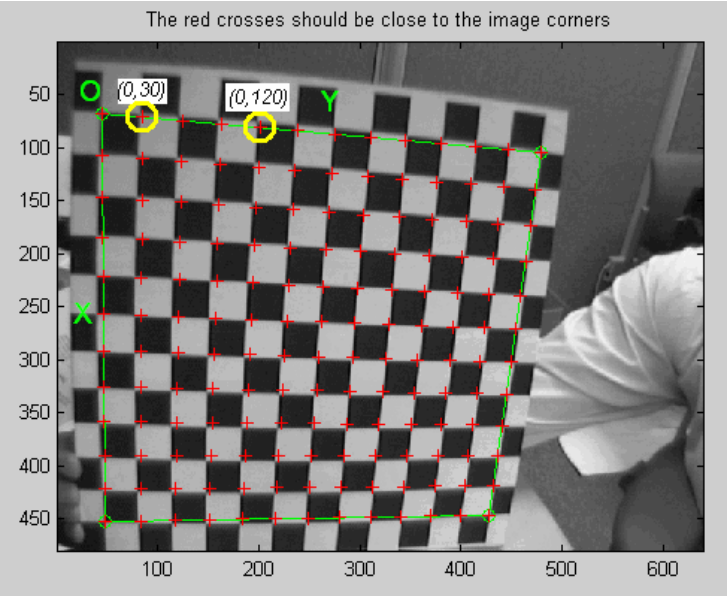


Figure B.1: Calibration pattern. Image from [23].

From Equations (A.2) and (A.4), we have that a point ${}^{\mathrm{w}}\mathbf{p}$ in the world reference system is projected on the image plane, as:

$$ {}^{\mathrm{f}}\mathbf{p} = \mathbf{K}[\mathbf{R}|\mathbf{t}]{}^{\mathrm{w}}\mathbf{p} \tag{B.1} $$

Taking into account that the world coordinate system is located in the calibration pattern, then the corners of the calibration patter have ${}^{\mathrm{w}}z = 0$, as shown in Figure B.1. Then Equation (B.1) can be written as:

$$ \begin{bmatrix} n{}^{\mathrm{w}}x \\ n{}^{\mathrm{w}}y \\ n \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{r_3} & \mathbf{t} \end{bmatrix} \begin{bmatrix} {}^{\mathrm{w}}x \\ {}^{\mathrm{w}}y \\ 0 \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix} \begin{bmatrix} {}^{\mathrm{w}}x \\ {}^{\mathrm{w}}y \\ 1 \end{bmatrix} \tag{B.2} $$

Therefore, ${}^{\mathrm{w}}\mathbf{p}$ is related with ${}^{\mathrm{f}}\mathbf{p}$ by a homography matrix $\mathbf{H}$ (3×3), defined up to a scale factor as shown in Equation (B.4), where $\lambda = 1/n$. This matrix can be computed with four matched points (8 equations). However, more correspondences are preferred for increasing the robustness in the estimation of the parameters.

$$ n \begin{bmatrix} {}^{\mathrm{f}}x \\ {}^{\mathrm{f}}y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} {}^{\mathrm{w}}x \\ {}^{\mathrm{w}}y \\ 1 \end{bmatrix} \tag{B.3} $$

$$\mathbf{H} = \begin{bmatrix} \mathbf{h_1} & \mathbf{h_2} & \mathbf{h_3} \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix} \tag{B.4}$$

Decomposing Equation B.4, we have that:

$$\begin{aligned} \mathbf{r_1} &= \lambda \mathbf{K}^{-1} \mathbf{h_1} \\ \mathbf{r_2} &= \lambda \mathbf{K}^{-1} \mathbf{h_2} \\ \mathbf{t} &= \lambda \mathbf{K}^{-1} \mathbf{h_3} \end{aligned} \tag{B.5}$$

Using the knowledge that $\mathbf{r_1}$ and $\mathbf{r_2}$ are orthogonal to each other, and extracting the scale factor, we have that $\mathbf{r_1}$ and $\mathbf{r_2}$ are orthonormal. This implies that the rotation vector's dot product is 0 (Equation (B.6)), and that the vectors' magnitudes are equal (Equation (B.7)).

$$\mathbf{r_1}^{\mathrm{T}} \mathbf{r_2} = 0 \tag{B.6}$$

$$\|\mathbf{r_1}\| = \|\mathbf{r_2}\| \qquad \text{or} \qquad \mathbf{r_1}^{\mathrm{T}} \mathbf{r_1} = \mathbf{r_2}^{\mathrm{T}} \mathbf{r_2} \tag{B.7}$$

Replacing the previous equations with the information in Equation B.5, we have as a first constraint that:

$$\mathbf{h_1}^{\mathrm{T}} \mathbf{K}^{-\mathrm{T}} \mathbf{K}^{-1} \mathbf{h_2} = 0 \tag{B.8}$$

and as a second one, that:

$$\mathbf{h_1}^{\mathrm{T}} \mathbf{K}^{-\mathrm{T}} \mathbf{K}^{-1} \mathbf{h_1} = \mathbf{h_2}^{\mathrm{T}} \mathbf{K}^{-\mathrm{T}} \mathbf{K}^{-1} \mathbf{h_2} \tag{B.9}$$

Where $\mathbf{K}^{-\mathrm{T}}$ means $(\mathbf{K}^{-1})^{\mathrm{T}}$. Using the above-mentioned constraints, it is possible to calculate the parameters of the $\mathbf{K}$ matrix by using an analytic solution followed by a non-linear optimization.

If we express $\mathbf{K}^{-\mathrm{T}} \mathbf{K}^{-1}$ as follows:

$$\mathbf{K}^{-\mathrm{T}} \mathbf{K}^{-1} = \mathbf{B} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

where $\mathbf{B}$ is a symmetric matrix whose elements are (taking into account only the intrinsic parameters c_x, c_y, f_x, f_y):

$$\mathbf{B} = \begin{bmatrix} \frac{1}{f_x^2} & 0 & \frac{-c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & \frac{-c_y}{f_y^2} \\ \frac{-c_x}{f_x^2} & \frac{-c_y}{f_y^2} & \frac{-c_x}{f_x^2} + \frac{-c_y}{f_y^2} + 1 \end{bmatrix} \tag{B.10}$$

Then, $\mathbf{B}$ can be written as a six-dimensional vector:

$$\mathbf{b} = \left[\; B_{11}, \quad B_{12}, \quad B_{22}, \quad B_{13}, \quad B_{23}, \quad B_{33} \; \right]^{\mathrm{T}} \tag{B.11}$$

If $\mathbf{h_i} = [\mathbf{h_{i,1}}, \mathbf{h_{i2}}, \mathbf{h_{i3}}]^{\mathrm{T}}$ is the i^{th} column vector of $\mathbf{H}$, then we have that:

$$\mathbf{h_i}^{\mathrm{T}} \mathbf{B} \mathbf{h_j} = \mathbf{v_{ij}}^{\mathrm{T}} \mathbf{b} = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix}^T \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{bmatrix}^T \tag{B.12}$$

Therefore, the constraints of Equations B.8 and B.9 can be written in the following form:

$$\begin{bmatrix} \mathbf{v}_{12}^{\mathrm{T}} \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^{\mathrm{T}} \end{bmatrix} \mathbf{b} = \mathbf{0} \tag{B.13}$$

If k images are used in the calibration process, then we can have k equations, such as the previous one, that can be written as $\mathbf{Vb} = \mathbf{0}$, where $\mathbf{V}$ is a $2k \times 6$ matrix. If $k \geq 3$, we will have a unique solution for the $\mathbf{b}$ matrix defined up to a scale factor, but if we impose the skewless constraint $\gamma = 0$ (that is the case of Equation (B.13)), then we will have a solution for $\mathbf{b}$ with $k \geq 2$.

