Pedro Alexandre de Sousa Prates

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

A Cooperative Approach for Autonomous Landing of UAVs

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: José António Barata de Oliveira, Professor Auxiliar, FCT/UNL

Júri

Presidente: Doutor Paulo da Costa Luís da Fonseca Pinto - FCT/UNL
Arguente: Doutor João Paulo Branquinho Pimentão - FCT/UNL
Vogal: Doutor José António Barata de Oliveira - FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2018

A Cooperative Approach for Autonomous Landing of UAVs

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Acknowledgements

I would like to express gratitude to my dissertation supervisor Prof. José Barata by granting me the outstanding opportunity to contribute to the work of his fantastic research team. To all the team, who kept me company throughout the last year and a half: Eduardo Pinto, Carlos Simões, João Carvalho and in particular Francisco Marques, André Lourenço and Ricardo Mendonça, who were tireless in their support, always ready to give me a hand and discuss ideas throughout the whole development of this dissertation.

My next acknowledgements go to all my colleagues, along these years, in particular Luís Gonçalves who was my workmate throughout most of it, whom I wish the best success in his future career and life.

Thanks to all my friends, Diogo Rodrigues, Pedro Fouto, João Alpalhão, Duarte Grilo, Catarina Leote, Joana Gomes, Diogo Freire and Pedro Mestre who always found a way of making me laugh in the toughest moments, making these last years some of the most enjoyable I have ever had.

Last but definitely not least, I would like to express my wholesome gratitude to my family who always gave me unconditional support, both financial and emotional, through this entire journey. In particular my parents and my sister who made me the man I am today, and without who all I have accomplished in my entire student career would have been impossible.

Abstract

This dissertation presents a cooperative approach for the autonomous landing of MR-VTOL UAVs (Multi Rotor-Vertical Take-off and Landing Unmanned Aerial Vehicles). Most standard UAV autonomous landing systems take an approach, where the UAV detects a pre-set pattern on the landing zone, establishes relative positions and uses them to perform the landing. These methods present some drawbacks such as all of the processing being performed by the UAV itself, requiring high computational power from it. An additional problem arises from the fact most of these methods are only reliable when the UAV is already at relatively low altitudes since the pattern's features have to be clearly visible from the UAV's camera. The method presented throughout this dissertation relies on an RGB camera, placed in the landing zone pointing upwards towards the sky. Due to the fact, the sky is a fairly stagnant and uniform environment the unique motion patterns the UAV displays can be singled out and analysed using Background Subtraction and Optical Flow techniques. A terrestrial or surface robotic system can then analyse the images in real-time and relay commands to the UAV.

The result is a model-free method, i.e independent of the UAV's morphological aspect or pre-determined patterns, capable of aiding the UAV during the landing manoeuvre. The approach is reliable enough to be used as a stand-alone method, or be used along traditional methods achieving a more robust system. Experimental results obtained from a dataset encompassing 23 diverse videos showed the ability of the computer vision algorithm to perform the detection of the UAV in 93,44% of the 44557 evaluated frames with a tracking error of 6.6%. A high-level control system that employs the concept of an approach zone to the helipad was also developed. Within the zone every possible three-dimensional position corresponds to a velocity command for the UAV, with a given orientation and magnitude. The control system was tested in a simulated environment and it proved to be effective in performing the landing of the UAV within 13 cm from the goal.

Keywords: UAV landing systems, Robotic cooperation, Background Subtraction, Optical Flow, motion patterns, high-level control.

Resumo

Esta dissertação apresenta uma aproximação cooperativa para a aterragem autónoma de UAVs MR-VTOL. A maior parte dos sistemas deste tipo baseam-se numa abordagem, na qual o UAV detecta um padrão pré determinado na zona de aterragem, estabelece posições relativas e usa-as para realizar a aterragem. Estes métodos apresentam algumas desvantagens tais como, o facto de todo o processamento ser feito pelo próprio UAV, requerendo um alto desempenho computacional. Outro problema surge do facto da maior parte destes métodos só serem fiáveis a altitudes relativamente baixas, uma vez que as caracteristicas do padrão têm de ser bem visíveis pela câmara do UAV. O método apresentado ao longo desta dissertação usa uma câmera RGB, colocada na zona de aterragem apontando para cima em direcção ao céu. Dado que o céu é um fundo significativamente estático e uniforme, os padrão únicos que o UAV apresenta podem ser identificados e analisados, usando técnicas de *Background Subtraction* e *Optical Flow*. Um sistema robótico, terrestre ou aquático, pode então análisar as imagens em tempo-real e transmitir comandos para o UAV.

O resultado é um método livre de modelo ou seja, independente do aspecto morfológico do UAV ou de qualquer padrão pré determinado. A aproximação é fiável o suficiente para ser usado por si só ou então ser usado juntamente com métodos tradicionais alcançando um sistema mais robusto. Os resultado experimentais obtidos através de um conjunto de 23 vídeos, demonstram a capacidade do algoritmo de visão por computador de realizar a detecção do UAV em 93,44% dos 44557 frames de video analisados com um erro de *tracking* médio de 6,6%. Um sistema de controlo de alto nível que usa o conceito de zona de aproximação à aterragem foi ainda desenvolvido. Dentro da zona cada posição tri-dimensional corresponde a um comando de velocidade para o UAV, com uma determinada orientação e magnitude. O sistema de controlo foi testado num ambiente de simulação no qual provou ser eficaz na realização da aterragem do UAV, dentro de uma margem de 13 cm do objectivo.

Palavras-chave: Sistemas de aterragem para UAVs, Cooperação robótica, *Background Subtraction*, *Optical Flow*, padrões de movimentos, controlo de alto nível.

Contents

List of Figures

List of Tables

Glossary of terms and Acronyms

AI	Artificial Intelligence (AI) is the theory and development of computer systems able to perform tasks normally requiring human intelligence. Encompasses fields such as visual perception, speech recognition, decision-making, language translation etc.
AR	Augmented Reality (AR) is a direct or indirect live view of a physical, real-world environment whose elements are enhanced by computer-generated perceptual information.
Blob	Blob is a continuous, generally white pixel zone in a binary mask. Each blob is usually considered an object of interest for further analysis.
BS	Background Subtraction (BS) is a technique in Computer Vision used to extract objects of interest in an image or video. Generally there is a static background, in front of which objects move. The algorithm detects those moving objects by computing the difference between the current frames and a background model previously established.
Camshift	Tracking algorithm that uses colour signatures as input. It works by locating the maximum value of density functions, for example, finding the zone in a image where the density of pixels of a given colour are maximized.
CCD	Charge-Coupled Device (CCD) is a device used to move electrical charge, usually from within a device to another area where it can be manipulated. It can be used for example to convert charge into digital numbers.

CCL Connected Component Labelling (CCL) is an algorithm based on graph theory used for the detection of regions in binary digital images.

CMOS Complementary Metal-Oxide-Semiconductor(CMOS) is a technology for constructing integrated circuits. It uses complementary and symmetrical pairs of p-type and n-type metal oxide semiconductor field effect transistors for the creation of logic functions.

CPU Central Processing Unit (CPU) is a electronic circuit that carries out the instructions of a computer program by performing basic arithmetic, logical, control and input/output (I/O) operations. It is generally the component that perform most of the computation in a computer.

CV Computer vision (CV) is the field that develops methods to acquire high-level understanding from digital images or videos.

FoV Field of View (FoV) is the extent of observable world by an given observer, a person or camera. It is generally measured in degrees.

GPS Global Positioning System (GPS), is a satellite-based radio navigation system, designed for outdoors. It provides geo-location and time information to receivers anywhere on Earth.

GPU Graphics Processing Unit (GPU) is a circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. It has highly parallel architecture, making highly adept to perform multiple task at the same time.

IMU Inertial Measurement Unit (IMU) is an electronic device that measures and reports forces, angular rate, and the magnetic field surrounding it. Generally composed by a combination of accelerometers, gyroscopes and magnetometers.

KDE Kernel Density Estimation (KDE) is a non-parametric method used to estimate the probability density function of a random variable.

KF Kalman Filter (KF) is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone.

LIDAR Light Detection And Ranging (LIDAR) is a optic technology for detection, that measures reflected light to obtain the distance between objects. Generally is performed by emitting a laser light and measuring the time spent between the emission and the reception of the laser's reflection.

ML Machine Learning (ML) is the field of computer science that tries gift computer systems with the ability to "learn", i.e., progressively improve performance in a specific task.

MoG Mixture of Gaussian (MoG) is a method that consists on combining multiple Gaussians into a single Probability Density Function, improving the representation of sub-populations present within the general one. In CV it is associated with Background Subtraction methods.

MR-VTOL Multi Rotor-Vertical Take Off Landing (MR-VTOL) are UAVs equipped with multiple rotors attached to propellers set parallel to the ground. Thus are capable of landing and taking off vertically just like a helicopter.

NN Neural Network (NN) is as computational model inspired by animals central nervous system capable progressively learning and recognizing patterns.

OF Optical flow (OF) is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene.

OpenCV	Open Computer Vision (OpenCV) is a library of programming functions aimed at real-time computer vision. It provides common low-level operations as well as implementations of high-level algorithms.
PDF	Probability Density Function (PDF) is a function whose value at any given point in the sample space represents the likelihood of random variable being equal to the value at that point.
PID	Proportional Integrative Derivative (PID) is a control loop feedback mechanism used in a wide variety of applications. It computes the error value as the difference between a desired set point and the measured value and applies a correction based on three variable denominated P,I,D, hence its name.
RANSAC	RANdom SAmple Consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers.
RGB	Red Green Blue (RGB) is an additive colour model in which red, green and blue light are added together to reproduce a broad array of colours.
ROS	Robot Operating System (ROS) is a middleware, i.e. collection of software frameworks, for the development of robot software. It provides services designed for heterogeneous computer clusters such as hardware abstraction, low-level device control, implementation of commonly used functionalities, message-passing between processes, and package management.
SURF	Speeded Up Robust Features (SURF) is a patented local feature detector and descriptor. It can be used for tasks such as object recognition, image registration, classification or 3D reconstruction.
UAV	Unmanned Aerial Vehicle (UAV) is a robotic aircraft without a human pilot aboard. It is generally capable of sensory data acquisition and information processing.

UGV Unmanned Ground Vehicle (UGV) is a vehicle that operates while in contact with the ground without onboard human presence.

USV Unmanned Surface Vehicle (USV) is a robotic element designed to navigate over the water's surface. Generally capable of sensory data acquisition and information processing.

Voxel Representation of a value in a three-dimensional space in a regular grid.

List of Symbols

$\sum_{i,s,t}$	Covariance matrix of the i_{th} Gaussian mixture of pixel s at the time t.
τ	Background Subtraction distance threshold.
η	Gaussian Probability Density Function.
λ	Background Subtraction foreground threshold. Represents the maximum deviation a pixel can have to belong to the background.
γ	Non-negative weight factor between the even and the odd parts of the Farneback signal.
$\eta(I_{s,t}, \mu_{i,s,t}, \sum_{i,s,t})$	Gaussian Probability Density Function of pixel s at the time t.
ΔT	Time lapse.
Δ_{RGB}	Difference between RGB channels maximum and minimum value.
α_g	Gompertz curve maximum value.
γ_g	Gompertz curve scale factor.
β_g	Gompertz curve offset.
$\omega_{i,s,t}$	Weight of the i_{th} Gaussian mixture.
$\mu_{i,s,t}$	Mean of value of the i_{th} Gaussian mixture of pixel s at the time t.
α_{mog}	MoG primary learning rate.

ρ_{mog}	MoG secondary learning rate.
ψ_{v_c}	Yaw orientation of the velocity command.
θ_{v_c}	Pitch orientation of the velocity command.
ϕ_{v_c}	Roll orientation of the velocity command.
Δmax_{pos}	Maximum admitted variance in the UAV position between frames.
A	Farneback signal matrix.
A_{kf}	Transition matrix of Kalman Filter.
B	Blue channel of a pixel.
B_g	Image background model.
Cl	Cloud mask.
D_f	Frame diagonal (pixels).
D_{uav}	UAV's real dimension (m).
$F(x,y)$	Optical flow at coordinate (x,y) in between two frames.
G	Green channel of a pixel.
$H(s)$	Entropy of sample s.
H_{kf}	Measurement matrix of Kalman Filter.
H_{mog}	Most reliable Gaussians used for background model computation.
I	Pixel intensity/colour value.
$I(x,y,t)$	Pixel intensity at coordinates (x, y) at the time t.

$I_{s,t}$ Intensity of the pixel s at the time t.

K Number of mixtures used in a Mixture of Gaussian.

$L(s)$ Length of the trajectory of sample s.

N_{avg} Number of frames used for the computation of the average UAV position.

N_{of} Number of frames used for Optical Flow analysis.

$P(I_{s,t})$ Probability of colour I, in the pixel s at the time t.

P_{uav} UAV's 3D position.

Q_{kf} Covariance matrix of Kalman Filter.

R Red channel of a pixel.

S_c Cluster set.

T Orientation Tensor.

T_{mog} Parametrizable threshold measuring the minimum amount of data to be accounted for the background computation in a Mixture of Gaussians.

V Video Sequence.

V_x Motion flow in the x axis.

V_y Motion flow in the y axis.

X X axis of the helipad 3D coordinates frame.

X_{kf} State matrix of Kalman Filter.

X_s	Foreground mask.
Y	Y axis of the helipad 3D coordinates frame.
Z	z axis of the helipad 3D coordinates frame.
a_{az}	Approach zone curve scale factor.
$az(x)$	UAV's landing approach zone.
b	Vector used for the signal approximation of a neighbourhood of a pixel in the Farneback algorithm.
b_{az}	Approach zone curve offset factor.
c	Scalar used for the signal approximation of a neighbourhood of a pixel in the Farneback algorithm.
c_k	Cluster number k.
c_{max}	Maximum value of the RGB channels.
c_{min}	Maximum value of the RGB channels.
c_{th}	Clustering distance threshold.
$c_{th_{min}}$	Minimum value of c_{th}.
$c_{th_{rat}}$	Contribution ratio for c_{th} by the UAV size when compared to the frame size.
$d(s)$	Minimum radius of sample s trajectory.
d_{az}	Ratio between az(x) and d_h.
d_{bs}	Background Subtraction distance metric.

d_{error}	Distance between real position of the in UAV the image and tracker output in a frame.
d_h	Distance between UAV and helipad plane parallel to it.
d_{uav}	UAV's apparent size in a frame (pixels).
f	Camera's focal length(m).
h_{bs}	Background Subtraction history for background model computation.
i_{th}	Mixture number in MoG.
k_m	K-Means number of clusters.
k_{v_c}	θ_{v_c} normalizing factor.
m_{v_c}	Velocity command magnitude.
n	Frame number.
p_{avg}	Weighted average UAV 2D position.
$p_{c_{k(x,y)}}$	Central position of cluster k.
p_{kf}	UAV 2D position computed by the Kalman Filter.
p_s	Interval of pixels between samples for Optical Flow analysis.
r_{c_k}	Radius of cluster k.
rb_{max}	Maximum $\frac{R}{B}$ ratio for a pixel to belong to the cloud mask.
rb_{min}	Maximum $\frac{R}{B}$ ratio for a pixel to belong to the cloud mask.
s	Individual pixel/sample of a frame.

$s_1, ..., s_t$	Pixel past values.
$s_{n(x,y)}$	Pixel of frame n at position (x,y).
sat	Saturation value of a colour.
sat_{max}	Maximum saturation value for a pixel to belong to the cloud mask.
t	Time instant.
$u(s)$	Uniformity of sample s.
v_c	Velocity command issued to the UAV.
v_x	UAV's speed on the x axis in between frames (pixel/miliseconds).
v_y	UAV's speed on the y axis in between frames (pixel/miliseconds).
w_n	Weight of the frame n for the computation of the average UAV position.
x	2D Position on the x axis of an image.
x_{uav}	UAV's 3D position in the x axis.
y	2D Position on the y axis of an image.
y_{uav}	UAV's 3D position in the y axis.
z_{uav}	UAV's 3D position in the z axis.
dx	Distance to the image centre in the x axis(pixels).

CHAPTER 1

INTRODUCTION

1.1 The applications of UAVs

Just like most technological advancements and innovations in the history of mankind, Unmanned Aerial Vehicle (UAV)s were originally developed for war. UAVs possess the ability to transmit to the battlefield commander, real-time intelligence, surveillance and reconnaissance information from hostile areas. They can also act as communication relays, designate targets for neutralization, or even actively participate in the battlefield by engaging targets themselves. All of this can be achieved without ever putting at risk the lives of a human aircrew [1]. For all these reasons they quickly became an indispensable asset in the hands of military forces all around the globe.

In recent years, UAVs have become a prominent technology not only to military but also in robotics and the general public. Multi-rotor vehicles, in particular, capable of vertical take-off and landing (MR-VTOL), offer high versatility, have relatively low production cost, and are easier to teleoperate than their fixed-wing counterparts. Thus they have become an attractive asset in a vast number of scientific and civil applications as well as for recreational purposes. Nowadays UAVs are used in the most diverse areas such as:

Natural Disasters One of the most obvious assets of a UAV is its mobility, being able to get basically anywhere no matter the access conditions. UAVs can be the first to reach zones affected by natural disasters such as hurricanes or tsunamis. Imagery obtained by them can be used to produce hazard maps, dense surface models, detailed building rendering, elevation models, etc. [2]. A practical example of this application was in the aftermath of Hurricane Katrina, where UAVs were used to collect imagery of the structural damage on buildings [3]. Another example is in the aftermath of Hurricane Wilma where a UAV/Unmanned Surface Vehicle (USV)

team was deployed to conduct the post-disaster inspection of bridges, seawalls, and damaged piers, a similar application to the one presented in [4]. In this case, the UAV and USV cooperative work was able to extend or complement the individual capabilities of both. For example, the UAV would acquire imagery that the USV could use to plan its route through the affected zone [5].

Delivery With the increase of e-commerce, giant companies like Amazon and UPS are faced with the problem of increasing their delivery capabilities. One possible solution is the use of UAVs. After the introduction of Amazon's own UAV delivery service, called Prime Air, some products bought on Amazon's website can be delivered in just 30 minutes. Competition with other companies in the goods delivery industry, like UPS, increased tremendously due to the introduction of services like these [6]. There are actually studies in ways to use UAVs to extend delivery capabilities, with delivery trucks acting as base stations. While the delivery staff do deliveries, multiple UAVs can be released and make other deliveries in the neighbourhood and quickly return, if a problem arises with one of the UAVs, the delivery staff should be close by and prepared to solve it. In fact, 80% of the packages delivered through internet shopping are light enough for a UAV to carry [7], so it is foreseeable that this method and other similar systems for delivery may grow in the future.

Farming and Agriculture Even in Agriculture, Robotics, in particular UAVs, can have an important role. In [8] a system with a symbiotic relationship between a UAV and a Unmanned Ground Vehicle (UGV), is described. Its goal was the collection of nitrogen levels across a farm for precision agriculture. UAVs are also capable of monitoring and acquiring footage to create panoramas, through geo-referencing and image stitching, that can be used to take decisions about future plantations [9].

Surveillance Surveillance and monitoring is another field that could benefit from UAV technology. In [10] teams of autonomous UAVs are used to monitor complex urban zones, maximizing their coverage through coordination algorithms. This could, for example, be applied in conflict zones, to detect imminent threats. Other possible application of UAVs in surveillance, is in the maintenance of infrastructures. For instance, [11] presents a method for the detection of power lines using Computer Vision (CV) on footage obtained from UAVs. Other applications such as traffic and crowd monitoring, accident detection, etc, could also be an interesting field to be developed in the future [12].

These are just some examples of possible UAV applications. With the continued development of technology, and the increased processing power of our computers, UAV applications become almost limitless. It may not be too far-fetched to believe that in the future our skies will be filled with UAVs designed for the most various tasks of our everyday life.

1.2 The problem

Generally, independently of its purpose or capabilities, there are two manoeuvres any UAV must be able to perform: taking off and landing. Landing, in particular, is a especially delicate process since a failed landing can result in damage to the UAV itself, other equipment in the field or even worse, put at risk the safety of people in the surroundings. Landing requires precise control of the UAV and reliable knowledge of the location of both the UAV and the landing zone.