The solution of a system such as the previously mentioned one corresponds to the eigenvector of $\mathbf{V}^{\mathrm{T}}\mathbf{V}$ associated with the smallest eigenvalue. Once $\mathbf{b}$ is estimated, it is possible to compute all the intrinsic camera parameters.

$$f_x = \sqrt{\lambda / B_{11}} \tag{B.14}$$

$$f_y = \sqrt{\lambda B_{11} / (B_{11}B_{22} - B_{12}^2)} \tag{B.15}$$

$$c_x = -B_{13} f_x^2 / \lambda \tag{B.16}$$

$$c_y = (B_{12}B_{13} - B_{11}B_{23}) / (B_{11}B_{22} - B_{12}^2) \tag{B.17}$$

Once the intrinsic parameters are found (matrix K is known), the extrinsic parameters can be calculated using Equation B.5. Hence, we have that:

$$\mathbf{r_1} = \lambda \mathbf{K^{-1}} \mathbf{h_1}, \quad \mathbf{r_2} = \lambda \mathbf{K^{-1}} \mathbf{h_2}, \quad \mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}, \quad \mathbf{t} = \mathbf{K^{-1}} \mathbf{h_3}$$

with $\lambda = 1/\|\mathbf{K^{-1}h_1}\| = 1/\|\mathbf{K^{-1}h_2}\|$.The previously calculated parameters are used as an initial guess for a refinement stage through iterative techniques in which, additionally, the distortion parameters are calculated as well.

Appendix C

Autonomous Vehicles' Specifications

In this appendix, there is a brief description of specifications of the different UAVs used to conduct the tests of this theses. Most of the information has been taken from previous thesis from the Computer Vision Group [140, 153, 135].

C.1. Autonomous Helicopter

The different movements of a helicopter are generated actuating on the attitude angles and the elevation force. The helicopter can be controlled through three commands, as shown in Figure C.1 [55]:

- Cyclic pitch (lateral and longitudinal): this control tilts the main rotor disk (swashplate) to a specific direction, changing the angle of attack of the rotor blades cyclically to obtain forward, backward and sideward movement directions of the helicopter.
- Collective pitch of the tail rotor: this control changes the angle of the tail rotor blades. The tail rotor blades provide enough thrust to the side that keeps the helicopter pointing in one direction. By increasing or decreasing the pitch of the tail rotor blades, the direction of the helicopter can be changed.
- Collective pitch of the main rotor: this command changes the pitch angle of the main rotor blades collectively (i.e. all at the same time) and independently of their

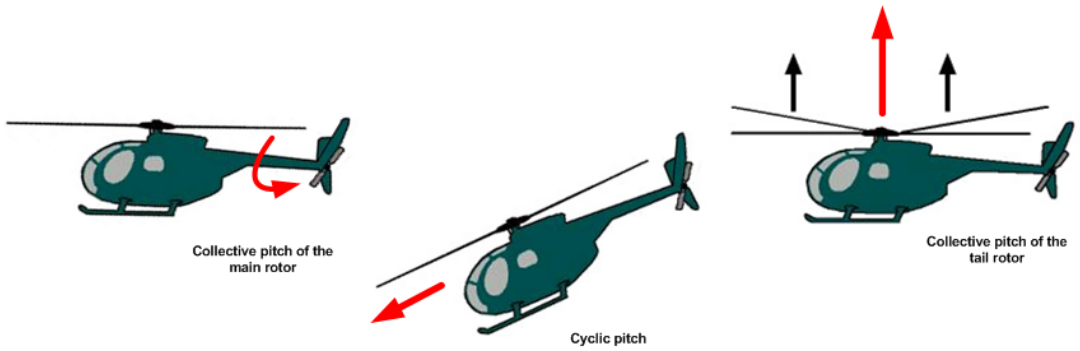


Figure C.1: Helicopter's control commands. The left command (Collective pitch of the main rotor) changes the pitch angle of the main rotor blades collectively, increasing or decreasing the helicopter's altitude. The center control (Cyclic pitch) tilts the main rotor disk (swashplate) to a specific direction, changing the angle of attack of the rotor blades cyclically to obtain forward, backward, and sideward movement directions of the helicopter. The right one (Collective pitch of the tail rotor) changes the angle of the tail rotor blades, keeping the helicopter pointing in one direction.

position. Therefore, if a collective input is made, all the blades change equally, and as a result the helicopter increases or decreases in altitude.

A helicopter's flight control is usually based on PID controllers, arranged in a cascade formation so that each controller generates references to the next one, being the attitude control the one which generates the commands to the helicopter's actuators.

The controllers read the information from the state estimator and stabilize the helicopter in a specific position. The attitude control reads roll, pitch, and yaw values and generates the adequate commands to stabilize the helicopter's attitude. The velocity control reads the velocity values from the state estimator and generates references of roll, pitch, and collective of the main rotor in order to achieve lateral and longitudinal displacements. The position controller is implemented with PID controllers. It is at the highest level of the system and is designed to receive GPS coordinates or visual-based references to generate velocity references to the velocity controller and to the collective of the main rotor. A complete description of the control system can be found in [128].

C.1.1. Rotomotion SR 20

The Colibri III system shown in Figure C.2 is an electric helicopter. It is a modified Xcell Electric RC helicopter SR20. The principal characteristics of this vehicle are summarized in Table **??**. External processes (ground clients and on-board system) interact with the low level autopilot through a UDP socket over an ethernet network. The API (application programming interface) consists in a series of messages that are used to send and receive information from the autopilot. Some of these messages include state information, such as GPS state (quality of the position solution and the raw Latitude/Longitude coordinates), position control commands, velocity teleoperation commands, helicopter desired position and heading confirmation (indicating the position in the NED frame to which the UAV will try to fly), or pan and tilt platform control, among others.

Figure C.2: Autonomous helicopter. The Rotomotion SR20 modified Xcell Electric RC helicopter, used in this thesis.

Specifications	
Length (mm)	1700
Height (mm)	650
Main Blade Size (mm)	880
Tail Blade Size (mm)	130
Power Unit	Electric motor @1400 W.
Max Payload (kg)	4
Endurance (min)	15
Autopilot	AFCS 3.4
Sensor	GPS, IMU and Magnetometer
Communications	WiFi/Ethernet
Classification	Light UAV
Visual System	
CPU	VIA nano-itx 1.5 GHz
RAM (GB)	2.0
camera interfaces	USB 2.0, 1394a, ethernet, Analog captured
OS	Linux
Communication with Autopilot and Ground Station	WiFi/Ethernet

Table C.1: Technical Specifications of the Rotomotion SR20 Electrical UAV

C.2. Autonomous Quadrotors

A quadrotor, also called a quadrotor helicopter or quadcopter, is a vertical take off and landing system (VTOL) lifted and propelled by four rotors. Unlike most helicopters, quadrotors generally use symmetrically pitched blades. Control of vehicle motion is achieved by altering the pitch and/or rotation rates of one or more rotor discs, thereby changing its torque load and thrust/lift characteristics.

There are several advantages to quadrotors over comparably-scaled helicopters. Firstly, quadrotors do not require mechanical linkages to vary the rotor blade pitch angle as they spin. This simplifies the design and maintenance of the vehicle. Secondly, the use of four rotors allows each individual rotor to have a smaller diameter than the equivalent helicopter rotor, allowing them to possess less kinetic energy during the flight. This reduces the damages caused if rotors hit anything. The required abilities of pilots of quadrotors are significantly lower than those of pilots of helicopters.

C.2.1. Basic Quadrotor Mechanics

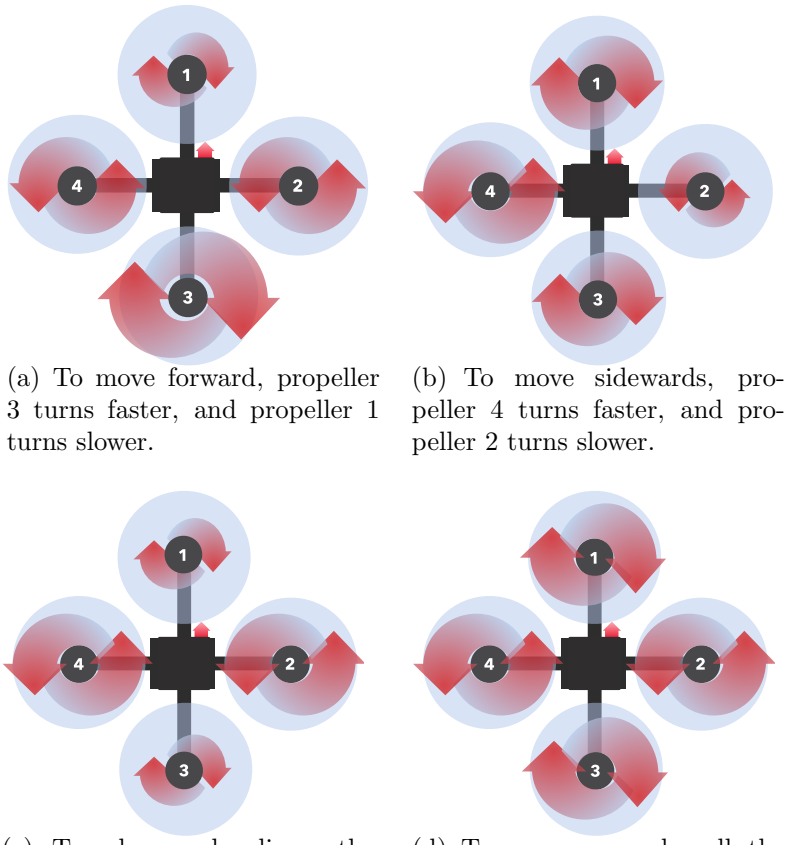
A quadrotor is controlled by the angular speeds of four electric motors, as shown in C.3. Each motor produces a thrust and a torque, whose combination generates the main thrust, the yaw torque, the pitch torque, and the roll torque acting on the quadrotor. Two opposite rotors are rotating clockwise, while the other two rotors are rotating counter-clockwise, canceling out their respective torques. The following actions can be taken to maneuver the quadrotor:

- Hovering: ideally for the vehicle to hover, all propellers rotate at the same speed. When doing so, all the propeller forces are balanced out, making the quadrotor hang steady in the air.

- Horizontal motion: to be able to fly in one direction, the quadrotor has to be brought out of balance. The speed of the propeller that opposes the desired direction is increased. This makes the quadrotor tip over in a certain direction. See examples in Figures C.3(a) and C.3(b); e.g. to fly forward, the back-propeller has to turn faster and the front-propeller slower. This action enables the quadrotor to move forward and backward. These actions are always balancing the rotation speed of the propellers, which permits to obtain the same global upward thrust. The consequence is that the quadrotor will ideally stay at the same altitude while maneuvering in a horizontal plane.

- Heading: in order to change the heading of the vehicle, the equilibrium of the rotation speeds between the clockwise and counter-clockwise speeds has to be broken. For instance, to rotate the vehicle clockwise, the forward/backward propellers will turn faster and the left/right propellers will slow down a little. This will make the quadrocopter turn clockwise, while maintaining the same height. See Figure C.3(c).

- Vertical motion: to change the altitude of the vehicle, all the propeller speeds are increased or decreased simultaneously, see Figure C.3(a).



(a) To move forward, propeller 3 turns faster, and propeller 1 turns slower.

(b) To move sidewards, propeller 4 turns faster, and propeller 2 turns slower.

(c) To change heading, the left/right propellers turn faster and the forward/backward propellers slow down.

(d) To move upwards, all the propeller speeds are increased.

Figure C.3: Quadrotor flying principle. Image from [153]

On the other hand, accurate sensors and advanced controlling routines are required to control the speed engines of each rotor and to control the quadrotor's movements and accelerations.

C.2.2. Ascending Technologies Pelican Quadrotor

The Pelican is a quadrotor developed by Ascending Technologies [9]. It is built in a modular fashion, allowing to change boards quickly and easily. The main core is designed like a tower, making it plug-and-play. This testbed, shown in Figure C.4, has a low-level stability controller based on PID that fuses information from the GPS, the IMU, the pressure altimeter and the magnetometer using a Kalman filter. This controller is embedded, closed and unmodifiable, but its gains are tunable. Onboard vision processing is achieved using a dual core Atom 1.6 GHz processor with 1 GB RAM; has a wireless interface, and supports several types of USB cameras (mono or stereo). This computer runs Linux OS working in a multi-client wireless 802.11(a,b,g) ad-hoc network, allowing it to communicate with a ground station PC used to monitor and supervise the mision.

It has a maximum payload of 650 g.

Figure C.4: The Pelican. Developed by Ascending Technologies [9]

Table C.2 summarizes the principal technical characteristics of the Pelican.

	AscTec Pelican
Rotors	4
Area (mm^2)	600
Blade Size (mm)	250
Power Unit	4 Electric motors @120 W.
Max Payload (kg)	0.5
Endurance (min)	15
Autopilot	AscTec AutoPilot
Sensor	GPS, IMU, Magnetometer, and pressure altimeter
Communications	Serial Xbee RF
Classification	Light UAV
Visual System	
CPU	AscTec 1.6 GHz Atom Board
RAM (GB)	1.0
Camera interfaces	USB 2.0
OS	Linux
Communication with Autopilot and Ground Station	Serial Interface /Wifi-Ethernet

Table C.2: Technical specifications of the Pelican AscTec UAV

External processes (ground clients and on-board system) interact with the Low Level AscTec autopilot through a serial interface (running at 57600bps) and a series of messages, that are used to send and get information to/from the autopilot.

C.2.3. AR.Drone Parrot Quadrotor

The AR.Drone is a low cost quadrotor developed by Parrot [7]. Figure C.5 shows the AR.Drone Parrot. Original designed as a sophisticated toy for augmented reality games, it is now a very popular quadrotor because of the available apps for controlling it using a smart phone or a tablet PC. It has two different styrofoam hulls for indoor and outdoors flights. Depending of the mounted hull, the quadrotor has a payload of 80 to 120 g. The drone is equipped with two cameras (forwards-looking and downwards-looking), an ultrasound altimeter, a 3-axis accelerometer, a 2-axis gyroscope (for roll and pitch), and a 1-axis yaw precision gyroscope. The on-board controller is composed of an ARM9 468 MHz processor with 128 Mb DDR Ram, on which a BusyBox-based GNU/Linux distribution is running. It has a USB service port and is controlled via wireless LAN.

Figure C.5: AR.Drone Parrot

Bibliography

[1] Combining stereo vision and inertial navigation system for a quad-rotor uav. Journal of Intelligent and Robotic Systems **65**(1-4) (2012). DOI 10.1007/s10846-011-9571-7

[2] Allen, J.G., Xu, R.Y.D., Jin, J.S.: Object tracking using camshift algorithm and multiple quantized feature spaces. In: Proceedings of the Pan-Sydney area workshop on Visual information processing, VIP '05, pp. 3–7. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2004)

[3] Altuğ, E., Ostrowski, J.P., Taylor, C.J.: Control of a quadrotor helicopter using dual camera visual feedback. International Journal of Robotics Research **24**(5), 329–341 (2005)

[4] Amidi, O., Kanade, T., Fujita, K.: A visual odometer for autonomous helicopter flight. Journal of Robotics and Autonomous Systems **28**, 185 – 193 (1999)

[5] Andersen, E., Taylor, C.: Improving MAV pose estimation using visual information. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)., pp. 3745 –3750 (2007)

[6] Anderson, C.H., Bergen, J.R., Burt, P.J., Ogden, J.M.: Pyramid methods in image processing. RCA Engineer, vol. 29, no. 6, pp. 33-41, Nov./Dec. (1984)

[7] ArDrone: Parrot. `http://ardrone.parrot.com` (2012)

[8] Artieda, J., Sebastian, J., Campoy, P., Correa, J., Mondragón, I., Martínez, C., Olivares, M.: Visual 3-D SLAM from UAVs. Journal of Intelligent & Robotic Systems **55**, 299–321 (2009). 10.1007/s10846-008-9304-8

[9] Ascending: Technologies. `http://www.asctec.de` (2010)

[10] AutonomyLab: A ROS Driver for ARDrone 1.0 and 2.0. `https://github.com/AutonomyLab/ardrone_autonomy` (2013)