One of the possible ways to perform this manoeuvre is manually through a human operator. This has several disadvantages, the most obvious being human error. The success of the landing rests entirely in the hands of the operator and his skill, which could be proven unreliable or unavailable. Furthermore, having one dedicated person to perform every single landing, wastes manpower that could be spent in other, more productive, endeavours. An alternative is to develop an autonomous system capable of landing on a fixed pad, whose position the UAV can either know beforehand or detect through the processing of sensory information, such as RGB (Red Green Blue) imagery. One more interesting and difficult approach would be to have a mobile platform, a USV or UGV, acting as a mobile base for the UAV. This last approach requires cooperation between the two robotic systems, which represents an overall increase in complexity. However, a cooperative system like this can vastly expand the capabilities of both robots, improving their performance on the field [4]. The design of such a system requires the use of different technologies like Computer Vision (CV) algorithms, robot communication, cooperation behaviours, Artificial Intelligence (AI), etc.

Most current autonomous landing systems are pattern-based, where a camera equipped on the UAV is used to detect a certain pattern present on the landing zone. After the pattern is detected the UAV computes its relative positions and performs the landing according to them. However, these methods have certain drawbacks. First of all, they put all the stress on the UAV itself forcing it to do most of the computational work. UAVs are generally equipped either with single-board micro controllers or lightweight micro computers whose processing capabilities pale when in comparison with full fledged-computers. And so, there may be situations where the UAV may not have enough computational power to perform complex algorithms in real time or the cameras employed on it may not have enough resolution or picture quality to perform reliable computer vision processing. It is even possible to think that UAVs due to their unstable nature may be more prone to accidents, collisions or system failures during flight, ending up with reduced sensory and processing capabilities, hindering the landing manoeuvre. For instance, the camera's footage or the camera's supporting gimbal may stop working mid-flight. On the other hand, pattern-based systems pose the question of what, and how big should the pattern be. Small patterns may not be able to be identified by the UAV at high altitudes, while larger patterns may create a problem at low altitudes by not fitting in the totality of the camera's Field of View (FoV). The pattern should also not blend with the environment to

avoid making its detection harder or downright impossible.

1.3 Proposed solution

This dissertation proposes a cooperative vision-based landing system for MR-VTOL UAVs, that attempts to solve some of the issues traditional pattern-based systems face. Rather than performing the landing by itself, the UAV receives and relies on information made available by the helipad. The helipad becomes, in essence, a smart element capable of sensory data acquisition and information processing that may, or may not, be coupled to a mobile platform. A camera is put at the centre of the helipad, pointing upwards towards the sky with the objective of detecting the UAV throughout the landing. Therefore a significant part of the computation is moved away from the UAV itself, going to the helipad landing system instead, allowing the UAV to save computational resources and battery life.

The detection method takes advantage of the relatively motionless aspect of the sky, using a Background Subtraction (BS) algorithm, to detect the UAV in the surrounding airspace without the need to rely on a pre-set physical pattern. To distinguish it from other objects, like aeroplanes, the UAV's chaotic movement patterns, in most part created by the rotation of its propellers, are measured and quantified using an optical flow algorithm. After the UAV is correctly detected its position can be extracted and used to relay commands to it, allowing the landing to be performed. A high-level control approach that takes into account the relative positions between, UAV and helipad, is also proposed. The control module establishes an approach zone above the helipad airspace where each deviation from the centre corresponds to a certain velocity command. For each position within the zone, a velocity command with a set orientation and magnitude is issued.

The presented method is generic and modular, designed with the goal of either working as a stand-alone system, performing the landing by itself, or being part of a more complex system. For instance, the system can work together with a traditional approach overcoming or minimizing each other's shortcomings using multiple cues to perform a more reliable autonomous landing.

1.4 Dissertation Outline

This dissertation is organized as follows:

- **Chapter 2** reviews the state of the art about UAV automatic landing systems as well as control mechanisms used in this kind of systems. It gives yet a perspective on robot cooperation for the solving of problems like these;

- **Chapter 3** presents the supporting concepts behind the development of the algorithms applied in this work;

- **Chapter 4** describes the general model proposed, and the reasons for the choices made during its development;
- **Chapter 5** goes over the experimental results obtained using the proposed model;
- **Chapter 6** aggregates the conclusions, the main contributions of this dissertation, and possible further research topics on the subject.

Chapter 2

State of the Art

Due to its innate complexity, various different solutions for autonomous landing systems have been proposed and developed. The need to take into account multiple sensors, having to react to difficulties and external factors in real time, the requirement of precise controls of the UAV among other issues, make the design of such systems a challenge. This chapter tries to give an overview of the state of the art of vision-based autonomous landing systems, more specifically, the detection method employed as well as the control mechanisms used to manoeuvre the UAV in its descent. At the end of the chapter, a brief synopsis of cooperative systems, and their possible contribution in an autonomous landing system is presented.

2.1 Detection Methods

Computer Vision is a developing field that is generally at the centre of a lot, if not most, of the current auto-landing systems. Its versatility and the wide availability of RGB cameras, when compared to other sensors, makes it especially attractive. Normally vision-based autonomous landing systems can be decomposed into three distinct steps:

1. Analyse the image coming from the UAV or some other asset in the field and extract known features, landmarks or patterns.

2. Estimate the relative position of the extracted features with respect to the camera.

3. After determining all relative positions, it becomes a control problem with the goal of putting the UAV pose on top of the landing pad's position.

The most common approach is to put a camera on the UAV itself, and a recognizable pattern on top of the landing pad. The camera usually points downwards, and preferably

should be set on a gimbal to minimize trepidation in the captured footage. The UAV starts by approaching the landing pad using an alternative control method that can be for instance, based on GPS (Global Positioning System). When it flies over the landing pad, proceeds to identify the known pattern and then calculate its own relative position to it.

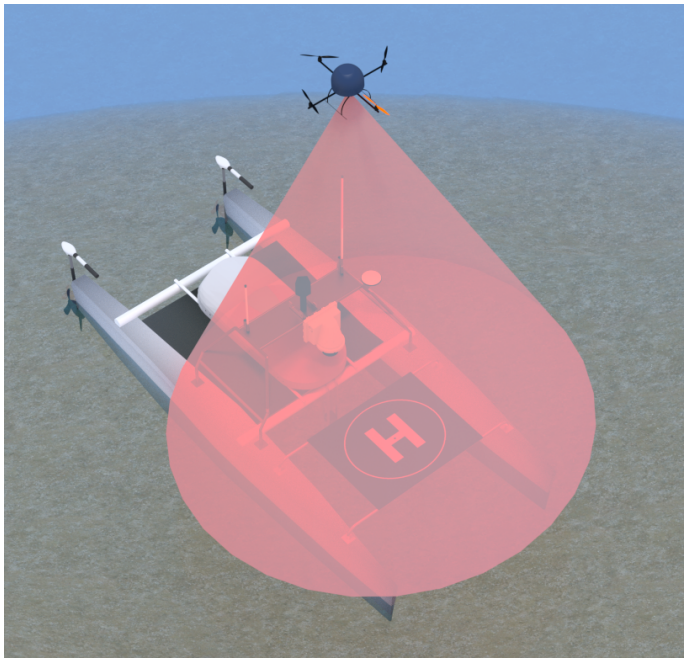


Figure 2.1: Traditional approach for the autonomous landing of MR-VTOL UAVs. The UAV flies to the proximity of the landing zone using, for instance, a GPS-based navigation method. It starts searching for the pre-determined pattern, in this case an "H" inscribed in a circle, with its camera whose FoV is represented by the red cone. After the relative positions are established the landing procedure can start.

The difference between the various approaches mostly lies in the chosen pattern, and the control method employed. The choice of pattern, in particular, is very important. It has to be distinct enough, to avoid being mistaken with other objects present in the environment, while at the same time simple enough to not overbear the system with great amounts of processing. One of the most common, and simpler, patterns used in this kind of systems is the "H" letter (i.e., initial from Helipad or Heliport), as the one seen in Fig. 2.1. [13] uses a green "H" to perform detection. Knowing the colour of the target *a priori* eases the identification process since a simple histogram analysis of the image, plus a tracking algorithm like Camshift [14], that uses colour signatures as input, allows for good results to be achieved. On top of Camshift a Speeded Up Robust Features (SURF) algorithm is used to locate and identify the features of the "H" pattern. This method only works in simple environments, whereas in more complex environments where other objects in the green spectrum might be present, the method is not robust enough. [15] [16] also use the "H" pattern, but instead of relying on a given colour it is assumed the intensity values of the landing pad are different from that of the neighbouring regions. The footage obtained from the UAV is converted into a grey scale, using luminance values,

followed by a fixed threshold binarization. The binarized image might contain objects other than the "H" so a Connected Component Labelling (CCL) is used to identify every object, followed by the computation of the invariant moments of every object. From that information, object descriptors are then created and the one whose values most resemble the "H" pattern calibration values is chosen and tracked. Since it doesn't rely on the pattern having a specific colour, this method is more environment independent, meaning that potentially works on a broader spectrum of surroundings in spite of their colour signatures.

Nonetheless, if the surroundings or multiple objects present in the image, have similar luminance values than the "H", detection might be compromised. A similar example is shown in [17] but instead of using luminance values, a pattern with a negative "H" formed by four white rectangles is used allowing the use of an edge detection algorithm to perform the detection. The edges, the corners of the rectangles, as well as their relative positions are computed and measured to assess if they fit within expected values. Of all letter pattern-based methods this is the most robust since it does not depend on the colours or luminance values of the surroundings. The pattern itself creates visible contrasts, which are used to improve detection. The same idea is applied in [18] where a pattern composed of a black square with smaller white squares in its interior was chosen. The geometric pattern and colour contrasts allow a relatively simple identification process. Thresholding, followed by segmentation, and finally, corner detection and feature labelling are used. The system is described to be accurate within 5 cm. Yet another similar approach is shown in [19] where an April Tag is used instead for a similar effect.

Another question when choosing the pattern, other than its content, is its size. A size too small might not be able to be identified from high altitudes, requiring the UAV to be at too low altitudes to begin the detection. On the other hand, if the pattern is too big, it may not fit entirely in the camera's FoV at lower altitudes, or even not fit on the landing pad. This may, or not, be an issue depending on the size of the landing platform and margin of error required. An approach that attempts to solve this problem is presented in [20]. It uses a pattern consisting of several concentric white rings on a black background, where each of the white rings has a unique ratio of its inner to outer border radius. This characteristic makes the identification of a given ring independent of other rings, making it possible to identify the pattern even when it is not completely in view of the UAV. The uniqueness of the pattern, it is said to reduce the number of false positives in both natural and man-made environments.

The use of patterns and symbols can be helpful, since creates a simple goal, to detect it, but can also be seen as a limitation. The need to follow a very specific marking makes it so, a change in environment might require a change in the pattern. For instance, the pattern appearance and colours might blend with certain environments, continuous sun exposure and wear over time might also degrade the pattern features. Varying lighting conditions, its aspect at high altitudes among other factors can make it harder to detect and recognize. Some patterns are also overly complex requiring long learning algorithms

a) b) c)

Figure 2.2: Examples of common helipad patterns. Most work by having a presence of high contrasts to facilitate the detection and feature extraction. a) Square shapes together with high contrasts are used to ease corner detection. b) Target-like pattern used in [20], to minimize problems related with pattern size. c) Negative "H" like the one used in [17].

to be run. Motion-based approaches can help overcome the aforementioned problems. Instead of relying on a specific pattern, by using an upwards-facing camera and studying the UAV's unique motion signature, it becomes possible to create a set of rules that allow for the detection and distinction of the UAV and its relative position to the camera. This method is model-free in the sense that it potentially works for all MR-VTOL UAV's no matter their configuration (i.e., quad, hexacopters, etc.), colour or shape. Furthermore, the fact the camera is static and facing perpendicularly the ground plane means the process of position estimation is greatly simplified. Most of data processing requirements are also taken away from the UAV itself, allowing, if necessary, more complex algorithms to be run, on more powerful computers on the field, freeing the UAV to focus on other tasks.

2.2 Control

The control of the UAV is a central part of the development of an autonomous landing system. The goal is to put the UAV on top of the landing platform with the most safety possible. For simplicity sake, it is assumed the helipad is parallel to the ground and there are no obstacles present in the airspace above it. The UAV starts from a high altitude, being guided to the helipad by other navigation methods like for example GPS-based ones. After the features, patterns or landmarks are detected it tries to align itself with the centre of the helipad, slowly starting its descent while trying to keep the needed alignment until it has safely landed.

Control can be divided into two areas:

- **Low-level:** how the motors should be actuated to perform a movement in a certain direction;
- **High-level:** which movements should be performed in a given situation.

2.2.1 Low-level control

One possible first step is to build a mathematical model of the UAV, as shown in [21], where the Newton-Euler and Euler-Lagrange equations were used to get the multi-rotor flight dynamics, together with quaternions to parametrize the UAV rotations. Quaternions were used instead of Euler angles as to avoid singularities. A similar approach was used in [22] where the Newton movement laws were used to similar effect. After the dynamic model of the system is built, either by estimation or by building a mathematical model, it is possible to apply a control loop. A classic solution is the use of a Proportional Integrative Derivative (PID) or any of its possible derivations (PD,PI) [21] [23]. Due to their simplicity and ease of use, PID controllers are one of the most common approaches to perform UAV control, however, despite its simplicity, correct tuning of a PID controller can be a complex matter. In [24] a deadbeat controller is implemented resorting to multiple control loops. The tuning of multiple PID may consume a vast amount of time since each PID has three parameters needing to be tuned and generally, there is a need to perform a great number of test flights to achieve the expected performance. Usually, UAV PIDs are tuned to perform under specific conditions, if these conditions change there might be a need to perform some additional tuning, for example, if the payload carried by the UAV changes. Since different weights might affect the flight dynamics of the UAV in different ways if a given UAV has to change its payload the PID might need to be tuned all over again. [25] proposes a self tuning PID, based on fuzzy logic to solve this problem. An additional Fuzzy Logic layer is added to the control loop with the goal to make adjustments to the control PID parameters. This way was possible to achieve the expected performance despite payload changes. In [26], fuzzy logic was used as well but as higher lever control system, using the vision sensory data as input, together with a PID for the actual low-level control. Using a Neural Network (NN) is another possibility for the control loop, their learning capabilities are ideal for rapid responses and adaptability in uncertain environments. Such a system was developed in [27], where the information coming from the onboard sensors is used to feed the network and control the UAV's direction and orientation. In [28] a comparison of performance between two controllers one based on a mathematical approach and other based on a neural network system was made, with the NN approach showing to have a similar performance to a purely mathematical approach.

2.2.2 High-level control

Most high-level controls are composed by a set of behaviours whose nature depends on the task at hand. In a landing manoeuvre there are generally, only two behaviours required: altitude control to perform the descent and positional control to perform the alignment with the helipad. Once the UAV is over the helipad, it should try to align itself with helipad's centre while maintaining altitude. After relative positions are accurately computed it should start its descent while making adjustments on horizontal position

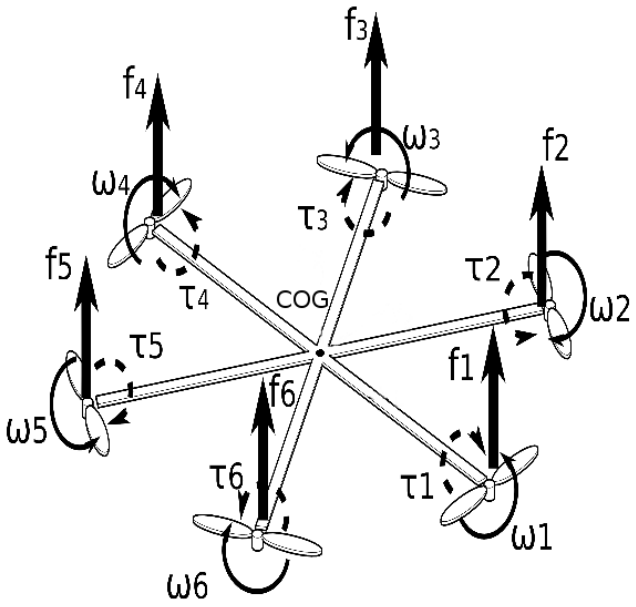


Figure 2.3: Mathematical analysis of a MR-VTOL flight dynamics, adapted from [21]. The UAV has to able to counteract gravity acceleration and so each propeller produces force in opposite direction producing lift f_n. Each propeller rotates in the opposite direction(ω_n) of its neighbours to cancel out the UAV's torque effect. The UAV can be controlled by individually varying the rotation speed of each propeller.

as needed. [20] uses a two module system: altitude controller, using a sonar sensor for measurements and a PID loop for control; Position and Velocity controller using an optical flow sensor facing the ground as input, both are also controlled via a standard controller (PD and PID respectively). In the landing manoeuvre, the position controller tries to keep the alignment with helipad performing corrections as needed, while the altitude controller slowly performs the descent. [15] and [16] propose a similar architecture but further subdivides the positional controller in lateral velocity controller and heading controller, this way a more granular control of the UAV is achieved.

2.3 Cooperative Systems

Systems composed of multiple robots can be especially interesting to consider. More robots in the field mean more data gathering capabilities, more processing power, etc. Having members with different capabilities, for example, UAVs working together with USVs or UGVs, extends the capabilities of both allowing them to achieve tasks not possible when working alone. Currently UAVs face two major problems: short flight time due to battery capacity and relatively low computational capacity when compared to full-fledged computers. Both of these problems can be partially overcome with the use of a cooperative system.

Coordination between members of a robotic team, however, can be a real problem. The

problem stems from the fact the more data is available the harder is select the relevant information and process it. In centralized systems where all information is available to all members of the system, they can quickly be overloaded with superfluous data, making processing and response times much slower. Furthermore, the scalability of the system may become an impossibility. On the other hand, in a decentralized system where each element of the system has only partial information, the question becomes what information should and should not be shared, and which information should be known by all. Another problem can be what to do when the communication systems fails, how long should old transmitted data be considered etc.

[29] introduces a UAV/UGV three-layer hierarchical cooperative control framework with the intent of being deployed in firefighting. Multiple UAV/UGV teams are deployed in the field, both with distinct objectives, the UAV should travel and drop water to specific points, while the UGV should establish a perimeter and avoid fire proliferation. Hierarchically above them is an airship, responsible for assigning individual tasks to each member in the field while receiving all the data from the UAV/UGV teams.

In [30], a symbiotic relationship between a USV and UAV is described. The USV piggybacks the UAV, saving the UAV's battery life. On the other hand, the UAV is able to take off, gather sensory information of the surroundings and build a navigation map, which will later be used by the USV, to create a cost map so it can plan a path. With the help of the UAV, the USV is able to plan much more further ahead than when working alone. The USV helps the UAV's landing, using its upwards facing camera to detect a pattern, in this case, an Augmented Reality (AR) marker, set on the bottom of the UAV. In turn, the UAV with its own camera learns the appearance of the landing zone during take-off, building a saliency map. This saliency map is then used to help the UAV to recognize the heliport during landing.

A similar relationship is presented in [8], but instead of a USV a UGV is used. The goal of this system is data collection for precision agriculture, detecting Nitrogen levels across a farm, with the purpose of aiding in fertilizer usage. The main feature of the system just like [30] is the fact the UGV carries the UAV to specific deployment points, decreasing the strain of the UAV battery life, which again, is generally short. Specific points indicated by the UAV should be inspected more closely by the UGV, for this, a path planning algorithm was developed where the UGV takes into account not only his own way-points but also, recovery points for the UAV, minimizing the total time spent in travelling and taking measurements. In summary, the use of cooperative robotic teams can be highly advantageous for the performance of the system as a whole. In the particular case of an autonomous landing system the use of multiple cues coming from different robots can add robustness and redundancy to the system, allowing it to achieve greater results.

CHAPTER 3

SUPPORTING CONCEPTS

This chapter introduces some key aspects, general concepts and algorithms used as the basis for the development of this dissertation. These are the use of motion patterns for UAV detection and data analysis for its respective identification. Background Subtraction techniques are introduced in section 3.1, whereas Optical Flow (OF) is overviewed in section 3.2. Finally in section 3.3, some phenomena, fast moving and rotating objects provoke when exposed to camera footage are explained.

3.1 Background Subtraction

Background Subtraction is a technique used to detect moving objects from an image, captured by a static camera. Generally speaking BS is performed by subtracting the current frame in relation to a background image, formally called "background model". This background model is an image in the absence of any moving objects, only with its static components. However it should not remain unchanged. In the best BS algorithms the background model must be capable of adapting to varying luminosity conditions, geometry settings, as well as new objects that become part of the background [31].