[11] Babenko, B., Belongie, M.H.Y.S.: Robust object tracking with online multiple instance learning (2011)

[12] Bachrach, A., Prentice, S., He, R., Roy, N.: Range - robust autonomous navigation in gps-denied environments. Journal of Field Robotics **28**(5), 644–666 (2011)

[13] Baker, S., Datta, A., Kanade", T.: "parameterizing homographies". Tech. Rep. "CMU-RI-TR-06-11", "Robotics Institute", "Pittsburgh, PA" ("2006")

[14] Baker, S., Matthews, I.: Equivalence and efficiency of image alignment algorithms. In: Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 1090 – 1097 (2001)

[15] Baker, S., Matthews, I.: Lucas-kanade 20 years on: A unifying framework. International Journal of Computer Vision **56**(1), 221 – 255 (2004)

[16] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. **110**(3), 346–359s (2008)

[17] Benhimane, S., Malis, E.: Real-time image-based tracking of planes using efficient second-order minimization. In: Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 1, pp. 943 – 948 vol.1 (2004)

[18] Bergen, J.R., Anandan, P., Hanna, K.J., Hingorani, R.: Hierarchical model-based motion estimation. In: ECCV '92: Proceedings of the Second European Conference on Computer Vision, pp. 237–252. Springer-Verlag, London, UK (1992)

[19] Beyeler, A., Zufferey, J.C., Floreano, D.: Optipilot: Control of take-off and landing using optic flow. In Proceedings of the European Micro Air Vehicle Conference and Competition 2009, Delft, Netherlands, (EMAV 2009) (2006)

[20] du Bois, J.L., , Thomas, P., Bullock, S., Bhandari, U., Richardson", T.: Control methodologies for relative motion reproduction in a robotic hybrid test simulation of aerial refuelling. In: AIAA Guidance, Navigation, and Control Conference. Minneapolis, Minnesota (2012)

[21] Bonin-Font, F., Ortiz, A., Oliver, G.: Visual navigation for mobile robots: A survey. J. Intell. Robotics Syst. **53**(3), 263–296 (2008)

[22] Bouguet, J.Y.: Pyramidal implementation of the Lucas Kanade feature tracker: description of the algorithm. Technical report, OpenCV Document, Intel Microprocessor Research Labs (2002)

[23] Bouguet, J.Y.: Camera calibration toolbox for Matlab (2008). URL `http://www.vision.caltech.edu/bouguetj/calib_doc/.`

[24] Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly (2008)

[25] Bradski, G.R.: Computer vision face tracking for use in a perceptual user interface. Intel Technology Journal (Q2) (1998)

[26] Brooks, R., Arbel, T.: Generalizing inverse compositional and esm image alignment. International Journal of Computer Vision **87**, 191–212 (2010)

[27] Brown, L.G.: A survey of image registration techniques. ACM Comput. Surv. **24**(4), 325–376 (1992)

[28] Bry, A., Bachrach, A., Roy, N.: State estimation for aggressive flight in gps-denied environments using onboard sensing. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2012). St Paul, MN (2012)

[29] Bryson, M., Sukkarieh, S.: Building a robust implementation of bearing-only inertial slam for a uav. J. Field Robotics **24**(1-2), 113–143 (2007)

[30] Buenaposada, J., Baumela, L.: Real-time tracking and estimation of plane pose. In: Proceedings 16th International Conference on Pattern Recognition, 2002, vol. 2, pp. 697 – 700 vol.2 (2002)

[31] Buenaposada, J., Baumela, L.: Speeding up ssd planar tracking by pixel selection. In: Image Processing. 2002. Proceedings. 2002 International Conference on, vol. 1, pp. I–565–I–568 vol.1 (2002)

[32] Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. Communications, IEEE Transactions on **31**(4), 532 – 540 (1983)

[33] Caballero, F., Merino, L., Ferruz, J., Ollero, A.: A visual odometer without 3D reconstruction for aerial vehicles. Applications to building inspection. In: IEEE International Conference on Robotics and Automation (ICRA 2005). (2005)

[34] Caballero, F., Merino, L., Ferruz, J., Ollero, A.: Vision-Based Odometry and SLAM for Medium and High Altitude Flying UAVs. Journal of Intelligent and Robotic Systems **54**, 137–161 (2009). 10.1007/s10846-008-9257-y

[35] Campos, M.F.M., de Souza Coelho, L.: Autonomous dirigible navigation using visual tracking and pose estimation. In: ICRA, pp. 2584–2589 (1999)

[36] Campoy, P., Correa, J., Mondragón, I., Martínez, C., Olivares, M., Mejias, L., Artieda, J.: Computer Vision Onboard UAVs for Civilian Tasks. Journal of Intelligent and Robotic Systems **54**, 105–135 (2009)

[37] Campoy, P., Mondragón, I.F., Olivarez-Méndez, M.A., Martínez, C.: Visual Servoing for UAVs, rong-fong fung (ed.) edn. INTECH (2010). URL `http://sciyo.com/articles/show/title/visual-servoing-for-uav`

[38] Can, A., Stewart, C., Roysam, B., Tanenbaum, H.: A feature-based, robust, hierarchical algorithm for registering pairs of images of the curved human retina. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**(3), 347 –364 (2002)

[39] Cao, X., Lan, J., Yan, P., Li, X.: Vehicle detection and tracking in airborne videos by multi-motion layer analysis. Machine Vision and Applications **23**, 921–935 (2012)

[40] Capel, D.: Image Mosaicing and Super-Resolution (Cphc/Bcs Distinguished Dissertations.). SpringerVerlag (2004)

[41] Cesetti, A., Frontoni, E., Mancini, A., Ascani, A., Zingaretti, P., Longhi, S.: A visual global positioning system for unmanned aerial vehicles used in photogrammetric applications. J. Intell. Robotics Syst. **61**(1-4), 157–168 (2011)

[42] Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P., Longhi, S.: A vision-based guidance system for uav navigation and safe landing using natural landmarks. Journal of Intelligent and Robotic Systems **57**(1-4), 233–257 (2010). DOI 10.1007/s10846-009-9373-3

[43] Chaumette, F., Hutchinson, S.: Visual servo control, part i: Basic approaches. IEEE Robotics and Automation Magazine **13**(4), 82–90 (2006)

[44] Chaumette, F., Hutchinson, S.: Visual servo control. ii. advanced approaches [tutorial]. Robotics Automation Magazine, IEEE **14**(1), 109–118 (March)

[45] Cheng, Y., Maimone, M.W., Matthies, L.: Visual odometry on the mars exploration rovers. In: SMC, pp. 903–910 (2005)

[46] Cobzas, D., Sturm, P.: 3d ssd tracking with estimated 3d planes. In: Computer and Robot Vision, 2005. Proceedings. The 2nd Canadian Conference on, pp. 129–134 (2005). DOI 10.1109/CRV.2005.4

[47] Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. Pattern Analysis and Machine Intelligence, IEEE Transactions on **25**(5), 564–577 (2003). DOI http://dx.doi.org/10.1109/TPAMI.2003.1195991

[48] Conte, G., Doherty, P.: Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information. EURASIP J. Adv. Signal Process **2009**, 10:1–10:18 (2009)

[49] Corke, P., Detweiler, C., Dunbabin, M., Hamilton, M., Rus, D., Vasilescu, I.: Experiments with underwater robot localization and tracking. In: Robotics and Automation, 2007 IEEE International Conference on, pp. 4556–4561 (2007). DOI 10.1109/ROBOT.2007.364181

[50] Corral, E.M.: Efficient model-based 3d tracking by using direct image registration. Ph.D. thesis, Facultad de Informática. Universidad Politécnica de Madrid, Spain (2012)

[51] Criminisi, A., Reid, I., Zisserman, A.: A plane measuring device. In: Proceedings of the 8th British Machine Vision Conference, Colchester. UK (1997)

[52] Computer Vision Group. Universidad Politécnica de Madrid. CVG 2013. http://www.vision4uav.com/

[53] Dame, A., Marchand, E.: Accurate real-time tracking using mutual information. In: IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'10, pp. 47–56. Seoul, Korea (2010)

[54] Dellaert, F., Collins, R.: Fast image-based tracking by selective pixel integration. In: ICCV 99 Workshop on Frame-Rate Vision (1999)

[55] Dieter, S.: Helicopteros de Radio Control. Cúpula (1994)

[56] Doebbler, J., Spaeth, T., Valasek, J., Monda, M.J., Schaub, H.: Boom and Receptacle Autonomous Air Refueling Using Visual Snake Optical Sensor. Journal of Guidance Control and Dynamics **30**, 1753–1769 (2007)

[57] Drummond, T., Cipolla, R.: Real-time visual tracking of complex structures. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**(7), 932–946 (2002). DOI 10.1109/TPAMI.2002.1017620

[58] Dufaux, F., Konrad, J.: Efficient, robust, and fast global motion estimation for video coding. Image Processing, IEEE Transactions on **9**(3), 497 –501 (2000)

[59] Dupac, J., Matas, J., Naiser, F.: Ultra-fast tracking based on zero-shift points. Image Vision Comput. **30**(12), 1016–1031 (2012)

[60] Dusha, D., Boles, W., Walker, R.: Attitude estimation for a fixed-wing aircraft using horizon detection and optical flow. In: Proceedings of the 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications, DICTA '07, pp. 485–492. IEEE Computer Society, Washington, DC, USA (2007)

[61] Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., Twombly, X.: Vision-based hand pose estimation: A review. Computer Vision and Image Understanding **108**, 52 – 73 (2007). ¡ce:title¿Special Issue on Vision for Human-Computer Interaction¡/ce:title¿

[62] Escalera, A.D.L., Moreno, L.E., Salichs, M.A., Armingol, J.M.: Road traffic sign detection and classification. IEEE Transactions on Industrial Electronics **44**, 848–859 (1997)

[63] Fischer, B., Modersitzki, J.: Ill-posed medicineâ an introduction to image registration. Inverse Problems **24**(3), 034,008 (2008)

[64] Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)

[65] Fleyeh, H.: Traffic signs color detection and segmentation in poor light conditions. In: Proceedings of the IAPR Conference on Machine Vision Applications MVA (2005)

[66] Ford, A., Roberts, A.: Colour space conversions. Available at http://www.poynton.com/PDFs/coloureq.pdf (1998)

[67] Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. Information Theory, IEEE Transactions on **21**(1), 32–40 (1975)

[68] Fukushima: Disaster. http://www.engadget.com/2011/04/21/t-hawk-uav-enters-fukushima-danger-zone-returns-with-video/

[69] García Carrillo, L., Rondon, E., Sanchez, A., Dzul, A., Lozano, R.: Stabilization and trajectory tracking of a quad-rotor using vision. Journal of Intelligent and Robotic Systems **61**, 103–118 (2011)

[70] Garcia-pardo, P.J., Sukhatme, G.S., Montgomery, J.F.: Towards vision-based safe landing for an autonomous helicopter (2001)

[71] Gazebo: ROS Gazebo Simulator. `http://www.ros.org/wiki/simulator_gazebo` (2012)

[72] Gomez-Balderas, J., Flores, G., García Carrillo, L., Lozano, R.: Tracking a ground moving target with a quadrotor using switching control. Journal of Intelligent and Robotic Systems **70**(1-4), 65–78 (2013)

[73] Gomez-Balderas, J.E., Castillo, P., Guerrero, J., Lozano, R.: Vision based tracking for a quadrotor using vanishing points. Journal of Intelligent and Robotic Systems **65**(1-4), 361–371 (2012). DOI 10.1007/s10846-011-9579-z. URL `http://dx.doi.org/10.1007/s10846-011-9579-z`

[74] Greenpeace: Greenpeace used drones to track japanese whalers on the high seas. http://en.wikinoticia.com/Technology/general-technology/104601-greenpeace-used-drones-to-track-japanese-wh (2011)

[75] Greer, D.G., Mudford, R., Dusha, D., Walker, R.: Airbone systems laboratory for automation research. In: 27^{TH} International Congress of the Aeronautical Sciences, ICAS 2010. Nice, France

[76] Guenard, N., Hamel, T., Mahony, R.: A practical visual servo control for an unmanned aerial vehicle. Robotics, IEEE Transactions on **24**(2), 331–340 (2008). DOI 10.1109/TRO.2008.916666

[77] Gurtner, A., Greer, D., Glassock, R., Mejias, L., Walker, R., Boles, W.: Investigation of fish-eye lenses for small-uav aerial photography. Geoscience and Remote Sensing, IEEE Transactions on **47**(3), 709–721 (2009). DOI 10.1109/TGRS.2008.2009763

[78] Hager, G., Belhumeur, P.: Efficient region tracking with parametric models of geometry and illumination. Pattern Analysis and Machine Intelligence, IEEE Transactions on **20**(10), 1025–1039 (1998)

[79] Hanna, K., Okamoto, N.: Combining stereo and motion analysis for direct estimation of scene structure. In: Computer Vision, 1993. Proceedings., Fourth International Conference on, pp. 357 –365 (1993). DOI 10.1109/ICCV.1993.378192

[80] Harris, C., Stephens, M.: A Combined Corner and Edge Detection, vol. 15, pp. 147–151 (1988)

[81] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, New York, NY, USA (2003)

[82] Hayashi, Y., Fujiyoshi, H.: Robocup 2007: Robot soccer world cup xi. chap. Mean-Shift-Based Color Tracking in Illuminance Change, pp. 302–311. Springer-Verlag, Berlin, Heidelberg (2008)

[83] Heintz, F., Rudol, P., Doherty, P.: From images to traffic behavior - a uav tracking and monitoring application. In: Information Fusion, 2007 10th International Conference on, pp. 1–8 (2007). DOI 10.1109/ICIF.2007.4408103

[84] Hellier, P., Barillot, C.: Coupling dense and landmark-based approaches for nonrigid registration. Medical Imaging, IEEE Transactions on **22**(2), 217 –227 (2003)

[85] Hess, R.: SIFT feature detector implementation in C. http://www.web.engr.oregonstate.edu/~hess/index.html (2007)

[86] Hess, R.: An open-source SIFT library. In: Proceedings of the international Conference on Multimedia, pp. 1493–1496 (2010). URL `http://portal.acm.org/citation.cfm?id=1874256`

[87] Holzer, S., Ilic, S., Navab, N.: Multilayer adaptive linear predictors for real-time tracking. Pattern Analysis and Machine Intelligence, IEEE Transactions on **35**(1), 105 –117 (2013)

[88] How, J., Bethke, B., Frank, A., Dale, D., Vian, J.: Real-time indoor autonomous vehicle test environment. Control Systems, IEEE **28**(2), 51–64 (2008). DOI 10.1109/MCS.2007.914691

[89] Hrabar, S., Sukhatme, G.: Omnidirectional vision for an autonomous helicopter. In: Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on, vol. 1, pp. 558–563 vol.1 (2003). DOI 10.1109/ROBOT.2003.1241653

[90] Hrabar, S., Sukhatme, G., Corke, P., Usher, K., Roberts, J.: Combined optic-flow and stereo-based navigation of urban canyons for a uav. Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on pp. 3309–3316 (2005). DOI 10.1109/IROS.2005.1544998

[91] Huang, A.S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., Roy, N.: Visual odometry and mapping for autonomous flight using an rgb-d camera. In: Int. Symposium on Robotics Research (ISRR). Flagstaff, Arizona, USA (2011)

[92] Hutchinson, S., Hager, G., Corke, P.: A tutorial on visual servo control. Robotics and Automation, IEEE Transactions on **12**(5), 651 –670 (1996). DOI 10.1109/70.538972

[93] Hwangbo, M., Kim, J.S., Kanade, T.: Inertial-aided klt feature tracking for a moving camera. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pp. 1909–1916 (2009)

[94] Irani, M., Anandan, P.: About direct methods. In: B. Triggs, A. Zisserman, R. Szeliski (eds.) Vision Algorithms: Theory and Practice, *Lecture Notes in Computer Science*, vol. 1883, pp. 267–277. Springer Berlin / Heidelberg (2000)