There are numerous approaches to implement BS algorithms as well as countless variations and improvements. The problem lies in choosing the criteria used to detect the changed pixels and the method to generate the background model. A simple image subtraction wouldn't suffice since the result would be too sensitive to noise and small movements of background objects. A BS algorithm generally makes the following assumptions, the video sequence, V, has a fixed Background , B_g, in front of which moving objects are observed. At any given time, t, a moving object's colour differs from the colour of the background. This can be expressed by:

$$X_s = \begin{cases} 1 & if \quad d_{bs}(V_s, t, B_g) > \tau \\ 0 & otherwise, \end{cases} \tag{3.1}$$

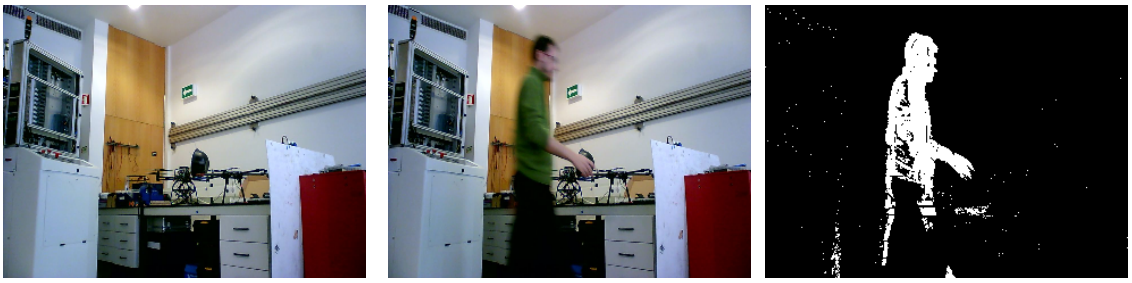
where X_s is the motion mask, meaning the pixels belonging to the moving objects; d_{bs} is the distance at a given pixel s between the current frame and B_g and τ is a set threshold.

Most BS techniques follow this approach, the difference lying in how B_g is modelled and the metric used to calculate d_{bs} [32]. One of the most common approaches is to model every background pixel as Probability Density Function (PDF). For example, the Gaussian function can be used together with a log likelihood or Mahalanois distance as the metric for d_{bs}. To reduce the effect of possible slight movements in the background objects (waves, branches, clouds etc.), multi-modal PDF can be used for the background model representation, for example in [33] each pixel is modelled by a mixture of K Gaussian distributions. The choice of method depends entirely on the application. Simpler methods like Gaussian average or the median filter offer good reliability while not being too computationally demanding. Other methods like MoG and Kernel Density Estimation (KDE), show better results when compared to simpler methods, being much more resistant to noise. They do, however, have slightly higher memory and processing demands and are also harder to implement. All these algorithms have good performance if the camera is static, but it is easy to understand if the camera starts moving they won't work since the background will be constantly changing. Multiple authors have proposed BS algorithms that work with moving cameras. [34] presents two algorithms capable of distinguishing pixels in consecutive frames into foreground or background by detecting and taking into account independent motion relative to a statistical model of the background appearance. [35] takes into account that all the static parts of the background lie in the same subspace, to classify the pixels belonging to the foreground and to the background. RANdom SAmple Consensus (RANSAC) algorithm is then run to estimate the background and foreground trajectories. With them, it is possible to create both the background and foreground model.

3.1.1 Mixture of Gaussian

In probability and statistics, a mixture distribution is the merge of multiple mathematical distributions to form a single one. They are particularity useful for the representation of data populations that present multi-modal behaviour i.e., data whose its random variable, has multiple distinct high probability values. Each one of the individual distributions that form the final PDF is called a mixture or a component, and are characterised by their mixture weight (probability) [36].

In CV a mixture of distributions, in particular of Gaussians is associated with a family of BS algorithms commonly called MoG. From all the existent background subtraction techniques MoG is one the most widely used, mostly due to its relative ease of implementation, moderately low computational requirements, high resistance to noise and slight



a) b) c)

Figure 3.1: Background Subtraction example. a) The background model represents the image captured by the camera only with its unmoving objects. b) The current frame being analysed, containing one person walking by. c) Since the pixels belonging to the moving person do not fit into the background model, they are marked as white in the foreground mask.

movements, all the while still being capable of achieving good results.

MoG makes the following assumptions: a "foreground model" is segmented by the exception of the background model, meaning every component which does not fit into the background is foreground. This in turn, means the foreground model is not formally modelled either by colour, texture or edges. Model analysis is made per-pixel instead of regions, with each pixel being individually evaluated and decided whether it belongs to the foreground or the background. This decision is made on a frame by frame basis, without assumptions about the object shape, colour or positions. The background model is generated through a cumulative average of the past h_{bs} frames. The objects are segmented based on the distance d_{bs} of the individual pixels on the current frame in relation to the background model [37], with pixel whose values do not fit the background distributions being considered foreground.

The mathematical theory behind MoG is described in [38] and [33], while some improvements to the algorithm are presented in [39]. MoG uses a mixture of normal (Gaussian) distributions to form a multi-modal background Fig.3.3. For every pixel, each distribution in its background mixture corresponds to the probability of observing a particular intensity of colour.

The values of a particular pixel are defined either as scalars, in grey images, or vectors, for colour images. At a given time, t, what is known about a particular pixel, $s_(x_0, y_0)$ is its history:

$$s_1, ..., s_t = I(x_0, y_0, i) : 1 \leq i \leq t, \tag{3.2}$$

The recent history of each pixel, $s_1, ..., s_t$, is modelled by a mixture of K Gaussian distributions. The probability $P(I_{s,t})$ of observing the occurrence of a given colour I at a given pixel s, at the time t is given by:

$$P(I_{s,t}) = \sum_{i=1}^{K} \omega_{i,s,t} * \eta(I_{s,t}, \mu_{i,s,t}, \sum_{i,s,t}) \tag{3.3}$$

where K is the number of distributions used; η is the Gaussian probability density function; $\omega_{i,s,t}$ is an estimate of the weight of the i_{th} Gaussian in the mixture at the time t; $\mu_{i,s,t}$ is the mean value of the i_{th} Gaussian in the mixture at time t and $\sum_{i,s,t}$ is the covariance matrix of the i_{th} Gaussian in the mixture at time t; To improve the computational performance of the algorithm, in RGB images, it is assumed the red, green, and blue channel values of a given pixel are independent and have the same variance, expressed by:

$$\sum_{K,t} = \sigma_K^2 * I \tag{3.4}$$

After every new frame, if $I_{s,t}$ is within λ of the standard deviation of $\mu_{i,s,t}$, the i_{th} component is update as follows:

$$\omega_{i,s,t} = (1 - \alpha_{mog})\omega_{i,t-1} + \alpha_{mog}, \tag{3.5}$$

$$\mu_{i,s,t} = (1 - \rho_{mog})\mu_{i,s,t-1} + \rho_{mog} I_{i,s,t}, \tag{3.6}$$

$$\sigma_{i,s,t}^2 = (1 - \rho_{mog})\sigma_{i,s,t-1}^2 + \rho_{mog}(I_{i,s,t} - \mu_{i,s,t})^2, \tag{3.7}$$

Where α_{mog} is a user-defined learning rate and ρ_{mog} is a secondary learning rate defined as $\rho_{mog} = \alpha_{mog}\ \eta(I_{s,t}, \mu_{i,s,t}, \sum_{i,s,t})$. If it is not within the standard deviation the update is instead:

$$\omega_{i,t} = (1 - \alpha_{mog})\omega_{i,t-1}, \tag{3.8}$$

$$\mu_{i,t} = \mu_{i,t-1}, \tag{3.9}$$

$$\sigma_{i,t}^2 = \sigma_{i,t-1}^2, \tag{3.10}$$

Furthermore, when no component matches $I_{s,t}$ the component with lowest probability is replaced by a new one with $\omega_{i,s,t} = I_{s,t}$, a large $\sum_{i,t}$ and small $\omega_{i,s,t}$. After every Gaussian is updated, the weights $\omega_{i,s,t}$ of each one of them are normalized so they add up to 1.

The K Gaussian distributions are then ordered based on a fitness value given by: $\frac{\omega_{i,s,t}}{\sigma_{i,s,t}}$, meaning mixtures with high probability and low variance are deemed to have better quality. Then only the most reliable H_{mog} are chosen as part of the background model, using the criteria:

$$H_{mog} = argmin_h(\sum_{i=1}^{h} \omega_i > T_{mog}) \tag{3.11}$$

where T_{mog} is parametrizable threshold measuring the minimum amount of data that should be accounted for the background computation. Every pixel at more than λ is considered to not be part of the background, and so by exclusion, it belongs to the foreground. Furthermore, in some implementations, foreground pixels can be segmented into regions using connected a component labelling algorithm which can be useful for future object analysis and representation.

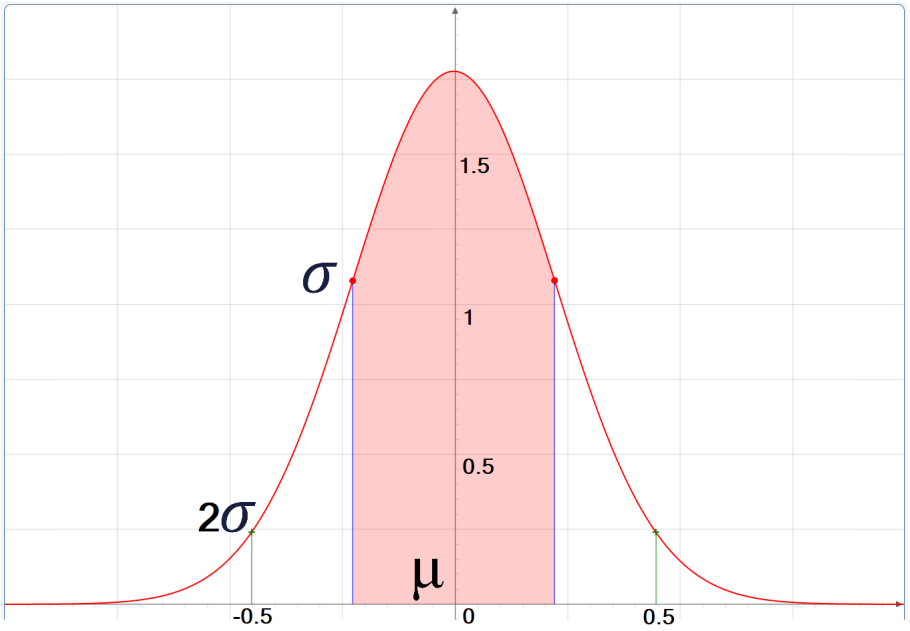


Figure 3.2: Gaussian distribution example. A Gaussian curve can be characterized by two parameters: its mean value μ, which decides where its centre and maximum point are and its variance σ which affects the value at the maximum point and smoothness of the curve. In MoG each Gaussian curve represents the probability of a given pixel having a certain colour. Only the colour intensities I, within a certain deviation (for example, the red area) are to be considered part of the background.

3.2 Optical Flow

OF is traditionally defined as the change of structured light in an image, e.g. on the retina or the camera's sensor, due to the relative motion between them and the scene (see Fig. 3.4). OF methods try to calculate and measure the relative motion, certain regions or points suffer, between two image frames taken at times t and $t + \Delta t$. These methods are called differential since they are based on local Taylor series approximations of the image signal, that is, they use partial derivatives with respect to the spatial and temporal coordinates. If we imagine that each frame in a video sequence is stacked on top of each other a spatiotemporal volume with three coordinates, two dimensional and one temporal, is obtained. This way every movement on the video sequence produces structures with certain orientations in the volume. For instance, a point performing a translation is

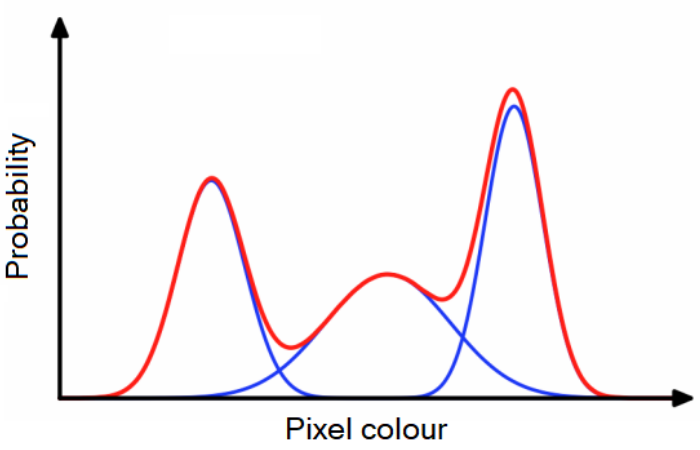


Figure 3.3: Multi-modal Gaussian example adapted from [40]. In MoG each pixel is modelled by a multi-modal Gaussian with K mixtures (Gaussian curves). In this particular figure, 3 curves (blue lines) were used. The final PDF (red line) is obtained by adding each individual mixture. Its local maximums of correspond to the most likely pixel values.

transformed into a line whose direction in the volume has a direct correspondence to its velocity vector [41].

So in a $2D + t$ geometric space, in essence $3D$, a given Voxel (x, y, t) with intensity $I(x, y, t)$ will move Δx, Δy and Δt, between two image frames. A brightness constraint can be set by:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t), \tag{3.12}$$

Supposing the movement between consecutive frames to be small:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + H.O.T. \tag{3.13}$$

where H.O.T means Higher Order Terms.

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{3.14}$$

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \tag{3.15}$$

where V_x and V_y are the x and y components of the velocity or flow of $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x, y, t). Working eq.3.15 the following is obtained:

$$I_x V_x + I_y V_y = -I_t \tag{3.16}$$

This equation has two unknown variables, V_x and V_y, and so it cannot be solved by itself. This is known as the aperture problem of optical flow algorithms [42]. The direction

vector of the movement of a contour, observed from a small opening, is ambiguous due to the fact the parallel motion component of a line cannot be inferred based on the visual input. This means that a variety of contours of different orientations moving at different speeds can cause identical responses in a motion sensitive neuron in the visual system, either an eye or a camera sensor, as illustrated in fig.3.5. OF methods introduce additional constraints, that add new equations to the mix allowing eq.3.16 to be solved.

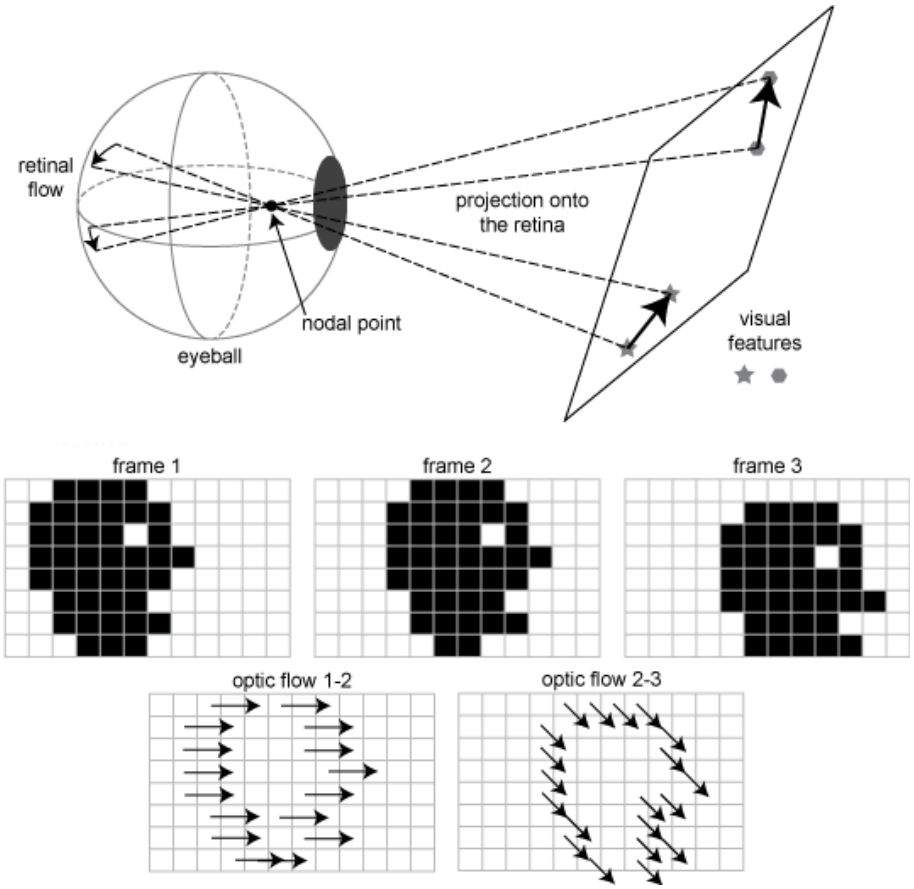


Figure 3.4: Optical Flow example adapted from [42]. Optic flow is perceived by the observer, in this case, an eye retina, as changes in the patterns of light captured by it. The image displays the positions changes of visual features (star and hexagon) on a plane and their respective displacements as perceived by the eye. If the structured light is captured for instance by a camera it is sampled spatially and temporally resulting in an image sequence. The three frames show the movement of a head. The optic flow is depicted as the correspondence of contour pixels between frames (1 and 2, 2 and 3). Optical Flows techniques search for ways to compute this correspondence for each pixel in the image.

3.2.1 Farneback Algorithm

Farneback Algorithm is a dense optical flow algorithm based on polynomial expansion. It is dense in contrast to sparse OF techniques, where only the motion of a pre-set number of trackers is computed. In a dense approach, all the regions of the frame are analysed and their motion tracked. A more in-depth look at the theoretical concepts used in the

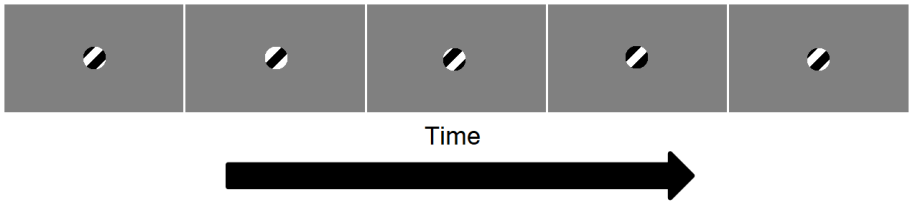


Figure 3.5: Aperture problem example retrieved from [43]. The images represent a moving sequence. It is impossible to access the direction of the movement by observing only a portion of the object.

development of this method can be done by looking at [41], and additional improvements in [44] and [45].

The algorithm uses the concept of orientation tensors to represent local orientation. Movement in an image sequence creates structures that acquire certain orientations in the 3D space. Like mentioned before a translating point is transformed into a line whose direction in the 3D space directly corresponds to its velocity. The use of tensors allows for a more powerful and easier representation of such orientations. The tensor takes the form of a 3 by 3 symmetric positive semi-definite matrix T and the quadratic form $\hat{u}^T T \hat{u}$, which is interpreted as a measure of how much the signal varies locally in the direction given by $\hat{u}$.

The orientation tensors are computed by polynomial expansion. Since an image can be seen as a two-dimensional signal, the neighbourhood of each pixel signal can be projected into a second-degree polynomial, following the model:

$$f(x) = x^T * \mathbf{A} * x + b\mathbf{T}x + c \tag{3.17}$$

where A is a symmetric matrix, b is vector and c a scalar. These parameters are computed by a weighted least squares approximation of the signal values in the neighbourhood, where the weighting function is a Gaussian.

From the model parameters, an orientation tensor is constructed by:

$$T = AA^T + \gamma bbT \tag{3.18}$$

where γ is a non-negative weight factor between the even and the odd parts of the signal.