[95] Irani, M., Peleg, S.: Motion analysis for image enhancement: Resolution, occlusion, and transparency. Journal of Visual Communication and Image Representation **4**, 324–335 (1993)

[96] Irani, M., Rousso, B., Peleg, S.: Computing occluding and transparent motions. Int. J. Comput. Vision **12**(1), 5–16 (1994). DOI http://dx.doi.org/10.1007/BF01420982

[97] Ishikawa, T., Matthews, I., Baker, S.: Efficient image alignment with outlier rejection. Tech. Rep. CMU-RI-TR-02-27, Robotics Institute, Pittsburgh, PA (2002)

[98] Johnson, A., Montgomery, J., Matthies, L.: Vision guided landing of an autonomous helicopter in hazardous terrain. Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on pp. 3966–3971 (2005)

[99] Jurie, F., Dhome, M.: Real time robust template matching. In: P.L. Rosin, A.D. Marshall (eds.) British Machine Vision Conference, BMVC 2002, September, 2002, pp. 123–132. British Machine Vision Association, Cardiff, Royaume-Uni (2002)

[100] Kaiser, K., Gans, N., Dixon, W.: Position and Orientation of an Aerial Vehicle through Chained, Vision-Based Pose Reconstruction. Proc. AIAA Conf. on Guidance, Navigation and Control (2005)

[101] Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on **34**(7), 1409–1422 (2012). DOI 10.1109/TPAMI.2011.239

[102] Kelly, J., Saripalli, S., Sukhatme, G.S.: Combined Visual and Inertial Navigation for an Unmanned Aerial Vehicle. In: Proc. 6th Int'l Conf. Field and Service Robotics (FSR'07). Chamonix, France (2007)

[103] Kendoul, F.: Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. Journal of Field Robotics **29**(2), 315–378 (2012)

[104] Kong, W., Zhang, D., Wang, X., Xian, Z., Zhang, J.: Autonomous landing of an uav with a ground-based actuated infrared stereo vision system. http://tams.informatik.uni-hamburg.de/people/kong/publications/130322.IROS2013.pdf (2013)

[105] Kruijff, G.J., Tretyakov, V., Linder, T., Pirri, F., Gianni, M., Papadakis, P., Pizzoli, M., Sinha, A., Pianese, E., Corrao, S., Priori, F., Febrini, S., Angeletti, S.: Rescue robots at earthquake-hit mirandola, italy: a field report. In: Proceedings of the 10th

IEEE International Symposium on Safety, Security, and Rescue Robotics. IEEE Press (2012)

[106] Kumar, R., Sawhney, H., Samarasekera, S., Hsu, S., Tao, H., Guo, Y., Hanna, K., Pope, A., Wildes, R., Hirvonen, D., Hansen, M., Burt, P.: Aerial video surveillance and exploitation. Proceedings of the IEEE **89**(10), 1518 –1539 (2001). DOI 10.1109/5.959344

[107] Lee, D., Ryan, T., Kim, H.: Autonomous landing of a vtol uav on a moving platform using image-based visual servoing. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 971–976 (2012)

[108] Lepetit, V., Fua, P.: Monocular model-based 3d tracking of rigid objects. Found. Trends. Comput. Graph. Vis. **1**(1), 1–89 (2005)

[109] Li, B., Mu, C., Wu, B.: A survey of vision based autonomous aerial refueling for unmanned aerial vehicles. In: Intelligent Control and Information Processing (ICICIP), 2012 Third International Conference on, pp. 1–6 (2012). DOI 10.1109/ICICIP.2012.6391480

[110] Li, H., Member, S., Manjunath, B.S., Mitra, S.K.: A contour-based approach to multisensor image registration. IEEE Transactions on Image Processing **4**, 320–334 (1995)

[111] Lindsey, Q., Mellinger, D., Kumar, V.: Construction with quadrotor teams. Auton. Robots **33**(3), 323–336 (2012)

[112] Lindsten, F., Callmer, J., Ohlsson, H., Tornqvist, D., Schon, T., Gustafsson, F.: Geo-referencing for uav navigation using environmental classification. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 1420–1425 (2010). DOI 10.1109/ROBOT.2010.5509424

[113] chi Liu, Y., hai Dai, Q.: A survey of computer vision applied in aerial robotic vehicles. In: Optics Photonics and Energy Engineering (OPEE), 2010 International Conference on, vol. 1, pp. 277–280 (2010). DOI 10.1109/OPEE.2010.5508131

[114] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)

[115] Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 674–679 (1981)

[116] Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An Invitation to 3-D Vision: From Images to Geometric Models. SpringerVerlag (2003)

[117] Madison, R., Andrews, G., DeBitetto, P., Rasmussen, S., Bottkol, M.: Vision-aided navigation for small uavs in gps-challenged environments. In: AIAA Infotech@Aerospace 2007 Conference and Exhibit, pp. 1420–1425 (2007). DOI doi:10.2514/6.2007-2986

[118] Malis, E., Benhimane, S.: A unified approach to visual tracking and servoing. Robotics and Autonomous Systems **52**(1), 39 – 52 (2005). Advances in Robot Vision

[119] Malis, E., Chaumette, F., Boudet, S.: 2 1/2 D Visual Servoing. IEEE Trans. on Robotics and Automation **15**(2), 238–250 (1999)

[120] Malis, E., Vargas, M.: Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA (2007)

[121] Martínez, C., Campoy, P., Mondragón, I., Olivares-Méndez, M.A.: Trinocular ground system to control UAVs. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). St. Louis, MO, USA (2009)

[122] Martinez, C., Mejias, L., Campoy, P.: A multi-resolution image alignment technique based on direct methods for pose estimation of aerial vehicles. In: Proceedings of the International Conference on Digital Image Computing Techniques and Applications (DICTA), 2011, pp. 542 –548 (2011)

[123] Martínez, C., Mondragón, I., Campoy, P., Sánchez-López, J., Olivares-Méndez, M.: A hierarchical tracking strategy for vision-based applications on-board uavs. Journal of Intelligent & Robotic Systems pp. 1–23 (2013). DOI 10.1007/s10846-013-9814-x

[124] Martínez, C., Mondragón, I.F., Olivares-Méndez, M.A., Campoy, P.: On-board and ground visual pose estimation techniques for uav control. Journal of Intelligent Robotics and Systems **61**(1-4), 301–320 (2011)

[125] Martínez, C., Richardson, T., Campoy, P.: Towards autonomous air-to-air refuelling for uavs using visual information. Robotics and Automation, 2013. ICRA 2013. Proceedings of the 2013 IEEE International Conference on (2013)

[126] Martínez, C., Richardson, T., Thomas, P., du Bois, J.L., Campoy, P.: A vision-based strategy for autonomous aerial refueling tasks. Robotics and Autonomous Systems (0), – (2013)

[127] McGee, T., Sengupta, R., Hedrick, K.: Obstacle detection for small autonomous aircraft using sky segmentation. Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on pp. 4679–4684 (2005)

[128] Mejias, L.: Control visual de un vehiculo aéreo autonomo usando detección y seguimiento de características en espacios exteriores. Ph.D. thesis, Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid, Spain (2006)

[129] Mejias, L., Campoy, I.: Omnidirectional bearing-only see-and-avoid for small aerial robots. In: Automation, Robotics and Applications (ICARA), 2011 5th International Conference on, pp. 23–28 (2011). DOI 10.1109/ICARA.2011.6144850

[130] Mejias, L., Fitzgerald, D.L., Eng, P.C., Xi, L.: Forced landing technologies for unmanned aerial vehicles : towards safer operations. In: L.T. Mung (ed.) Aerial Vehicles, pp. 415–442. In-Tech, Kirchengasse, Austria (2009)

[131] Mejias, L., McNamara, S., Lai, J., Ford, J.J.: Vision-based detection and tracking of aerial targets for UAV collision avoidance. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2010)

[132] Mejias, L., Saripalli, S., Campoy, P., Sukhatme, G.S.: Visual servoing of an autonomous helicopter in urban areas using feature tracking. Journal of Field Robotics **23** (2006)

[133] Mejias, L., Saripalli, S., Sukhatme, G., Campoy, P.: Detection and tracking of external features in a urban environment using an autonomous helicopter. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 3983–3988 (2005)

[134] Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 2520–2525 (2011). DOI 10.1109/ICRA.2011.5980409

[135] Mendez, M.A.O.: Soft-computing based visual control for unmanned vehicles. Ph.D. thesis, Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid, Spain (2013)

[136] Merino, L., Caballero, F., Martínez-de Dios, J., Ferruz, J., Ollero, A.: A cooperative perception system for multiple uavs: Application to automatic detection of forest fires. J. Field Robotics **23**(3-4), 165–184 (2006)

[137] Merz, T., Duranti, S., Conte, G.: Autonomous landing of an unmanned helicopter based on vision and inertial sensing. In: In Proc. of the 9th International Symposium on Experimental Robotics (2004)

[138] Metni, N., Hamel, T.: A {UAV} for bridge inspection: Visual servoing control law with orientation limits. Automation in Construction **17**(1), 3 – 10 (2007)

[139] Michael, N., Mellinger, D., Lindsey, Q., Kumar, V.: The grasp multiple micro-uav testbed. Robotics Automation Magazine, IEEE **17**(3), 56–65 (2010). DOI 10.1109/MRA.2010.937855

[140] Mondragón, I.: Onboard visual control algorithms for unmanned aerial vehicles. Ph.D. thesis, Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid, Spain (2011)

[141] Mondragon, I., Campoy, P., Correa, J., Mejias, L.: Visual model feature tracking for uav control. In: Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on, pp. 1 –6 (2007). DOI 10.1109/WISP.2007.4447629

[142] Mondragón, I.F., Campoy, P., Martinez, C., Olivares-Mendez, M.: 3D pose estimation based on planar object tracking for UAVs control. In: Proceedings of IEEE International Conference on Robotics and Automation 2010 ICRA2010. Anchorage, AK, USA (2010)

[143] Moss, S., Hancock, E.R.: Multiple line-template matching with the em algorithm. Pattern Recogn. Lett. **18**(11-13), 1283–1292 (1997)

[144] Neild, B.: CNN. not just for military use, drones turn civilian. http://edition.cnn.com/2012/07/12/world/europe/civilian-drones-farnborough (2013)

[145] Nguwi, Y., Kouzani, A.: A study on automatic recognition of road signs. In: Cybernetics and Intelligent Systems, 2006 IEEE Conference on, pp. 1 –6 (2006)

[146] Nister, D., Naroditsky, O., Bergen, J.: Visual odometry for ground vehicle applications. Journal of Field Robotics **23**, 2006 (2006)

[147] Nitrofirex: Forest fire fighting during night. http://www.nitrofirex.com/en (2013)

[148] Olivares-Mendez, M.A., Mejias, L., Campoy, P., Mellado-Bataller, I.: Cross-entropy optimization for scaling factors of a fuzzy controller: A see-and-avoid approach for unmanned aerial systems. J. Intell. Robotics Syst. **69**(1-4), 189–205 (2013)

[149] Ollero, A., Merino, L.: Control and perception techniques for aerial robotics. Annual Reviews in Control **28**(2), 167 – 178 (2004)

[150] Paragios, N., Chen, Y., Faugeras, O.: Handbook of Mathematical Models in Computer Vision. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2005)

[151] PEBRIANTI, D., KENDOUL, F., AZRAD, S., WANG, W., NONAMI, K.: Autonomous hovering and landing of a quad-rotor micro aerial vehicle by means of on ground stereo vision system. Journal of System Design and Dynamics **4**(2), 269–284 (2010)

[152] Pollini, L., Campa, G., Giulietti, F., Innocenti, M.: Virtual simulation set-up for uavs aerial refuelling. In: AIAA Guidance Navigation and Control Conference and Exhibit, August, pp. 1–8 (2003)

[153] Puerta, J.P.: On-board control algorithms for autonomous indoors navigation of multirotor micro air vehicles. MS Thesis, Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid, Spain (2012)

[154] Puri, A., Valavanis, K., Kontitsis, M.: Statistical profile generation for traffic monitoring using real-time uav based video data. In: Control Automation, 2007. MED '07. Mediterranean Conference on, pp. 1–6 (2007). DOI 10.1109/MED.2007.4433658

[155] Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)

[156] Raja, Y., McKenna, S.J., Gong, S.: Colour model selection and adaption in dynamic scenes. In: Proceedings of the 5th European Conference on Computer Vision-Volume I - Volume I, ECCV '98, pp. 460–474. Springer-Verlag, London, UK, UK (1998)

[157] Rao, C., Guo, Y., Sawhney, H., Kumar, R.: A heterogeneous feature-based image alignment method. In: Pattern Recognition, 2006. ICPR 2006. 18th International Conference on (2006)

[158] Ready, B., Taylor, C.: Inertially Aided Visual Odometry for Miniature Air Vehicles in GPS-denied Environments. Journal of Intelligent and; Robotic Systems **55**, 203–221 (2009)

[159] Munoz Salinas, R.: Aruco: a minimal library for augmented reality applications based on opencv. http://www.uco.es/investiga/grupos/ava/node/26

[160] Saripalli, S., Montgomery, J., Sukhatme, G.: Visually guided landing of an unmanned aerial vehicle. Robotics and Automation, IEEE Transactions on **19**(3), 371–380 (2003)

[161] Saripalli, S., Montgomery, J.F., Sukhatme, G.S.: Vision-based autonomous landing of an unmanned aerial vehicle. In: IEEE International Conference on Robotics and Automation (ICRA, pp. 2799–2804 (2002)

[162] Saripalli, S., Montgomery, J.F., Sukhatme, G.S.: Vision-based autonomous landing of an unmanned aerial vehicle. In: IEEE International Conference on Robotics and Automation (ICRA, pp. 2799–2804 (2002)

[163] Sattar, J., Dudek, G.: On the performance of color tracking algorithms for underwater robots under varying lighting and visibility. In: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pp. 3550 –3555 (2006)

[164] Sawhney, H., Kumar, R.: True multi-image alignment and its application to mosaicing and lens distortion correction. Pattern Analysis and Machine Intelligence, IEEE Transactions on **21**(3), 235 –243 (1999). DOI 10.1109/34.754589

[165] Sawhney, H.S., Hsu, S., Kumar, R.: Robust video mosaicing through topology inference and local to global alignment. In: In Proc. European Conference on Computer Vision, pp. 103–119 (1998)

[166] Scaramuzza, D., Fraundorfer, F.: Visual odometry [tutorial]. Robotics Automation Magazine, IEEE **18**(4), 80–92 (2011). DOI 10.1109/MRA.2011.943233

[167] Shakernia, O., Vidal, R., Sharp, C., Ma, Y., Sastry, S.: Multiple view motion estimation and control for landing an unmanned aerial vehicle. Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on **3**, 2793–2798 (2002)

[168] Sharp, C.S., Shakernia, O., Sastry, S.S.: A vision system for landing an unmanned aerial vehicle. In: IEEE International Conference on Robotics and Automation (ICRA), Seoul, Korea, pp. 1720–1727 (2001)

[169] Sheikh, Y., Khan, S., Shah, M.: Feature-based georegistration of aerial images. In: International Conference on Geosensor Networks (2004)

[170] Shi, J., Tomasi, C.: Good features to track. In: 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), pp. 593 – 600 (1994)

[171] Shum, H.Y., Szeliski, R.: Panoramic vision. chap. Construction of panoramic image mosaics with global and local alignment, pp. 227–268. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2001)

[172] Simon, G., Berger, M.O.: Pose estimation for planar structures. IEEE Comput. Graph. Appl. **22**(6), 46–53 (2002)

[173] Mixed Reality Lab Singapore: MXR Toolkit. http://www.mxrtoolkit.sourceforge.net/