The 2D velocity vector $(v_x v_y)^T$, can be extended to a 3D spatio-temporal directional vector v:

$$v = \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix} \tag{3.19}$$

$$\hat{v} = \frac{v}{\|v\|} \tag{3.20}$$

Instead of estimating the velocity from the tensors for each point it is assumed that the velocity field over a region can be parametrized according to some motion model:

$$v_x(x,y) = ax + by + c, \tag{3.21}$$

$$v_y(x,y) = dx + ey + f, \tag{3.22}$$

where x and y are images coordinates. This can be yet converted into a spatiotemporal vector as:

$$v = Sp, \tag{3.23}$$

$$S = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 & 0 \\ x & 0 & 0 & x & y & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.24}$$

$$p = \begin{pmatrix} a & b & c & d & e & f & 1 \end{pmatrix}^T \tag{3.25}$$

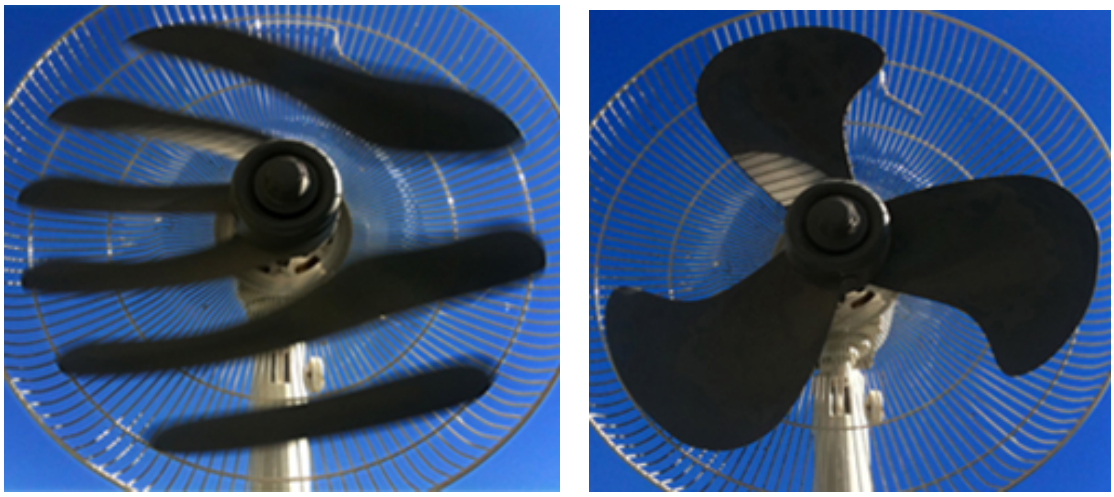
by further adding mathematical constraints and cost functions p parameters can be calculated, and therefore v_x and v_y [41].

3.3 Image capture effects

When fast rotating objects are exposed to camera sensors, certain visual phenomenona can happen. Cameras do not record footage continuously, but rather discretely by capturing series of images in quick succession, at specific intervals. Since cameras can only capture a subset of the positions of a given movement, the motion of rotation cannot be fully represented. Therefore if the rotation speed of an object is significantly faster than the capture frame rate, the object will appear to perform jumps in its rotation. Furthermore depending on the sensor type present in the camera two distinct capture modes, Charged Coupled Device (CCD) or Complementary Metal-Oxide-Semiconductor (CMOS), might cause other phenomena to happen:

- Rolling Shutter;
- Global Shutter.

A rolling shutter effect happens when cameras with a rolling shutter are used, as in most cameras with a CMOS sensor. When in a rolling shutter mode, pictures or frames from a video sequence are captured, by scanning across the scene in a short amount of time, either vertically or horizontally. This results in the adjacent rows or columns of the array of sensor cells being exposed at slightly different times, essentially creating waves that sweep through the sensor. Each row or column starts and end its exposure with a



a) Rolling Shutter b) Global Shutter

Figure 3.6: Rolling and Global shutter effects example retrieved from [46]. a) Using the rolling shutter the deformation on the fan's pads is visible producing a dragging-like effect. b) In the global shutter, the fan is like frozen in time without presenting any deformation.

slight offset in time when compared to its neighbours. Despite this during playback, the entire image of the scene is displayed at once, as if it represents a single instant in time. This produces predictable distortions of fast-moving objects or rapid flashes of light seen in Fig. 3.6a).

On the other hand, global shutter is a mode where every pixel of the image is captured simultaneously. Since there are no delays between neighbour rows, global shutter achieves an effect of a frozen image, reducing distortions as shown in Fig. 3.6b).

Yet another effect that can happen, is called the "wagon-wheel" effect. When the frequency of a wheel or a propeller's spin matches the frame rate of the camera recording it, each of the wheel's spokes or propeller blades completes a full revolution, ending up in the same position every time the camera captures a frame. Giving the sense they are motionless. However if the rotation speed changes and gets slower, the wheel or propellers will start to be a few degrees shy of their positions when compared to the last frame, and so they will seem to slowly rotate in the opposite direction.

Rotating propellers are one of the possible origin of these effects, they generate irregular and chaotic patterns when exposed to cameras, that can be useful to perform their identification in camera footage. For example, they might be used to identify a motion signature that might correspond to an UAV.

Chapter 4

Proposed model

This chapter describes the method and algorithms developed throughout the fulfilment of this dissertation. 4.1 gives a general overview of both the physical system used as basis and the developed software architecture. 4.2 goes through all the algorithms and general steps employed for the detection and identification of the UAV in imagery, obtained from the helipad's upwards camera. In 4.3 the tracking algorithm used to improve the computation of the UAV position in real time is described. To conclude 4.4 presents a possible high-level approach to control and guide the UAV on its descent, so it can perform a successful landing.

4.1 General Overview

The system is composed of two main elements, a MR-VTOL UAV and a helipad. The helipad can either be a static platform or be mounted on top of a mobile robot (UGV or USV) acting as a mobile base for the UAV, Fig. 4.1. Both UAV and the helipad are active elements capable of sensory information acquisition and data processing. Both have GPS and an Inertial Measurement Unit (IMU) used for outdoor pose estimation and are capable of communicating with each other. At the helipad's centre, there is an upwards looking camera with its optical axis perpendicular to the ground. The camera's operational characteristics such as sensor resolution, FoV and focal length are known, either by using manufacturer information or by experimental trials (i.e. calibration).

The overall software architecture is depicted in figure 4.2 and is composed of four main sub-systems:

- **Vision:** Responsible for analysing the footage coming from the helipad's camera. At each frame, it computes the estimated position and size of the UAV in the image.

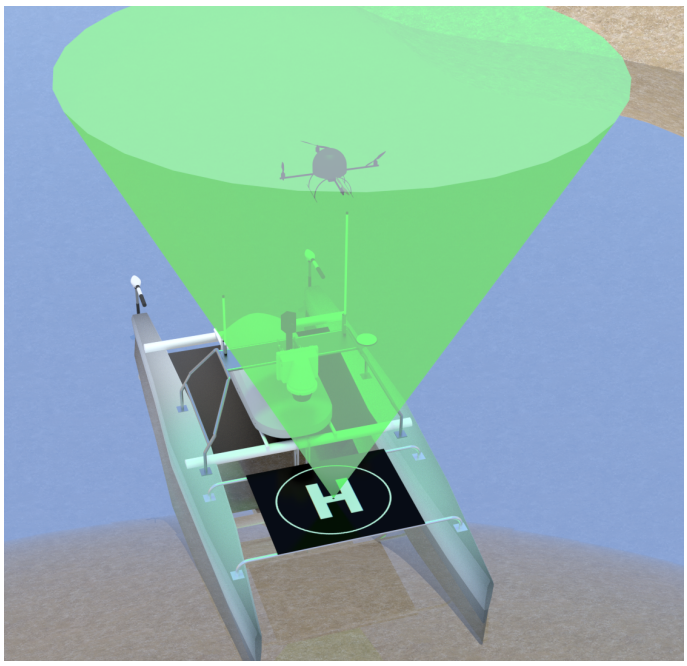


Figure 4.1: Proposed method physical system. Unlike traditional approaches, all major image processing is performed by the terrestrial/aquatic robot. When the UAV enters the helipad camera's FoV (green cone), detection and identification is performed through the analysis of motion patterns. After it is complete, commands are relayed from the terrestrial/aquatic robot to the UAV so the landing can be performed.

- **Tracker:** It receives the UAV position from the Vision sub-system and estimates its current position based on past positions and velocities. Its main goal is to minimize the noise and error coming from the vision processing modules and produce a more reliable UAV position.

- **Additional modules:** These are optional modules that may or may not exist in the system. They are composed of additional sensors such as a laser for altitude estimation, UAV's camera, etc. The data from these sources can be aggregated in a hypothetical "Robot Locator" module, with the goal of processing all the data and produce a final and more reliable estimation of the UAV's pose.

- **Controller:** It relays commands for the UAV based on the relative position between the helipad and the UAV, with the goal of performing the landing.

Considering that the proposed system is supposed to contribute to an autonomous landing, the timespan of the process should be taken into account. Therefore, the developed algorithms should be parsimonious to not introduce delays into the detection that could influence the overall robustness of the system.

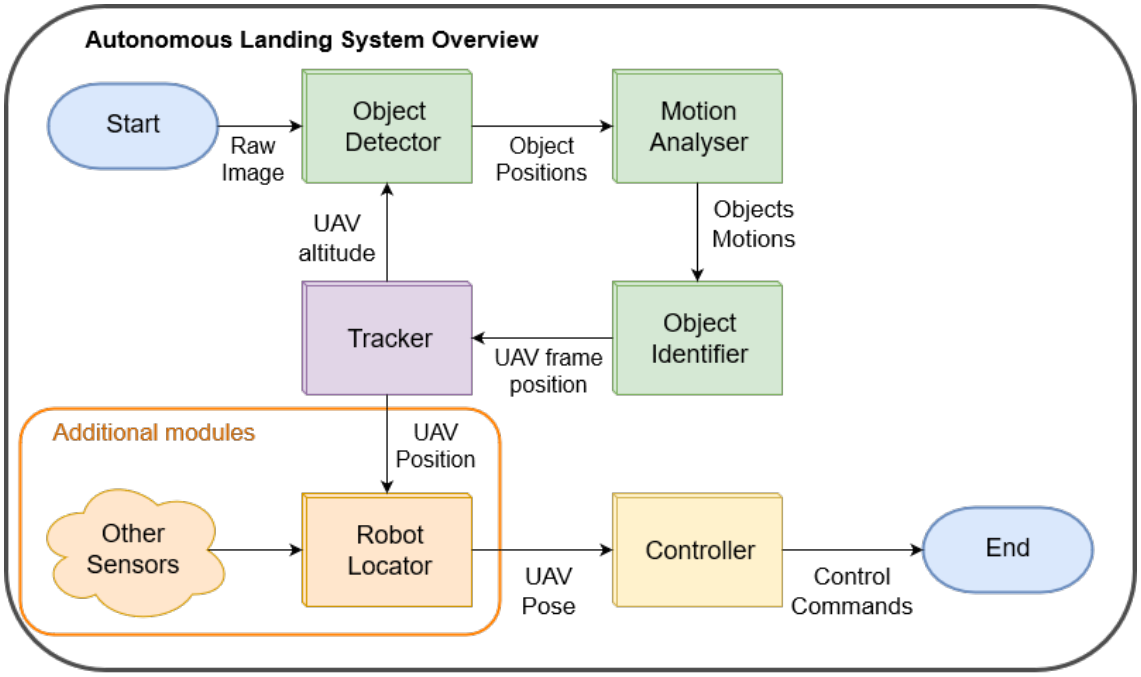


Figure 4.2: High-level architecture of the proposed system. Green rectangles represent the vision and image processing sub-system. Orange rectangles are optional parts that may, or not, exist to improve the system performance and robustness. Purple rectangles belong to the tracking sub-system and yellow ones to the controller module.

4.2 Vision

The vision sub-system is a central part of the overall system and comprises the bulk of this dissertation. Its main goal is to output the UAV position and size in each individual frame obtained from the helipad's camera. It is composed of three modules:

- **Object Detector:** Detect foreign objects in the image (i.e. all objects except the sky and clouds);
- **Motion Analyser:** Analyse the motion patterns present throughout the image and quantify the movements per pixel;
- **Object Identifier:** Take the motion analysis results and classify the objects present, deciding which, if any, corresponds to the UAV;

4.2.1 Object Detection

The detection of objects in the sky, is performed using background subtraction and clustering techniques as basis. An object is considered to be anything in the image that is not a part of the sky, clouds or the sun and its reflections.

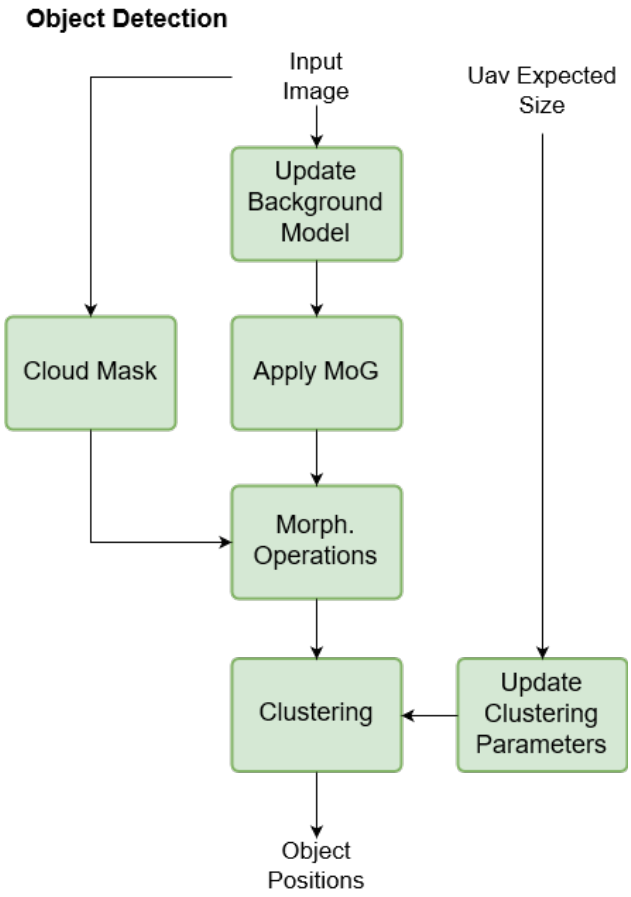


Figure 4.3: Object Detection module flowchart. The input image starts by being used to update the background model of MoG, afterwards the image is analysed and compared with the background model originating the foreground mask. Morphological operations are applied to it to remove noise while keeping the main objects in the image prominent. A cloud mask may also be used to detect noise originating from cloud movement and further remove it. A hierarchical cluster algorithm is run to merge blobs i.e., continuous white pixel zones in a binary mask, belonging to the same object that may have gotten split. The final object positions are finally computed and sent to the following modules.

4.2.1.1 Background Subtraction

Overall the sky is a fairly idle environment with the only natural objects occurring in it being clouds and sun reflections. The colour of the sky also doesn't suffer abrupt changes, having a smooth transition throughout the day depending on the sun's position on the sky. With these characteristics in mind, if we consider a relatively short timespan, short enough the sky's colour hasn't suffered a significant change and that any cloud movement is only marginal, it is possible to think of the sky as a static background. Any other objects like, planes, UAVs, etc. are then foreign objects that do not belong to it.

Using this concept the application of a BS for the detection of such foreign objects becomes the next logical step. The selected method was a MoG implementation based on [39]. Since our goal is to only detect the UAV and other moving objects like planes and

birds, the clouds should be integrated into the generated background model. Although usually slow, depending on the wind, clouds are capable of presenting significant movement that can be interpreted as a foreign object moving in the sky and therefore becoming part of the foreground mask created by MoG. To minimize this effect the system needs to be able to quickly adapt, inserting the new cloud's positions as they move, into the background model. To achieve this the number of samples h_{bs} (history) used for the learning of the background model should be low enough so that only the most recent cloud movements are considered foreground. This results in only sparse pixels (i.e. noise) in the foreground mask whose presence can be greatly reduced using morphological operations. On the other hand, T_{mog} referenced in 3.1.1, be relatively high, so components pertaining to clouds can more easily incorporated into the background model.

These parameters make the system capable of very quickly adapting to cloud's slight movements, however, their nature introduces the some undesired effects. Should the UAV stand still, or move at a very slow pace for a few moments it will too start to be added to the background model. However, the UAV's propellers produce significant movement. Due to effects explained in 3.3, the UAV propellers display irregular fast movements which prevent them from being considered a static background. In summary, even if the UAV's body is considered part of the background the propellers will not, meaning the entirety of the UAV will never be totally added to the background model. Due to the fact, the propellers are at the extremes of the UAV's body it continues to be possible to infer the its position and size. From a general standpoint it is acceptable to remove almost every cloud movement from the foreground at the expense of losing parts of the UAV. This is due to the fact, that it is easier to have our search area reduced and afterwards recover the UAV contours from a partial mask, than to distinguish between what is noise and what is not in larger areas of the image.

a) b)

Figure 4.4: Foreground mask example obtained from MoG. a) Image obtained from the helipad's camera. A UAV is present as well as some clouds on the bottom-left corner. Since the sky is fairly static, it serves as a good background model. b) The foreground mask is composed mostly by the UAV, with its propellers being the most prominent objects in it. Some cloud noise can also be seen.

4.2.1.2 Morphological Operation

Even if cloud presence in the foreground mask is minimal, some isolated pixels originating from the cloud's most recent movements can still be present. To minimize this problem, morphological operations are used. An opening (erosion followed by dilation), reduces noise without affecting the foreground objects that much, followed by further dilations to make them more prominent while at the same time merging some of the closest blobs. In this manner, a foreground mask is left with minimal, if any, cloud presence.

4.2.1.3 Cloud Mask

To improve even further the cloud removal from the foreground, a cloud mask is generated. A cloud mask is a binary representation of the pixels belonging to clouds in a given frame. Some works have been done with goal of creating such masks from sky imagery [47] [48] [49]. Due to not having a defined shape, form, or even colour, cloud identification is quite a difficult process. Clouds do however generally have, white or gray, low saturated colours. [48] deems saturation,sat, $\frac{R}{B}$ and $\frac{B-R}{B+R}$ ratios, where R and B and respectively the Red and Blue channel values of the pixel, as valid criteria for cloud identification, being on par with more complex state-of-the-art detection algorithms.

A cloud mask was obtained using the following criteria, a pixel s in the cloud mask Cl is:

$$Cl(s) = \begin{cases} 1 & if \quad sat < sat_{max},\ \frac{R}{B} < rb_{max},\ \frac{R}{B} > rb_{min} \\ 0 & otherwise, \end{cases} \tag{4.1}$$

where sat is given by:

$$sat = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta_{RGB}}{C_{max}} & , C_{max} \neq 0, \end{cases} \tag{4.2}$$

where Δ_{RGB} is the the difference between the highest c_{max} and lowest c_{min} RGB channel:

$$\Delta_{RGB} = c_{max} - c_{min}, \tag{4.3}$$

$$c_{max} = Max(\bar{R}, \bar{G}, \bar{B}), \tag{4.4}$$

$$c_{mix} = Mix(\bar{R}, \bar{G}, \bar{B}), \tag{4.5}$$

where $\bar{R}$, $\bar{G}$ and $\bar{B}$ are respectively the R, G and B channels normalized by their maximum values.

The tuning of parameters allow us to be more or less optimistic, for instance, a higher sat_{max} introduces more false positives, minimizing false negatives and a lower parameter the other way around. Every pixel simultaneously belonging to the foreground mask and the cloud mask is likely to be cloud noise and so is removed from the foreground mask.

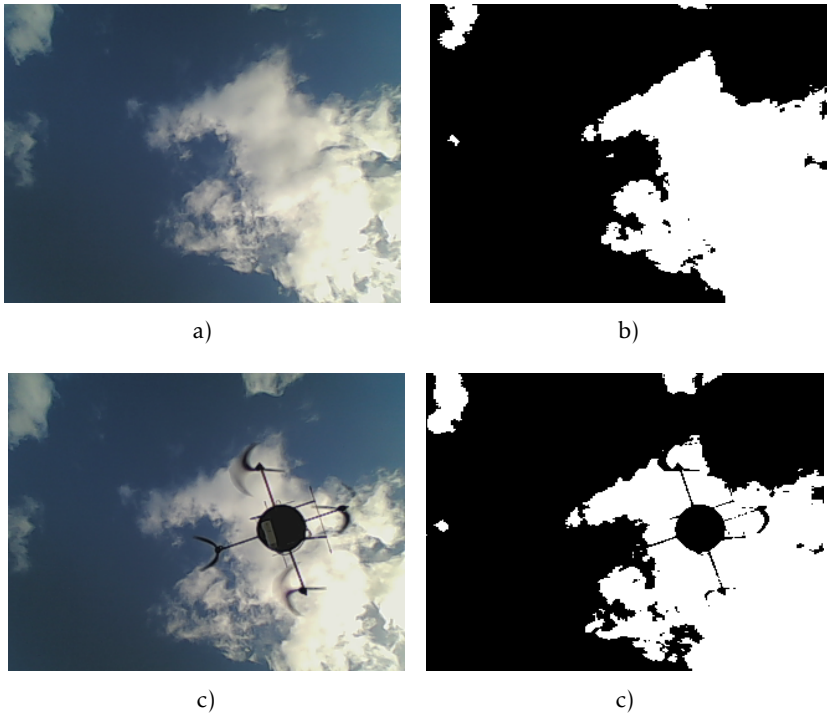


Figure 4.5: Cloud mask. b) The white pixels represent the cloud positions in the frame following eq. 4.1. As seen in d) the UAV does not belong to the mask.

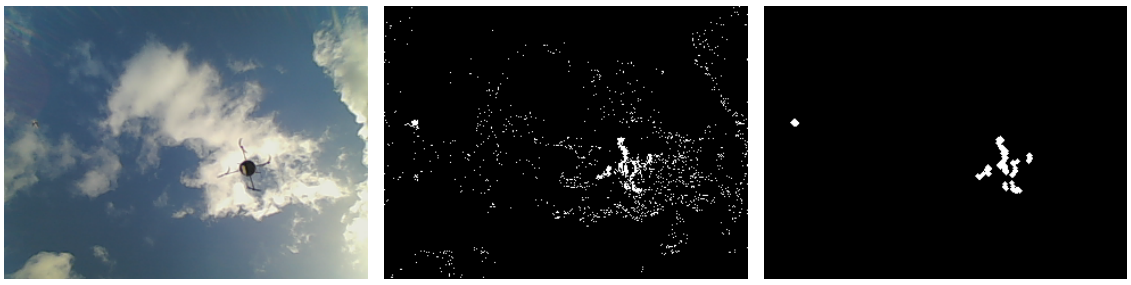


Figure 4.6: Object detection process. a) Raw image captured from the helipad's camera. The image has a considerable amount of clouds, as well as a plane passing by on its top right. b) Foreground mask obtained from the MoG algorithm. Cloud presence in the mask is heavy, however, most of it consists of isolated pixels, that represent the clouds most recent movements. c) Image after being processed by morphological and cloud removal operations. The result is a cleaner mask with only the pixels belonging to the UAV and plane remaining.

4.2.1.4 Clustering

Due to the previous operations, objects in the foreground mask might have been left disconnected, for instance, if the propellers are showing but the body only is partially part of the foreground, there is a high chance the UAV will be partitioned into several different blobs, resulting in the UAV to be considered as several different objects instead of a single one. And so, it becomes necessary to merge them in a way each set belongs to the same object. It is assumed that pixel blobs closer to each other have a higher likelihood of belonging to the same object, than blobs farther apart. A clustering algorithm based on Euclidean distances is then employed. As the number of objects in the image is unknown as well as their sizes, algorithms like K-means are too rigid due to the tendency to create clusters with similar sizes, which is not necessarily true in this particular case. Additionally, there is also the need to test various values for k_m, which increases the complexity of the system. Therefore, an agglomerative hierarchical clustering algorithm was chosen. As to simplify computation, each blob's contour, centre and radius is computed. Each one of them can then be represented by a circle with centre point $p_{c_{k(x,y)}}$, and radius r_{c_k}, i.e., its minimum enclosing circle. And so, a given cluster c_k, representing a Blob, can be seen as:

$$c_k = (p_c(x, y), r_c), \tag{4.6}$$

In a given frame we are left with a set, S_c, of clusters:

$$S_c = \{c_1, c_2, ..., c_n\}, \tag{4.7}$$

Inclusive circles are removed and overlapping circles are merged into one. The Euclidean distance, between each remaining circle centre, is then computed. The closest pair is then tested to check if the distance between them is lower than a set threshold c_{th}. If it is, a new contour formed by adding the points of both blob's contours is created. A new minimum enclosing circle is computed for the new contour and added to the set.

This can be expressed by the following equations, for a given c_1 and c_2, they should be merged if the following is verified:

$$\sqrt[2]{c1_{P(x,y)^2} - c2_{P(x,y)^2}} < c_{th}, \tag{4.8}$$

where c_{th} is a percentage of the frame size D_f.

$$c_{th} = c_{th_{min}} + \frac{UAV_{fsize}}{D_f} * c_{th_{rat}}, \tag{4.9}$$

where $c_{th_{min}}$ is a parameter representing the absolute minimum distance two clusters should be, to be considered a single cluster, in image size percentage. Meaning every cluster pair whose centres distance is inferior to $c_{th_{min}}$ will always be merged together.;D_f is the size of the frame diagonal in pixels; d_{uav} the perceived UAV size in the image; $c_{th_{rat}}$ is a linear component that represents the maximum additional percentage that can be

added to $c_{th_{min}}$; In other words when the UAV has a small size in the image c_{th} will tend to $c_{th_{min}}$, and when the UAV occupies almost the whole image, c_{th} will tend to $c_{th_{min}} + c_{th_{rat}}$, varying linearly in between. This value reflects the fact that if the UAV is at a lower altitude the object that it's being looked for is bigger in size, therefore a larger threshold between blobs should be accepted. The Tracking module is responsible to send the latest perceived UAV size, so the Object Detector module can adapt c_{th} as seen in Figs.4.2 and 4.3. If equation 4.8 is not true for any of the remaining pairs, no pixel blobs are close enough and the process finishes.

a) b) b)

Figure 4.7: Clustering algorithm results. a) Since the UAV is hovering without moving, the UAV's body is slowly added to the background model. b) Four blobs are identified corresponding to each one of the UAV's propellers (brown, green, purple and pink circles), which are the only parts of the UAV in permanent movement. c) After the clustering algorithm is run, the four blobs are merged and only one object (red circle) is identified in the image, corresponding to the UAV.

4.2.2 Object Identification

This section proposes a method for distinguishing UAVs from other objects present in the image based on motion signatures. The rationale is that a UAV presents more chaotic movements pattern than other objects (e.g. planes and birds), this is especially true thanks to the propeller movements captured by cameras. Planes are expected to have linear or close to linear movements. Birds can have slightly more irregular patterns, but generally still have fairly linear movements. More importantly, both planes and birds movements should be uniform, meaning every point on their detected objects should suffer similar translations. The same is not necessarily true for the UAV. Due to the rotating propellers, the UAV movements will not be completely uniform, for instance, the UAV may be performing a linear up to down trajectory in the image, however, the zone in the image corresponding to propellers may present movements with other orientations.

Due to the ever-changing aspect of UAV (e.g altitude variations, changes in the propellers aspect), feature extraction is particularly difficult. Resulting in trackers placed on the UAV tending to drift away with ease. For that reason a Dense OF approach was chosen, this way the flow of every individual pixel is computed without the need to create individual trackers. Instead of single points the movement patterns of entire regions can

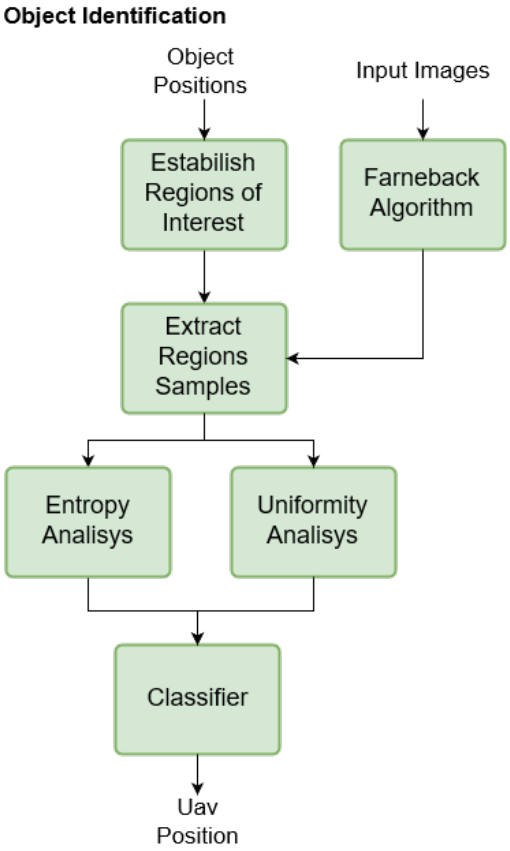


Figure 4.8: Object identification module flowchart. For each object detected by the previous modules, a region of interest is established around them. The Farneback algorithm is run for the whole image and samples for each object are selected. Each sample has its entropy and uniformity values computed. The average for each object is obtained and classified. Sending the position of the object corresponding to the UAV, if detected, to the following modules.

be extracted and analysed. The chosen method was a Farneback algorithm implementation based on [45]. The Farneback algorithm outputs a matrix with the same dimension of the analysed frame. Each element $F(x, y)$ of the matrix corresponds to a given pixel in the image and represents the displacement that given pixel suffered from the past frame $(n-1)$ to the current one n.

$$F_{(x,y)} = (\Delta x, \Delta y), \tag{4.10}$$

For every object detected in the previous step, an encompassing region of interest is established. This region should be slightly larger than the perceived object size, first to guarantee the whole object is inside the region and second to make sure the amount of data to be evaluated is sizeable i.e guarantee small objects have enough points to be analysed. To reduce computation only a sample of pixels at intervals of p_s are chosen, forming a grid throughout the region. For each region, each point in a frame n, is denominated a sample $s_{n(x,y)}$, and is given coordinates in the region. For instance, the upper left point

will be sample $s_n(0,0)$, to the left $s_n(1,0)$ and so on.

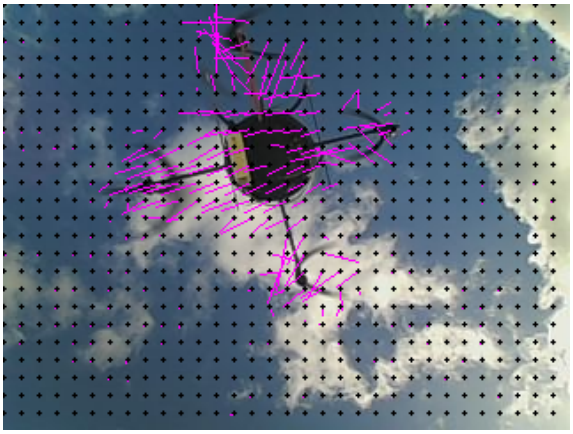


Figure 4.9: Farneback flows example. Each black dot corresponds to a sample candidate. The pink lines represent the flow of a given sample scaled up for a better visualization. The UAV has a typically chaotic flow set, especially in the propeller's area.

Two criteria are then employed to perform classification, one based on entropy measurements and another based on cosine similarity.

4.2.2.1 Entropy Criteria

For each sample, its flow relative to the past N_{of} frames is computed. Following:

$$s_{n-1}(x,y) = s_n(x,y) - F(x,y), \tag{4.11}$$

$s_{n-1}(x,y)$ is the sample pixel to be analysed in the frame $n-1$, whose position is given by a certain sample in the next frame, minus the movement $F(x,y)$ it performed in between frames, essentially forming a path. The negative signal of delta represents the fact we are essentially moving backwards in the path, as we are going from the most recent frame to the older ones.

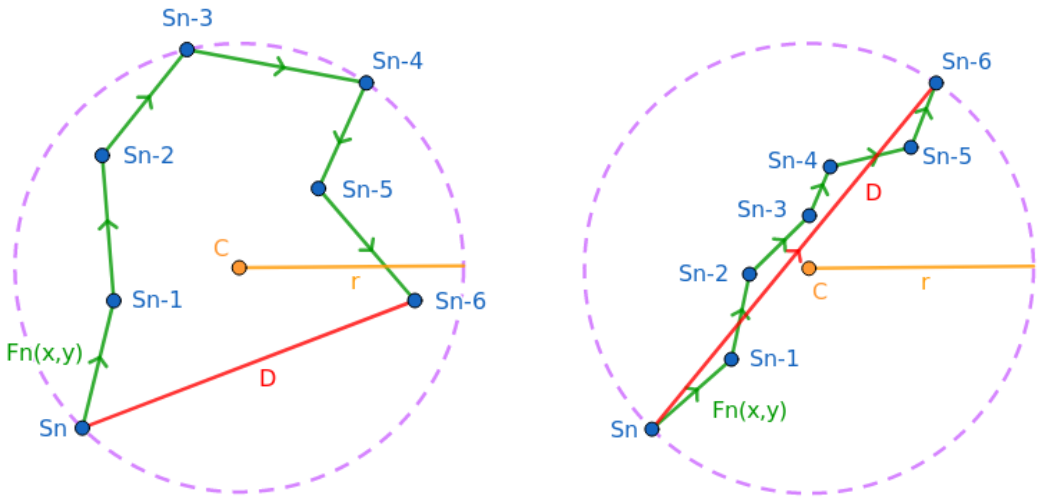
The total distance of the path $L(s)$ is given by adding each segment between samples:

$$L(s) = \sum_{n=0}^{N_{of}} s_n(x,y) - s_{n-1}(x,y), \tag{4.12}$$

To measure and characterize the detected motion an entropy metric based on the one presented in [50] was applied :

$$H(s) = \frac{log\left(\frac{L(s)}{d(s)}\right)}{log(N_{of}-1)}, \tag{4.13}$$

where N_{of} is the number of frames; $d(s)$ is the radius of the minimum enclosing circle of the trajectory; $L(s)$ is defined as the length of the sample trajectory (4.12); $H(s)$ entropy measure of a given sample.



a) High entropy

b) Low entropy

Figure 4.10: Entropy computation example for a single sample s_n. The green segments represent the flow between consecutive frames. Each flow originates a new sample s_{n-1} from where the next flow is computed. Adding their lengths we get the total length of the trajectory. The red line represents the total displacement of the trajectory. The purple circle is the minimum enclosing circle of the trajectory, the orange point its centre and the orange line is its radius.

$H(s)$ is then averaged out with every sample of the region of interest, and the average entropy of every object is computed. Considering the UAV should have a more chaotic motion pattern, this should result in higher entropy values than other objects in the image.

4.2.2.2 Cosine Similarity

While entropy measures a given path irregularity throughout time, it does not provide any spatial information i.e., the uniformity of a movement in a certain region. And so a new metric based on cosine similarity was added to the identification process. Cosine similarity is a measure of similarity between two non-zero vectors. The cosine of an angle can vary between 0 is 1, if we consider the angle formed between two vectors, starting from the same point (x,y), it is possible to establish a similarity metric. Two vectors with the exact same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1. It is thus a metric solely based on orientation and independent of magnitude.

The cosine similarity between a vector A and B can be obtained by:

$$Similarity(A,B) = \cos(\theta) = \frac{A.B}{\|A\| * \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{4.14}$$

where A_i and B_i are one of the n_c components of vectors A and B respectively.

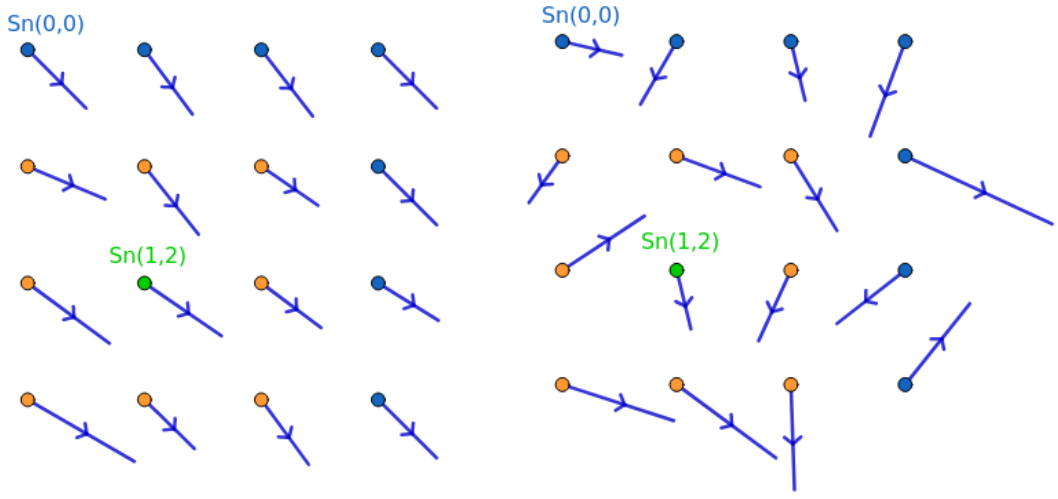
To normalize the values between $[0,1]$:

$$u = \frac{similarity(A,B)+1}{2}, \tag{4.15}$$

For each sample $s_{n(x,y)}$, its flow is compared to its eight neighbours, and a new criteria, uniformity, $u(s)$, is defined:

$$u(s) = \frac{\sum_{i=-1}^{1}\sum_{j=-1}^{1} u(s_n(x,y), s_n(x-j,y-i)}{8}, \quad if(x \neq y), \tag{4.16}$$

To improve robustness the similarity of the past frames is also taken into account and averaged out. If the flow of a given sample is too small in magnitude, it probably means its not over a portion of the object but rather over the sky or clouds, therefore if its value is considered to be too low, similarity values between it and its neighbours are discarded.



a) High Uniformity b) Low uniformity

Figure 4.11: Uniformity patterns evaluation. Objects whose flow vectors in the area have similar angles present high uniformity values, objects that do not, have lower values. The sample (green) is compared with all of its neighbours(orange). The values from the whole object are all added and averaged out, giving an object uniformity value.

Thus, with these two criteria, entropy and cosine similarity it should become possible to discriminate between the UAV and other objects present in the sky.

4.3 Tracking

This section describes the developed algorithm to estimate and track the UAV's real position based on the position of the extracted object.

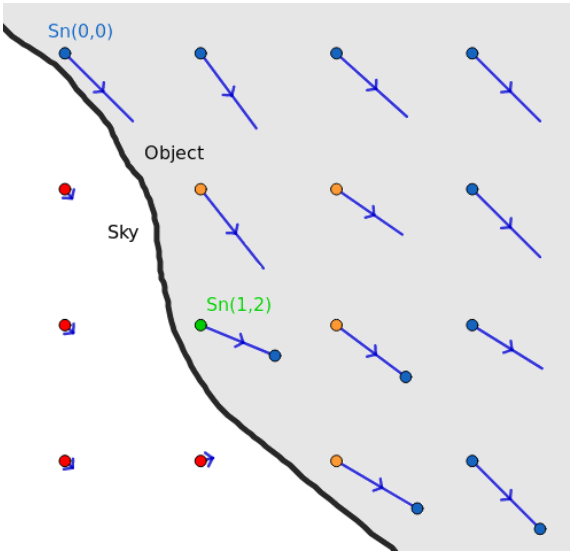


Figure 4.12: If a sample has an excessively low flow magnitude, it probably means it is not over a moving object but rather at the edge of it. Every neighbour whose flow magnitude is considered to be too low (red points), is not used for the calculation of uniformity. Only orange neighbours are, with the average being adjusted accordingly. If every neighbour is discarded the default value of 1 is given to that sample.

Figure 4.13: The result of the distinction between UAV and aeroplane. Identification performed correctly (Blue circle).

4.3.1 2D Position Estimation

The development of the tracking system was inferred around the following principles:

1. The size of the UAV in the image of an undistorted camera is directly related to its altitude;
2. UAV's position and size in the image may not vary significantly between consecutive frames;
3. The UAV can only start to appear and disappear in the image from its borders or by descending/ascending from/to very high altitudes.

These assumptions are used as support to develop a heuristic to be used in the tracking algorithm.

Since the proposed method does not rely on any mark or pattern set in the UAV to compute its centre, frame by frame analysis based on the object size can become error-prone. Due to the irregular aspect of the propellers, from frame to frame, errors in the calculus of both UAV's size and position can increase. Thus, to improve results a Kalman Filter (KF) was applied.

The state matrix X_{kf} of the filter is represented as:

$$\mathbf{X}_{kf} = \begin{bmatrix} x & y & v_x & v_y & w & h \end{bmatrix}$$

x and y represent the pixel coordinates of the UAV, v_x and v_y, the speed components in each coordinate, and h and w the width and height of the UAV in the frame, which since we're considering the UAV to have a circular shape, equal to d_{uav}.

Transition State Matrix, A_{kf}:

$$\mathbf{A}_{kf} = \begin{bmatrix} 1 & 0 & \Delta T & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta T & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

where ΔT is the time lapse between measurements, meaning the time lapse between consecutive frames.

Measurement matrix, H_{kf}:

$$\mathbf{H}_{kf} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

The covariance matrix Q_{kf}:

$$\mathbf{Q}_{kf} = \begin{bmatrix} \sigma_x & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{v_x} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{v_y} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_w & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_h \end{bmatrix}$$

To further improve results, a simple algorithm was applied on top of the filter. Since we know the UAV's position cannot suffer excessive changes between consecutive frames, it is safe to assume that the UAV position in the frame n is similar to its position in frame $n+1$ and $n-1$. As result an weighted average, where the most recent frames have greater weight than the older ones, can be used to compute the UAV current position. To reduce errors even further, the average, median and variance of the UAV positions in the past N_{avg} frames are computed to help identify and if necessary remove any outliers. With this, frames with unreliable or wrong information can be detected and discarded from use. And so an estimated UAV position, p_{avg}, in a given frame is:

$$p_{avg}(x,y) = \sum_{i=0}^{N_{avg}} p_{kf}(x,y,n-i) * w_{n-i}, \tag{4.17}$$

p_{kf} being the estimated position of the UAV, in a given frame n, by the KF and w_n the weight of frame n in the computed average. Principle number 2 can be then be expressed as follows:

$$\sqrt[2]{p_{avg}(x,y,n)^2 - p_{avg}(x,y,n-1)^2} < \Delta(x,y)_{max}, \tag{4.18}$$

where $P_{avg}(x,y,n)$ is the averaged position of the UAV in the current frame and $P_{avg}(x,y,n-1)$ the same but in relation to the previous frame. Δmax_{pos} is the value representing the maximum Euclidean distance the UAV position can change in consecutive frames, this value should vary based on principle number 1, since the closer the UAV is to the camera the bigger it will seem and a distance travelled in real life will correspond to a longer distance in the image.