[174] Smith, A.R.: Color gamut transform pairs. In: Proceedings of the 5th annual conference on Computer graphics and interactive techniques, SIGGRAPH '78, pp. 12–19. ACM, New York, NY, USA (1978)

[175] Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machine Vision, 2 edn. Chapman & Hall (1998)

[176] Spencer, J.H.: Optical tracking for relative positioning in automated aerial refueling. MS Thesis, Air Force Institute of Technology (2007)

[177] Stanescu, L., Burdescu, D., Stoica, C.: Color image segmentation applied to medical domain. In: H. Yin, P. Tino, E. Corchado, W. Byrne, X. Yao (eds.) Intelligent Data Engineering and Automated Learning - IDEAL 2007, *Lecture Notes in Computer Science*, vol. 4881, pp. 457–466. Springer Berlin / Heidelberg (2007)

[178] Starmac: Starmac-ros package. `http://www.ros.org/wiki/starmac-ros-pkg` (2012)

[179] Stewart, C.V.: Robust parameter estimation in computer vision. SIAM Rev. **41**(3), 513–537 (1999)

[180] Swain, M.J., Ballard, D.H.: Color indexing. Int. J. Comput. Vision **7**(1), 11–32 (1991)

[181] Szeliski, R.: Image alignment and stitching: a tutorial. Found. Trends. Comput. Graph. Vis. **2**(1), 1–104 (2006)

[182] Szeliski, R., Shum, H.Y.: Creating full view panoramic image mosaics and environment maps. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH '97, pp. 251–258. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1997)

[183] Teuliere, C., Eck, L., Marchand, E.: Chasing a moving target from a flying uav. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 4929 –4934 (2011)

[184] Theodore, C., Rowley, D., Hubbard, D., Ansar, A., Matthies, L., Goldberg, S., Whalley, M.: Flight trials of a rotorcraft unmanned aerial vehicle landing autonomously at unprepared sites. In Proceedings of the 62nd Annual Forum of the American Helicopter Society, Phoenix, AZ (2006)

[185] Thurrowgood, S., Soccol, D., Moore, R., Bland, D., Srinivasan, M.: A vision based system for attitude estimation of uavs. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pp. 5725–5730 (2009). DOI 10.1109/IROS.2009.5354041

[186] Torr, P.H.S., Zisserman, A.: Feature based methods for structure and motion estimation. In: Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, ICCV '99, pp. 278–294. Springer-Verlag, London, UK, UK (2000)

[187] Tournier, G.P., Valenti, M., How, J.P., Feron, E.: Estimation and control of a quadrotor vehicle using monocular vision and moir patterns. In: In AIAA Guidance, Navigation and Control Conference, pp. 2006–6711. AIAA (2006)

[188] Tsaig, Y., Averbuch, A.: Automatic segmentation of moving objects in video sequences: a region labeling approach. IEEE Trans. Circuits Syst. Video Techn. **12**(7), 597–612 (2002)

[189] Tseng, D.C., Chang, C.H.: Color segmentation using perceptual attributes. pp. 228–231 (1992)

[190] Turcajova, R., Kautsky, J.: A hierarchical multiresolution technique for image registration. In: Proceedings of SPIE Mathematical Imaging: Wavelet Applications in Signal and Image Processing (1996)

[191] UAV: Unmanned drones to watch over australia's marine mammals. (2010). URL `http://www.underwatertimes.com/news.php?article_id=10346987521`

[192] Umbaugh, S., Moss, R., Stoecker, W., Hance, G.: Automatic color segmentation algorithms-with application to skin tumor feature identification. Engineering in Medicine and Biology Magazine, IEEE **12**(3), 75 –82 (1993)

[193] Valasek, J., Gunnam, K., Hughes, D., Junkins, J., Kimmett, J., Tandale, M.: Vision-based sensor and navigation system for autonomous air refueling. Journal of Guidance Control and Dynamics **28**, 979–989 (2005)

[194] Vendra, S., Campa, G., Napolitano, M., Mammarella, M., Fravolini, M., Perhinschi, M.: Addressing Corner Detection Issues for Machine Vision Based UAV Aerial Refueling. Machine Vision and Applications **18**, 261–273 (2007)

[195] Vicon Motion Systems: VICON MX digital optical motion capture system. http://www.vicon.com (2011)

[196] Videography: http://videography.sourceforge.net/ (2010)

[197] Vo, N., Tran, Q., Dinh, T.B., Dinh, T.B., Nguyen, Q.: An efficient human-computer interaction framework using skin color tracking and gesture recognition. In: Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2010 IEEE RIVF International Conference on, pp. 1 –6 (2010)

[198] Wang, C., Wang, T., Liang, J., Chen, Y., Wu, Y.: Monocular vision and imu based navigation for a small unmanned helicopter. In: Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference on, pp. 1694–1699 (2012). DOI 10.1109/ICIEA.2012.6360998

[199] Wang, W.H., Chen, Y.C.: Image registration by control points pairing using the invariant properties of line segments. Pattern Recognition Letters **18**(3), 269 – 281 (1997)

[200] Weiss, S., Scaramuzza, D., Siegwart, R.: Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments. J. Field Robot. **28**(6), 854–874 (2011). DOI 10.1002/rob.20412. URL `http://dx.doi.org/10.1002/rob.20412`

[201] Wren, C., Azarbayejani, A., Darrell, T., Pentland, A.: Pfinder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence **19**, 780–785 (1997)

[202] Wu, H., Sankaranarayanan, A., Chellappa, R.: In situ evaluation of tracking algorithms using time reversed chains. In: Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, pp. 1–8 (2007)

[203] Wu, Y., Huang, T.: Nonstationary color tracking for vision-based human-computer interaction. Neural Networks, IEEE Transactions on **13**(4), 948 – 960 (2002)

[204] Xu, C., Qiu, L., Liu, M., Kong, B., Ge, Y.: Stereo vision based relative pose and motion estimation for unmanned helicopter landing. Information Acquisition, 2006 IEEE International Conference on pp. 31–36 (2006)

[205] Yao, P., Chen, C., Weng, D.: Markerless tracking algorithm based on 3d model for augmented reality system. In: J. Yang, F. Fang, C. Sun (eds.) Intelligent Science and Intelligent Data Engineering, *Lecture Notes in Computer Science*, vol. 7751, pp. 751–758. Springer Berlin Heidelberg (2013)

[206] Ye, G.: Image Registration and Super-resolution Mosaicing. Ph.D. thesis, The University of New South Wales (2005)

[207] Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Comput. Surv. **38**(4) (2006)

[208] Zhang, H., Yuan, F.: Vehicle tracking based on image alignment in aerial videos. In: EMMCVPR'07: Proceedings of the 6th international conference on Energy minimization methods in computer vision and pattern recognition, pp. 295–302. Springer-Verlag, Berlin, Heidelberg (2007)

[209] Zhang, K., Zhang, L., Yang, M.H.: Real-time compressive tracking. In: Proceedings of the 12th European conference on Computer Vision - Volume Part III, ECCV'12, pp. 864–877. Springer-Verlag, Berlin, Heidelberg (2012)

[210] Zhang, Z.: Flexible camera calibration by viewing a plane from unknown orientations. Computer Vision, IEEE International Conference on **1**, 666 (1999)

[211] Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000)

[212] Zheng, Z., Wang, H., Teoh, E.K.: Analysis of gray level corner detection. Pattern Recognition Letters **20**(2), 149 – 162 (1999)

[213] Zhou, Y., Xue, H., Wan, M.: Inverse image alignment method for image mosaicing and video stabilization in fundus indocyanine green angiography under confocal scanning laser ophthalmoscope. Computerized Medical Imaging and Graphics **27**(6), 513 – 523 (2003)

[214] Zimmermann, K., Matas, J., Svoboda, T.: Tracking by an optimal sequence of linear predictors. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(4), 677–692 (2009)

[215] Ziou, D., Tabbone, S.: Edge Detection Techniques - An Overview. International Journal of Pattern Recognition and Image Analysis **8**, 537–559 (1998)

[216] Zitová, B., Flusser, J.: Image registration methods: a survey. Image and Vision Computing **21**(11), 977–1000 (2003)