Failure to comply these principles either means that there was an error in computing the UAV position in the current frame due to excessive noise, or there were some lost frames in between, requiring the tracking system to reset and start performing the UAV tracking from the a new position.

4.3.2 3D Position Estimation

To be able to perform the landing successfully, an estimate of the 3D position of the UAV relative to the helipad has to be available. With the 2D position of the UAV in the image known, it becomes possible to compute the estimated UAV position relative to the camera, and thus, the helipad in a 3D axis system [51]. A three-dimensional Cartesian coordinates

system is used. The helipad's centre is put at the origin of the referential with the z axis perpendicular to its surface, aligned with the camera's optical axis.

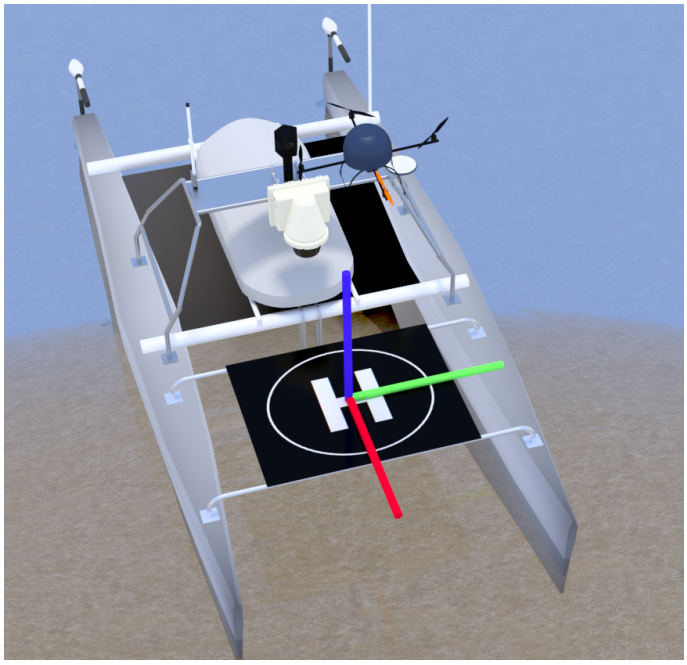


Figure 4.14: 3D coordinate frame used for UAV positioning and control commands. Its origin is at the centre of the helipad, corresponding to the coordinate $(0,0,0)$. The X, Y and Z axis are respectively the red, green and blue lines of the referential frame on the image.

Knowing the camera's focal length in an ideal without accounting any error caused by distortions:

$$z_{uav} = \frac{D_{uav} * f}{d_{uav}}, \tag{4.19}$$

where z_{uav} is the distance from the object to the camera(m);f the camera focal length(m); D_{uav} the object real dimensions (m); d_{uav} the object size in the image (pixels);

Since the camera is facing upwards the UAV will be projected in the image plane from fig. 4.15 parallel to the camera optical axis, and the distance to the camera is the altitude the UAV is from the landing pad.

From Z it is possible to estimate the X and Y relative coordinates using:

$$x_{uav} = dx * \frac{z_{uav}}{f}, \tag{4.20}$$

where x_{uav} is the the object estimated position in the X axis (m) and dx the object distance from image centre (pixels). The same process can be performed for the computation of y_{uav}. This way, the final UAV position, $P_{uav} = (x_{uav}, y_{uav}, z_{uav})$ can be computed.

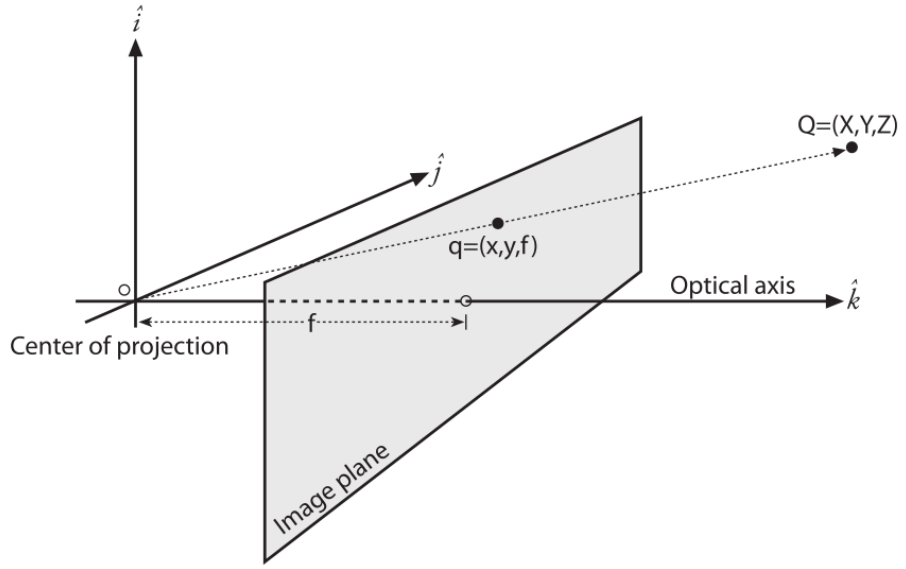


Figure 4.15: Object 3D position computation from the camera image. A given point Q = (X, Y, Z) is projected onto the image plane by the ray passing through the centre of projection, thus resulting in a point on the image, q = (z, y, f). Knowing the camera's intrinsic f, and at least the distance between two points q, in pixels, and the real-life distance between their correspondent projections Q becomes possible to compute an estimated Z by computing their ratio 4.20.

4.4 Controller

This section presents the high-level control system used to perform the autonomous landing. The goal of the system is to safely guide the UAV from an airborne position to the surface of the landing pad. In this particular case, we're only concerned about the high-level control, meaning which direction should the UAV take when it is at a given 3D P_{uav} coordinate relative to the helipad's position. Low-level control, i.e which effort should be put into each of the motors attached to the propellers is not within the scope of this dissertation. Therefore it will be assumed a low-level controller is already at place, such as a PID controller, to guarantee the UAV performs the high-level commands it is issued.

The direction and intensity of the commands should adapt to the relative positions between the UAV and helipad, issuing smoother commands the closer the UAV is to the helipad's centre, and the lower it is. As referenced in 4.3.2, the helipad is given the $(0, 0, 0)$ coordinate which is the final goal.

An approach zone is established on top of the helipad's surface. This zone represents the area where the UAV can manoeuvre in safety without endangering surrounding objects or people, while at the same time maintaining needed alignment with the helipad. It also was assumed that the airspace above the helipad is free of any obstacles. When

the UAV is inside the approach zone it starts or resumes the landing, with direction commands decreasing its altitude, and making it closer to the goal. On the other hand, when the UAV is outside the zone, the goal is to get inside of it, so the landing manoeuvre can start.

The approach zone is modelled using a multiplicative inverse function, as basis Fig.4.16 generically given by:

$$y = \frac{1}{x}, \tag{4.21}$$

from which it is possible to get:

$$y = \frac{1}{|x|}, \tag{4.22}$$

and:

$$y = -\frac{1}{|x|}, \tag{4.23}$$

finally adding a factor of scale a_{az} and an offset b_{az}, so it better fits our needs we get the approach zone $az(x)$.

$$az(x) = -\frac{a_{az}}{x} - b_{az}, \tag{4.24}$$

This function was chosen due to its strong resemblance to a funnel as seen in Fig.4.17, which guides the UAV from a large area at a high altitude, to the centre of the helipad in fairly smooth and progressive way.

Considering an YoZ or XoZ plane, where Z and Y/X 3D coordinates correspond to the function's y and x 2D coordinates respectively. The helipad is considered to be perpendicular to that plane with its centre at the $(0,0)$ position.

The UAV is considered to be inside the zone if it is above the curve:

$$z_{uav} > az(d_h), \tag{4.25}$$

z_{uav} corresponds to the UAV's altitude and d_h is the distance between the UAV and the helipad in XoY plane parallel to the helipad given by:

$$d_h = \sqrt{x_{uav}^2 + y_{uav}^2}, \tag{4.26}$$

For each P_{uav}, a velocity command is applied. The velocity command, v_c, can be represented as a vector with following parameters:

$$v_c = (x,y,z)(\phi_{v_c}, \theta_{v_c}, \psi_{v_c}, m_{v_c}), \tag{4.27}$$

where (x,y,z) represents its origin point,ϕ_{v_c}, θ_{v_c}, ψ_{v_c}, roll, pitch and yaw, respectively, in relation to the 3D coordinates frame and m_{v_c}, $[0,1]$ is its magnitude i.e the intensity of the command. Since we are controlling the UAV the origin of the vector will always correspond the UAV's current position. Additionally, roll has no impact whatsoever on the velocity vector direction and therefore is always set to 0, leaving only θ_{v_c} and ψ_{v_c} left to compute. Since the goal is to keep closing the gap between the UAV ant the $0,0,0$

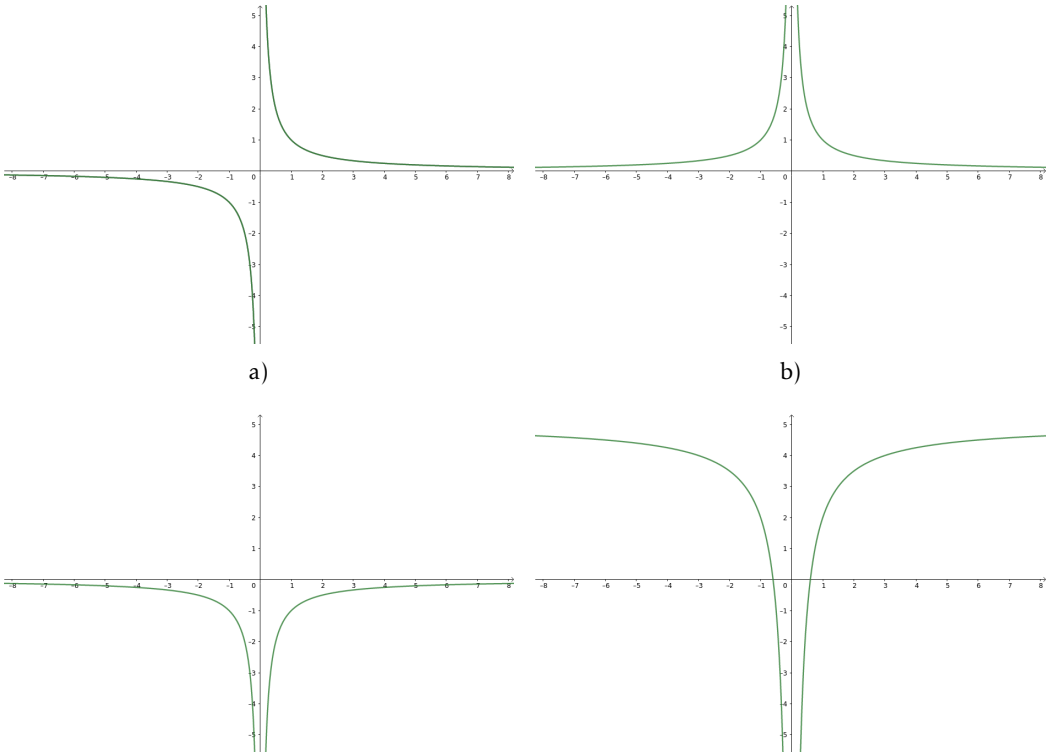


Figure 4.16: Multiplicative Inverse function transforms. a) Generic multiplicative inverse. b)The function with an always positive denominator creates a funnel effect. c) Multiplying it by -1 turns it upside down. d) Function scaled to have a wider "neck"and added an offset to start from positive values.

coordinate, the command vector should always point in its direction. And so ψ_{v_c}, can be obtained via trigonometry:

$$\psi_{v_c} = \begin{cases} \pi + \arctan(\frac{y}{x}), & for \;\; x \geq 0, \;\; y > 0 \\ \arctan(\frac{y}{|x|}), & for \;\; x < 0, \;\; y > 0 \\ \arctan(\frac{y}{x}), & for \;\; x \leq 0, \;\; y < 0 \\ \pi + \arctan(\frac{|y|}{x}), & for \;\; x > 0, \;\; y < 0 \\ \pi, & for \;\; x > 0, \;\; y = 0 \\ 0, & for \;\; x > 0, \;\; y = 0 \end{cases} \tag{4.28}$$

The pitch represented by θ_{v_c} represents how much we descend compared to the horizontal movement, varying in between $[0, \frac{\pi}{2}]$. The UAV should descend at a sharper angle, the closer it is to the helipad. To achieve this effect the derivative of 4.24 at z_{uav}. To add the effect of increased sharpness the closer it is to $(0,0)$ position of the helipad, the distance between the UAV's position and the approach zone width at that z_{uav} is factored in. And so θ_{v_c} can be obtained:

$$\theta_{v_c} = \frac{\pi}{2} - \arctan\left(\frac{dy}{dx} az(x)\right) * k_{mc} * d_{az}, \tag{4.29}$$

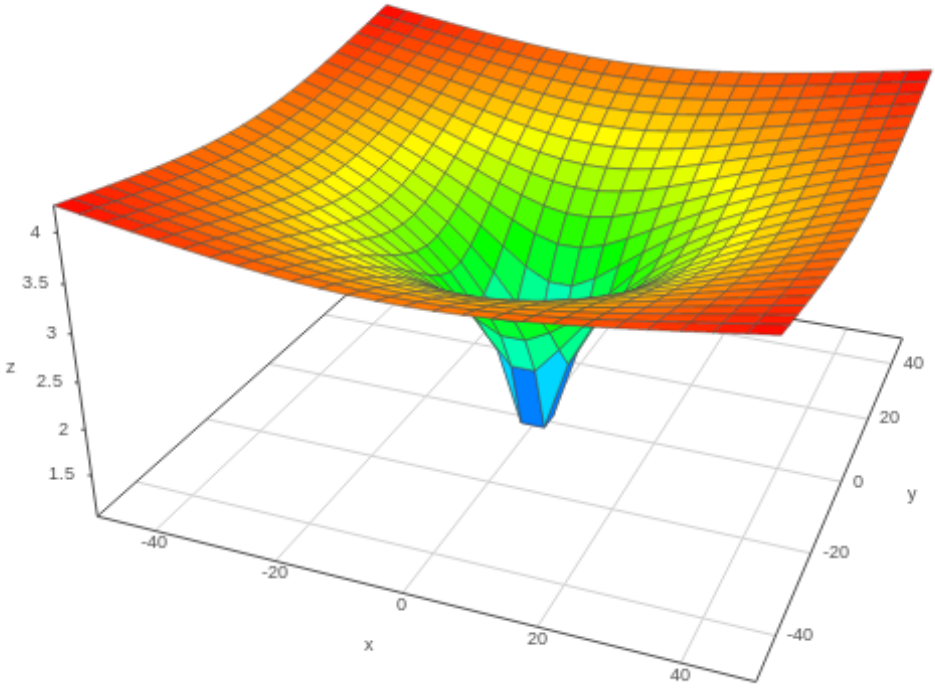


Figure 4.17: 3D perspective of the approach zone. When the UAV is above the surface is considered to be "inside" the zone when below it is "outside". For each point within the approach zone a certain v_c is to be issued to the UAV. The orientation of v_c is computed based on the derivative of the surface at a given P_{uav}, eq. 4.29. The colours represent the intensity of v_c, m_{v_c}. Warmer colours like red, mean a greater magnitude and cooler colours like blue, a smaller one.

where k_{v_c} is a normalizing constant to guarantee the command vector is slightly at a less shaper angle than the function derivative allowing the command velocity a quicker convergence; d_{az} is the ratio between the width of the approach zone at a given z_{uav} and d_h.

$$d_{az} = \frac{d_h}{az^{-1}(z_{uav})}, \tag{4.30}$$

where $az^{-1}(z_{uav})$ is inverse function of $az(z_{uav})$. This way for each z_{uav} the approach zone has certain radius given by:

$$az^{-1}(z_{uav}) = \frac{a_{az}}{z_{uav} - b_{az}}, \tag{4.31}$$

The magnitude ,m_{v_c}, of the velocity command, v_c, varies depending on the altitude z_{uav}. This variation is modelled by a Gompertz curve Fig. 4.22, given by:

$$g(z_{uav}) = \alpha_g * e^{-e^{-\gamma_g * (z_{uav} - \beta_g)}} \tag{4.32}$$

where α_g, γ_g, β_g are all adjustable parameters, to make the curve sharper or softer.

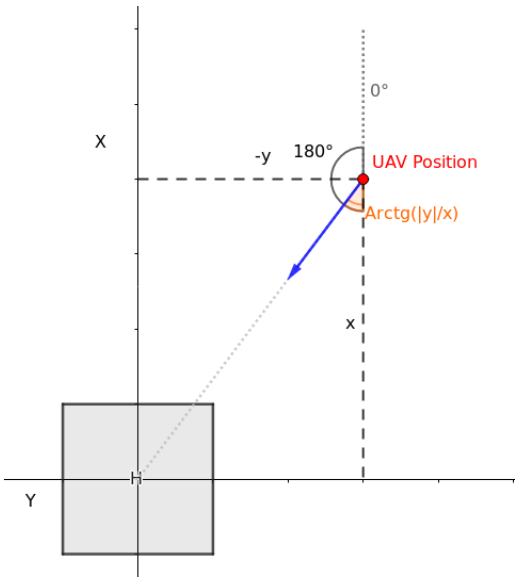


Figure 4.18: ψ_{v_c} computation. a) The grey square at the origin represents the helipad viewed from above. The red dot corresponds to the UAV position in the XoY plane. The dashed line next to 0° represents the orientation of the X-axis where ψ_{v_c} is equal to 0. In this quadrant $(+,-)$ the desired ψ_{v_c} is given by subtracting the arctan($\frac{|y|}{x}$ to $\pi(180\,\text{deg})$. The blue arrow represents the orientation of the velocity command at the current UAV's position.

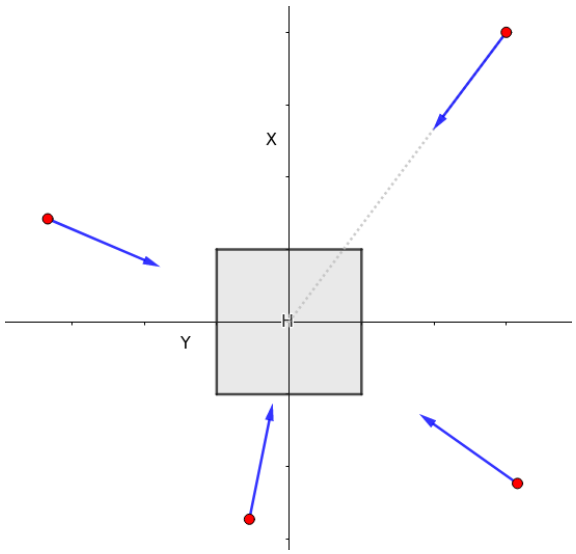


Figure 4.19: Another possible ψ_{v_c} orientations in the different quadrants.

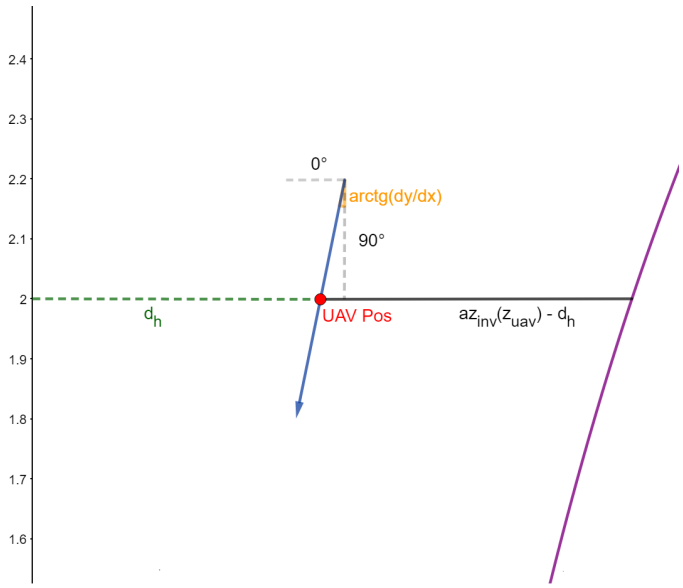


Figure 4.20: Computation of θ_{v_c} for a given UAV position. For a given UAV position (red dot) a θ_{v_c} is computed following 4.29. The green dashed line corresponds to d_h and the purple line the approach zone surface.

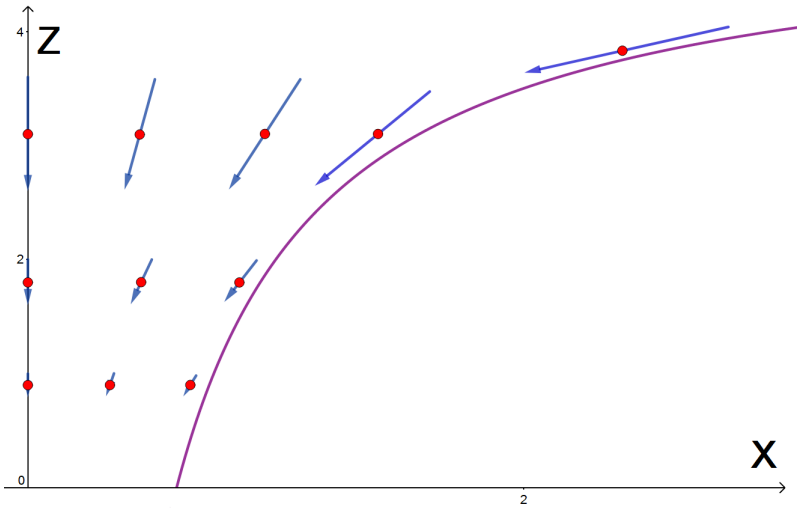


Figure 4.21: A sample of the velocity commands vector field. The lower the UAV position the smaller the magnitude of the movement should be. The closer the UAV is to the helipad centre in XoY plane the closer the angle of θ_{v_c} should be to 90°.

$$m_{v_c} = g(z_{uav}), \tag{4.33}$$

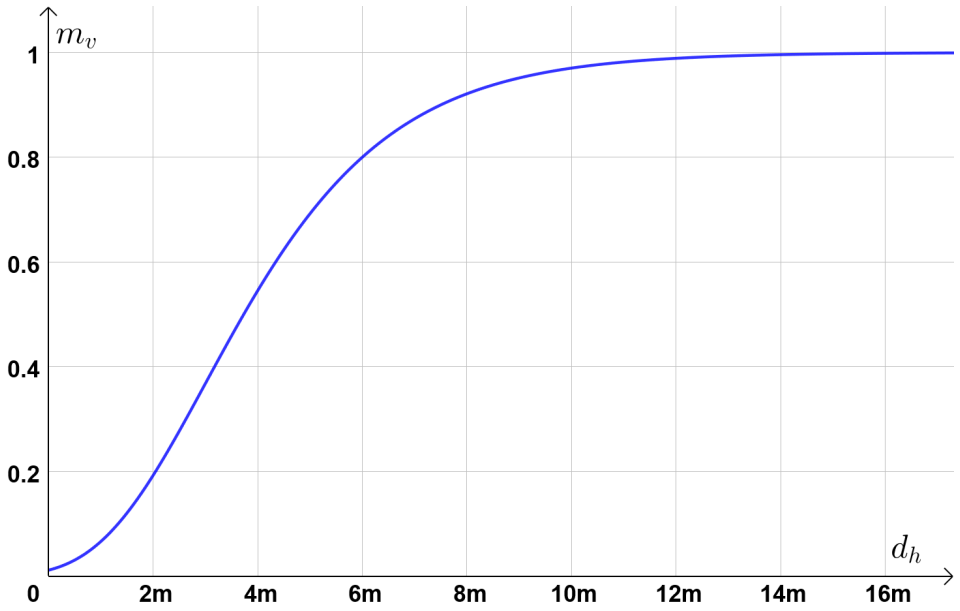


Figure 4.22: Gompertz function. For each z_{uav} (x-axis) a velocity command magnitude between 0 and 1 is computed. At high altitudes, the command is sharper, smoothly decreasing the lower z_{uav} gets.

Chapter 5

Experimental Results

This section goes over the experimental results obtained using the proposed method.

5.1 Experimental Setup

The proposed method was entirely implemented in the C++ programming language and it was made fully compliant with the Robot Operating System (ROS). The system was tested in an Intel Core i5-4200U quad-core running at 1.60GHz, with 6 Gb of RAM. The used OS was the 64-bit Linux distribution Ubuntu 16.04 (Xenial Xerus) and Open Computer Vision (OpenCV) 3.2 was used for all low-level computer vision routines.

In order to measure the performance of the vision sub-system, an extensive data-set of 23 colour videos, encompassing a total of 44557 analysed frames with a resolution of 320×240, was used. The footage was obtained from a static camera placed at ground level, with its optical axis pointing towards the sky. The dataset includes both clear and cloudy sky videos obtained at different times of the day. The control module was tested individually in a simulated environment where some simple landing scenarios were run. For a better analysis, the output of each module was evaluated independently with appropriate success metrics applied for each one of them.

5.2 Model Parametrisation

This section presents the values given to each parameter during the experimental trials, as well as the thought process behind the choices made.

The parameter h_{bs} was set to 8, low enough to minimize cloud presence in the foreground mask, Fig.5.1. Complete cloud removal from the foreground mask by the BS algorithm was shown to be impossible, especially in footage with heavy cloud presence,

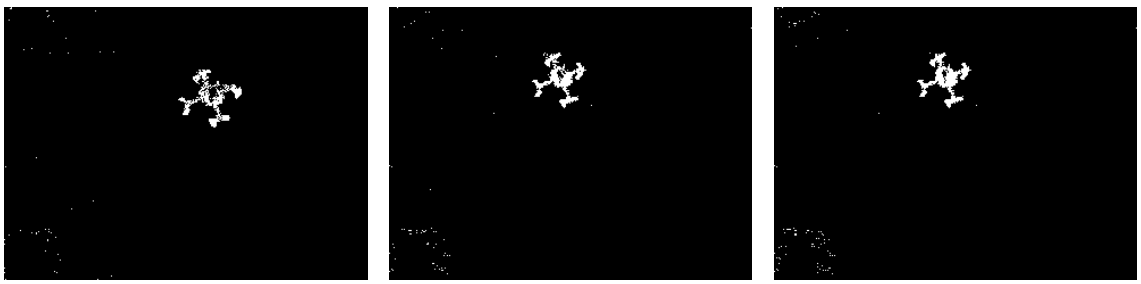
Table 5.1: MoG and cloud mask parametrisation

Parameter	Value
h_{bg}	6
T_{mog}	0,7
λ	3.0
sat_{max}	0,25
rb_{max}	1,15
rb_{min}	0,85

Table 5.2: Clustering Parametrisation

Parameter	Value
$c_{th_{min}}$	0,15
$c_{th_{rat}}$	0,25
p_s	8

however, with good enough parameters it becomes possible through additional image processing to remove most of it. T_{mog} has a similar effect to h_{bs} but the other way around, the lower it is the stronger cloud presence will be and vice versa, and so was set to 0.7. λ was set as value slightly higher than the default value of the algorithm (2.5), at 3.0, so clouds are more easily incorporated into the background. sat_{max}, rb_{min} and rb_{max} cloud mask parameters were set to $0,25$, $0,85$ and $1,15$ respectively, which are tendentiously pessimistic values so false positives are minimized.



a) b) c)

Figure 5.1: Various h_{bs} values. The higher the value the more significant cloud presence is in the foreground mask.

Parameter $c_{th_{min}}$ was chosen to be 0.15, which represents 15 % of the frame size. It corresponds approximately to the minimum size the UAV should have to start to be effectively detected and identified. $c_{th_{rat}}$ was set as 0.35, this way the $c_{th_{min}} + c_{th_{rat}}$ at its maximum equals to 0.55 which means that when the UAV is filling most of the image, objects at the distance of 55 % of the image should merged.

Variables m_i and b_i represent the line equation used to classify objects in the image and were experimentally obtained through data analysis as will be seen in section 5.3.

Table 5.3: Classifier Parametrisation

Parameter	Value
N_{of}	10
m_i	-0,03333
b_i	0,98

Table 5.4: Tracking Parametrisation

Parameter	Value
Δ_{max}	0,35
N_{avg}	5/8

Table 5.5: Frame weights user for the computation of the UAV average position

w_n	n	n - 1	n - 2	n - 3	n - 4	n - 5	n - 6	n - 7
$N_{avg} = 8$	0.175	0.15	0.13	0.12	0.125	0.125	0.10	0.075
$N_{avg} = 5$	0.3	0.25	0.15	0.10	0	0	0	0

The number of frames used for the entropy and uniformity calculation was selected to be N_{of}=10, the idea is to the system to have some memory (i.e., the movements made in recent past are taken into account) without being too slow reacting to the most recent ones. Δmax_{pos} is the maximum distance the UAV is expected to move in between consecutive frames, since that depends on its proximity to the camera, the closer the UAV, the more pixels the same distance represents in the image, the chosen value was a percentage of the UAV size in the image in the recent past. Imagining an UAV with 1 metre diameter moving at about $5m/s$, if we consider that between each frame there is a 0.066 seconds delay (15 fps), the UAV would move 0.33 metres, which correspond to 33.3% of its size. Rounding up, Δmax_{pos} was chosen to be 0.35. The interval between samples for optical flow analysis was set as p_s=8. It is low enough so optical flow has significant data to work with while not overbearing computational requirements. Furthermore, objects smaller than 8 pixels do not warrant analysis. The number of frames N_{avg} used for the averaged position of the tracking algorithm was 5 and 8, with the algorithm being run with both values and its results evaluated.

The $az(x)$ parameters, a_{az} and b_{az} were set to 3 and 5 respectively making the $az(x)$ "neck" start to get wider at around the 4 metres of altitude which was considered to be good value. Finally the Gompertz curve parameters α_g was set to 1 so the maximum value of the curve corresponds to it. β_g is 3, which means the curve increased slope will start around that value, which coincides with the widening of $az(x)$. γ_g is 0.5, so the curve increases smoothly until around the 10 metres of altitude.

A summary of parameters deemed best to be used in the experimental trials are shown in tables 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6.

Table 5.6: Control Parametrisation

Parameter	Value
a_{az}	3
b_{az}	5
α_g	1
γ_g	0.5
β_g	3
k_{v_c}	1.07

5.3 Vision System

The goal of the vision system is to analyse the imagery obtained from the helipad's camera, and correctly detect and identify the UAV in every frame and computing its three dimensional position relative to the camera.

5.3.1 Object Detection

The Object Detection module provides a binary mask of which pixels belong to foreign objects (i.e. white pixels). The output of the module was qualitatively compared with the expected results in the 23 videos from the dataset totalling 44557 frames, Fig. 5.2. Depending on its contents, each frame was categorised into four different classifications:

- **Correct (C):** The UAV and other foreign objects such as planes, are correctly identified and have their expected sizes (with some margin of error) in the foreground mask.
- **False Positives (FP):** Some parts of the frame that should belong to the background model are labelled as foreground. Generally this is due to cloud parts that were not removed during the image processing stage or due to abrupt luminosity changes captured by the camera's sensor.
- **False Negatives (FN):** The UAV is not totally detected in the foreground mask and there is not sufficient information to correctly infer its correct size and position in the frame.
- **Incorrect (I)** Frames where there are large amounts of False Positives and False Negatives, rendering any frame analysis inefficient.

As shown in table 5.7 the module proved to be very efficient with 93,44% of the analysed frames providing the expected output, producing reliable information so that the UAV position can be accurately computed by the following modules. The second most common occurrence was False Negatives, in 5,93% of frames followed by False Positives with 0,48%. Only in 0.015% of the analysed frames no information whatsoever could be retrieved which can be considered a positive result.

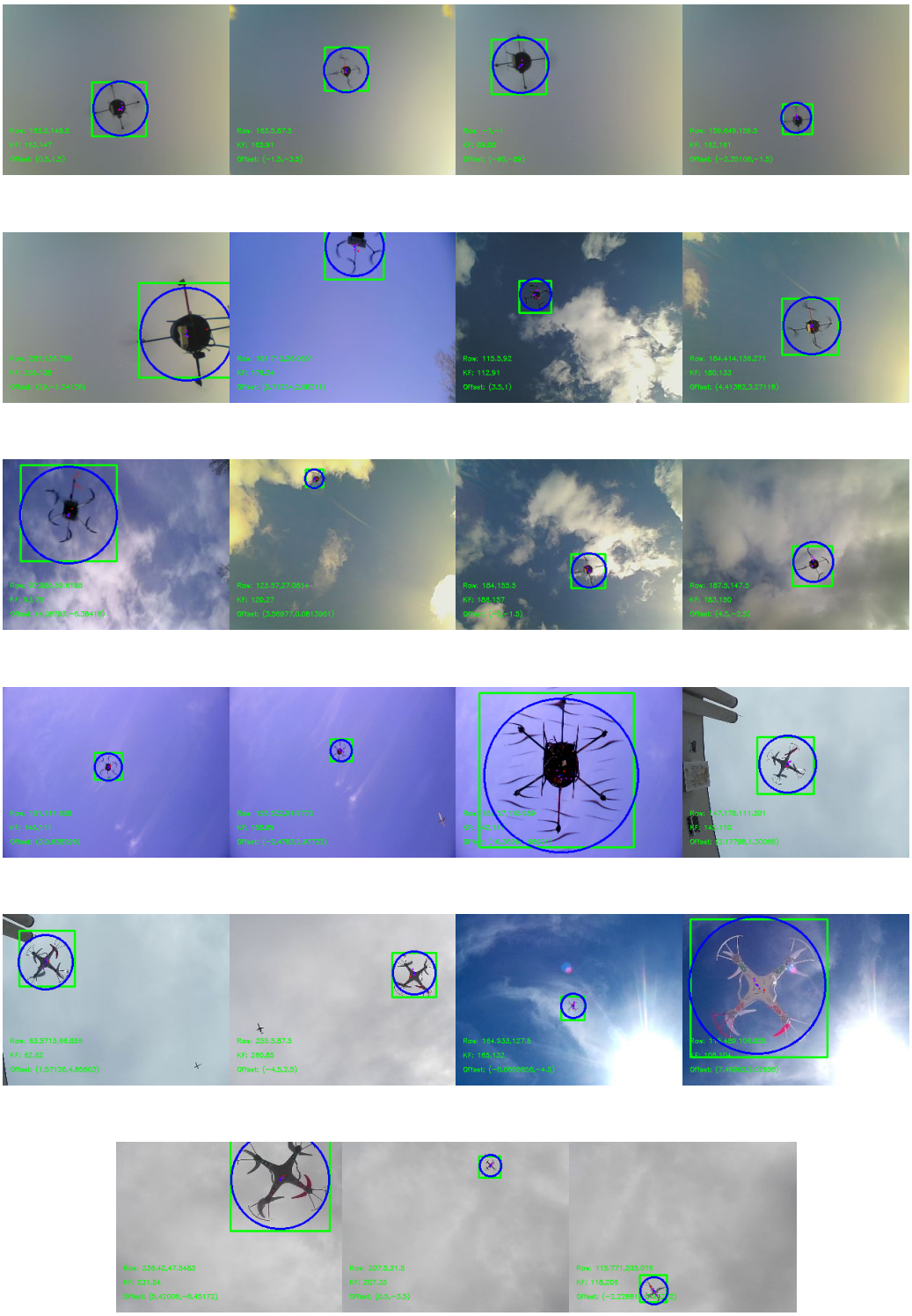


Figure 5.2: Data-set representative frames. Each image corresponds to a different video from the data-set, with numeric IDs, starting from 0, being assigned from the top-left. The set is diverse with with cloudy and clear environments where were used three distinct UAVs with different colours, size and morphologies (Hexa-copter and quad-copter). The green square represents the output of the KF tracker. The blue circle represents the average position of the UAV in last $N_{avg} = 5$ frames.

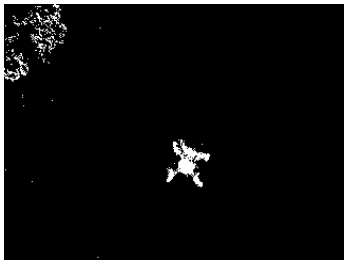
Table 5.7: Object Detection Results

Classification			
C(%)	FP(%)	FN(%)	I(%)
93,44	0,48	5,93	0,15

a) Correct frame

b) Correct frame mask

c) False Positive frame



d) False Positive mask

e) False Negative frame

f) False Negative mask

Figure 5.3: Object Detection frame classification. a) and b) frames where the correct identification was performed. c) and d) due to luminosity changes and significant cloud movement, some clouds appear on the mask (top-right corner), e) and f) The UAV is hovering at a high altitude and so only a few sparse pixels show on the mask.

5.3.1.1 Discussion

FP cases tend to occur during sudden luminosity changes in the image, caused by variations in the camera's exposure due to the sun movements. Since most of the image suffers a change in the intensity of its colours, MoG needs a certain number of frames according to parameter h_{bg}, to adapt to the new luminosities values as shown in Fig. 5.4. Given enough frames to adapt, the algorithm showed it was capable of recovering and work correctly again.

Other FPs are caused by clouds moving at significant speed. Generally, this effect is punctual and only generates some sparse lone blobs in the mask, which proved to be not problematic since the Object Identification module should be able to discard blobs originated by clouds.

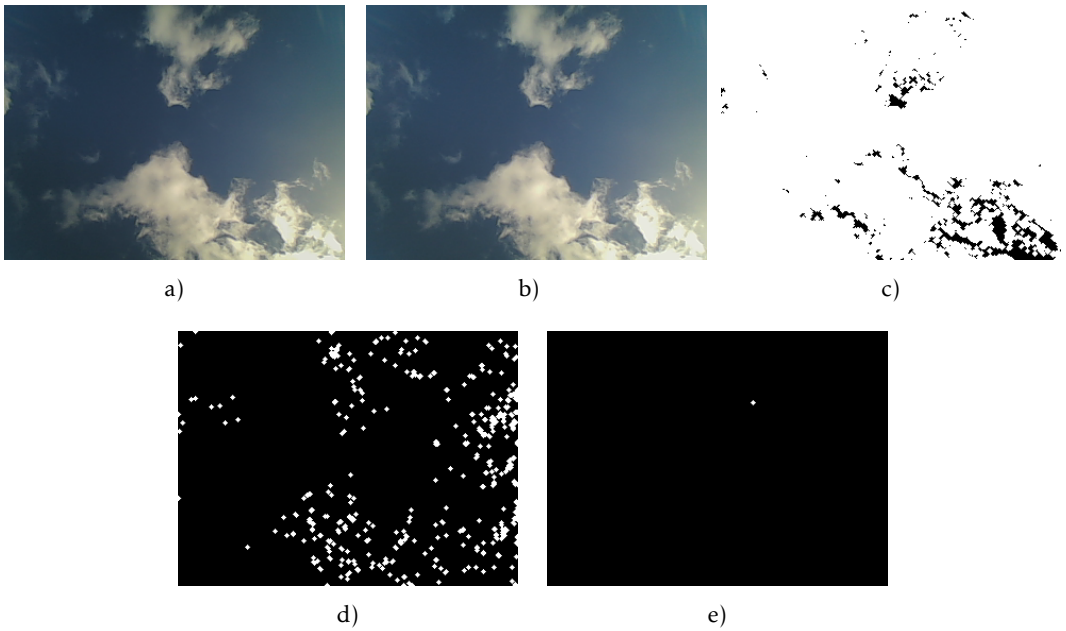
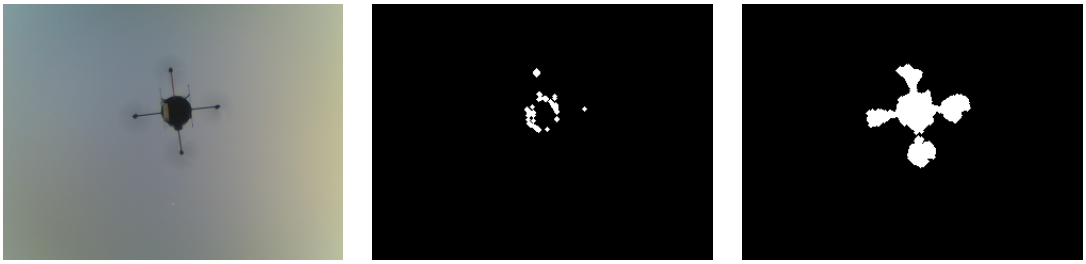


Figure 5.4: Effect of sudden luminosity variations. From frame a) to b) there is a slight, yet sudden luminosity change on the image. The foreground mask, c), becomes almost completely white since most of the colours in the image will not fit into the previous background model. Due to the short h_{bg} value, after a single frame, the algorithm will start to adapt. e) After only 5 frames the algorithm is working correctly again since the new background model will be completely updated.

FN cases mostly occur due to the UAV hovering, with minimal translation movement, at very high altitudes which makes propeller's movement less noticeable. Certain luminosity conditions like the one shown in Fig. 5.5 also show to be problematic. The propeller's movement can become very tenuous, leading to the UAV being integrated in the background model. To solve this, the h_{bs} can be increased and T_{mog} decreased so the mask becomes more sensitive to slight movements. This might induce the emergence of more false positive cases, especially in cloudy environments, but is usually a trade-off

worth making. Since the problem only occurs in a very specific situation (e.g., the UAV hovering without moving at high altitudes), the Tracker module is capable to keep track of the UAV's position until its movements become noticeable again. If a certain amount of time goes by, the UAV should be considered to have left the frame.



a) b) c)

Figure 5.5: Effect of bad illumination. a) Bad illumination conditions coupled with poor image quality, makes the propeller's movement very subtle and hard to detect in b). c) Increasing h_{bg} to 15 improves significantly the quality of the mask.

Yet another problematic situation is when the sun is high in the sky, close to the centre of the image as seen in Fig. 5.6. The sun and its glares on the camera lens cause a saturation effect, i.e. pixels turn almost completely white in that region, which can make the detection of the UAV when it is flying over that zone of the image considerably harder or downright impossible. There is very little that can be done in terms of image processing to solve this problem, however, adjustments in the hardware used can help minimize it. For instance, the adjustment of the camera's aperture among other settings can be used to decrease the effect and even the use a luminosity sensor to perform the adjustments automatically could be an interesting solution. In a similar sense sun glares create parts of the image with low colour saturation, which end up be considered as clouds in the cloud mask lowering its overall quality, Fig. 5.7. For these reasons the cloud mask must be fairly pessimistic in its parametrization to minimize this problem.

5.3.2 Object Identification

To test the object identification algorithm, a sample of the entropy and uniformity values of various objects detected along the dataset videos were registered. A training set was established and analysed as depicted in Fig. 5.8. Aeroplanes seem to always have high uniformity values (i.e. very close to 1). On the other hand despite generally having entropy values below 0.1, there are some samples with higher values, with the maximum observed being 0.28. UAVs have a more disperse data set, with entropy values varying between 0.13 and 0.3 and uniformity values reaching as high as 0.97 and as low as 0.48.

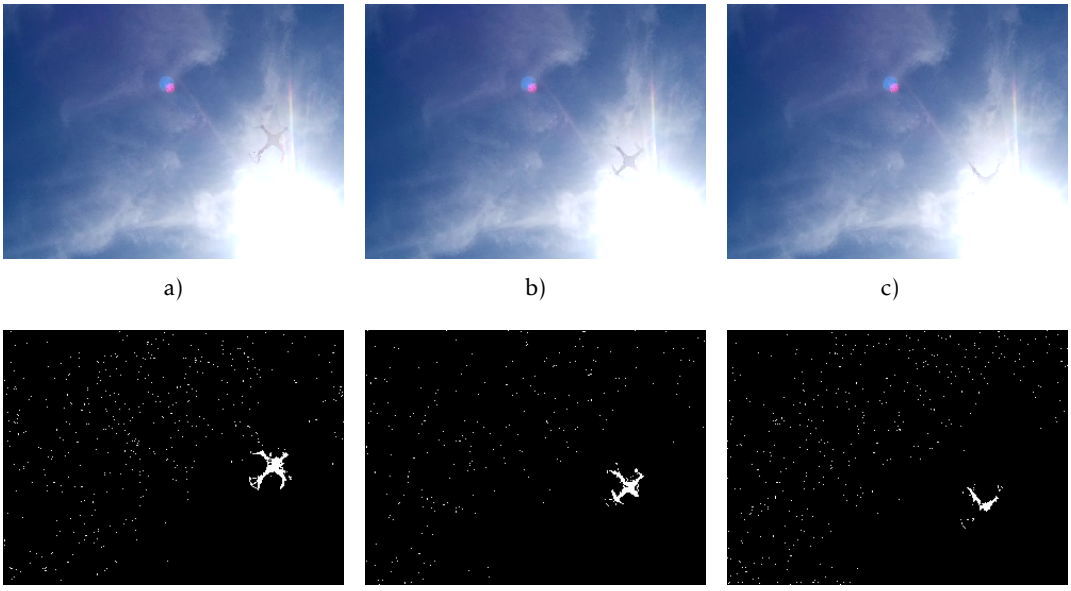


Figure 5.6: Effect of the sun in the detection of the UAV. When the UAV is flying directly below the sun in the image, it becomes hidden making its detection considerably harder or downright impossible.

Figure 5.7: Effect of sun glares on the Cloud Mask. Sun glares tend to be detected as clouds due to their low colour saturation values. This can produce false negatives in the detection mask, since areas of the images that are not clouds are considered as such, and therefore considered as noise and removed.

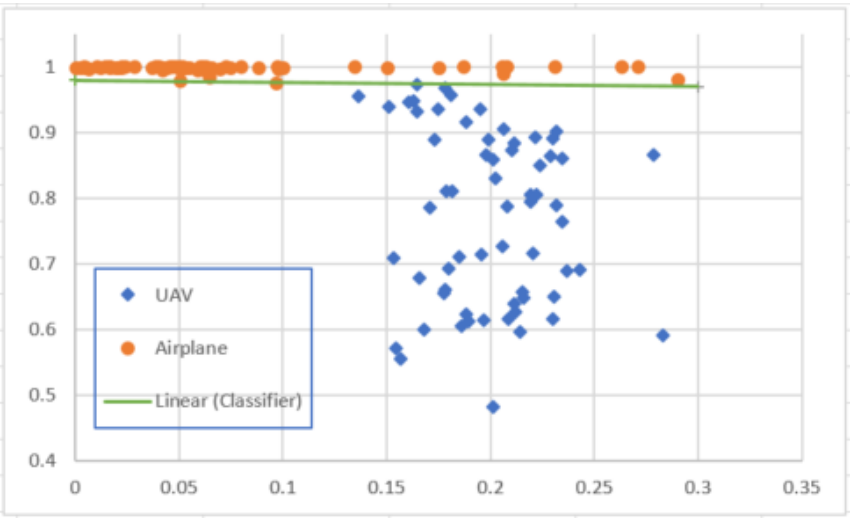


Figure 5.8: Entropy/Uniformity plot. Every dot corresponds to an object in given frame. Orange circles correspond to values belonging to aeroplanes, the blue diamonds belong to UAVs. The green line represents the classifier obtained from the training set.

5.3.2.1 Discussion

It is safe to assume objects with uniformity values above 0.97 should be considered aeroplanes while objects below are UAVs, and so, the function given by:

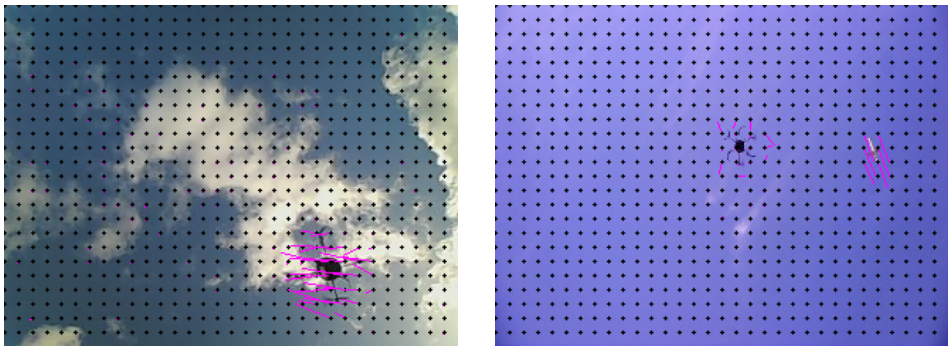
$$f(x) = -0.0333x + 0.98 \tag{5.1}$$

is used as the classifier to distinguish between aeroplanes and UAV. Points below it are considered UAVs, and above it aeroplanes. If the values are too close to the line, the entropy values are used as a tie-breaker criteria. UAVs entropy values never go below 0.125, so objects close to the line with entropy values above it have a higher likelihood to be UAVs. The classifier was tested by classifying every object present in the dataset including the training set. It showed to perform the correct identification 88,54% of the cases.

UAV samples with high uniformity, probably occur during quick translation movements, as the one seen in Fig. 5.9 where the chaotic patterns of the propellers become less prominent. Since translation movements have larger a magnitude than the propellers movements, the flows from the region will mostly correspond to the translation movement which are uniform in nature. For most all other situations the UAV movement pattern is clearly distinct from the aeroplane's, Fig. 5.10.

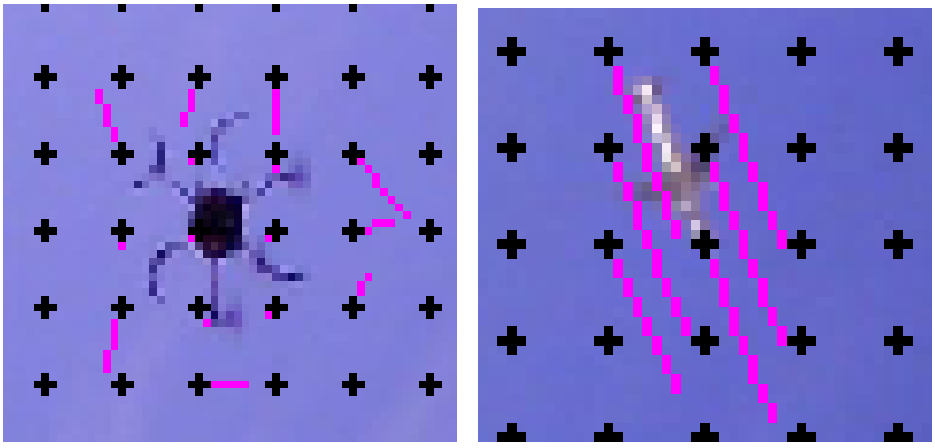
5.4 Tracker

For the assessment of the tracker module, at intervals of 10 frames for each video in the dataset, the correct position of the UAV was determined and compared to the position given by the system tracker. The distance between them was computed and normalized in relation to the measured size of the UAV in the image, given by:



a) b)

Figure 5.9: a) Effect of a UAV translation movement in the overall optical flow. When the UAV performs a quick translation movement most of the obtained flows belong to that movement and the chaotic flows belonging to the propellers movement is partially lost. This results in the UAV's movement pattern becoming more similar to the aeroplane's movement that can be seen in b).



a) b)

Figure 5.10: Zoomed aeroplane and UAV flow patterns. The flows belonging to the UAV a) have more chaotic pattern than the ones belonging to the aeroplane which are generally very uniform and linear.

$$error_{\%} = \frac{d_{error}}{d_{uav}} * 100; \tag{5.2}$$

where d_{error} is the euclidean distance between the estimated position and the real one (pixels); d_{uav} is is the correct radius of the UAV (pixels).

Furthermore, to improve the analysis, every frame from the dataset was classified, based on its content, into:

- **Moving:** UAV is moving at a significant speed:
- **Hovering:** UAV is hovering or moving at a significantly low velocity.
- **Partially Out:** UAV is partially outside of the camera's FoV.

The process was repeated for the different outputs: raw data coming directly from single frame analysis, after applying the KF, and putting the averaged position heuristic algorithm working on top of the KF, with $N_{avg} = 5$ and $N_{avg} = 8$. The results shown in Table 5.8 were obtained.

Table 5.8: Tracker Results

	Tracker error				
Method	Avg.(%)	Hover(%)	Mov.(%)	Out.(%)	Real(cm)
Raw	15,2	8,0	17,2	25,1	14,9
KF	7,6	5,6	9,8	13,3	7,9
$N_{avg} = 5$	6,6	5,1	10,8	13,0	6,2
$N_{avg} = 8$	10,1	6,9	18,9	15,1	12,1

5.4.1 Discussion

Results are best in Hovering frames since the tracker has more frames available without changes in the UAV's position, to more accurately pinpoint it. Moving frames are somewhere in between, with the output of the KF showing the best results. Using $N_{avg} = 8$ the results seem to deteriorate when compared to $N_{avg} = 5$. This happens due to the delay effect that using a larger amount of past frames to compute current position introduces.

The KF and our tracking algorithm with $N_{avg} = 5$ show the best overall results. KF by itself seems snappier, very quickly adjusting to the UAV translation movements while the averaged position with $N_{avg} = 5$ has noticeable a "delay effect" in moving frames, despite this it always ends up "catching up" when the UAV starts moving more slowly again, Fig. 5.11. Overall in hovering frames the averaged UAV position seems to provide a significant improvement when to compared to the KF.

When the UAV is partially outside the frame, the error increases considerably in all outputs, due to being hard to account for how much of the UAV is outside the frame, as

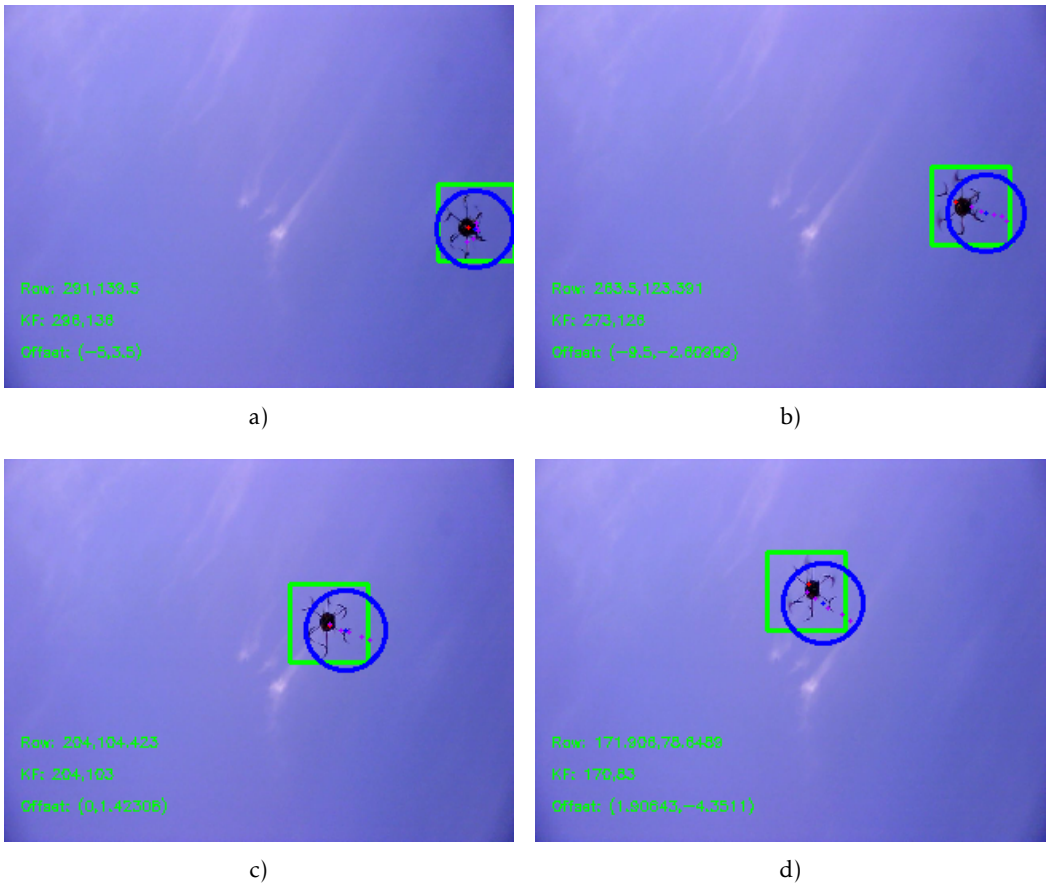


Figure 5.11: Averaged position delay effect. When the UAV performs quick translation movements the tracker based on the UAV averaged past positions presents a slight delay while following the UAV. This is due to the contribution of past frames into the average computation. a) The averaged position is correctly computed as seen by the blue circle. During b), c) and d) the UAV is performing quick movements, which results in the tracker having difficulties in keeping up.

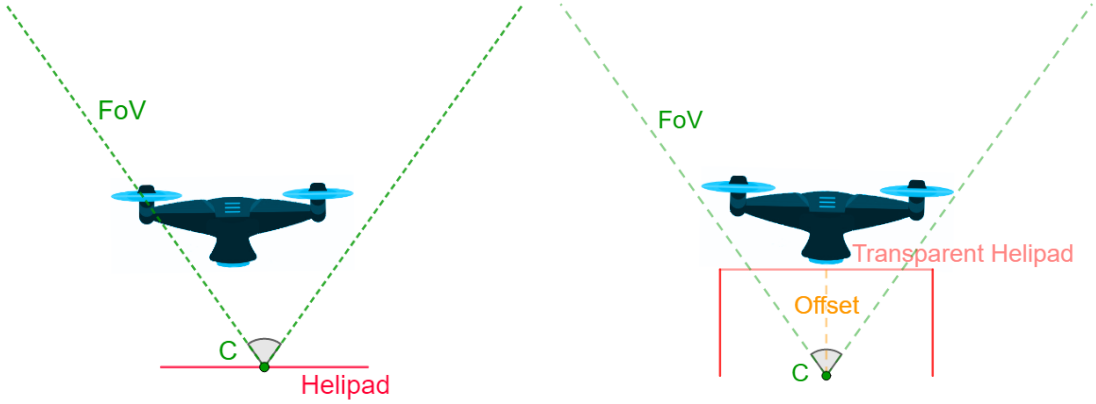
can be seen in Fig.5.12. This renders difficult to compute its centre with high precision. This may be especially problematic during the final part of the landing, where the UAV may not fit into the camera's FoV.

There are possible two solutions for the UAV not fitting in camera's FoV during the final centimetres of the descent. The first is the use of high FoV cameras or even 360°& ones. This however, has the drawback of increasing the image's resolution and in turn increasing the amount of computational power needed. While it is always possible to downscale the images, downscales from high resolutions to significantly lower ones tends to create artefacts in the image which can lead to increased noise. The second solution is the positioning of the camera farther down, effectively creating an offset between the camera point of view and the landing platform. This can be achieved by having a transparent landing platform, and the camera positioned below it. This way the camera can still capture the UAV, but it won't completely fill the camera's field of view when it is at

a) b)

Figure 5.12: When the UAV is partially outside of the frame accurate computation of its centre becomes considerably harder. This is especially problematic when the UAV is closer to the helipad since that, depending on the camera's FoV, the UAV may, or not, fit completely in the frame.

the platform's altitude, Fig. 5.13. The two solutions can even be combined decreasing the required offset, the FoV, or both.



a) b)

Figure 5.13: If the camera has not a wide enough FoV, during the final centimetres of landing the UAV won't fit completely in the image(a).If an offset between the landing platform and the camera's point of view is created, the problem can be solved. UAV's image retrieved from [52]

5.5 Controller

For the testing of the Controller module, a simulated environment was used, Fig.5.14. The following scenario was established: There is a static Helipad, placed on top of a hill, equipped with GPS and a camera placed at its centre pointing upwards. The UAV is also

equipped with a GPS and additionally has a IMU for pose estimation. Both helipad and UAV, are capable of communicating with each other and exchange data. The UAV starts the simulation at a considerable distance of the helipad with the goal being to perform a safe landing on the helipad. To accomplish that, the UAV should use GPS coordinates to close the distance. When the UAV enters inside the helipad's camera FoV, the helipad should start to detect it and then establish the relative positions between them. After it is done, the helipad should start relaying commands to the UAV so the landing procedure can start. Due to being a simplified simulated environment where there are no clouds or aeroplanes in the sky, and the UAVs's propellers are not dynamically animated, a simpler version of the detection algorithm was run, Fig. 5.15. The Object Detection module is run with a higher h_{bs} value so the detection can be performed without relying on the propellers movement. The identification module is also completely bypassed since there is no need to distinguish the UAV from other objects. Due to the fact, the primary goal is for the simulation to work as the controller module test-bed these limitations are not a problem. The simulation also works a concept proof of the main concepts throughout the dissertation, giving some the validation to the entire proposed model.

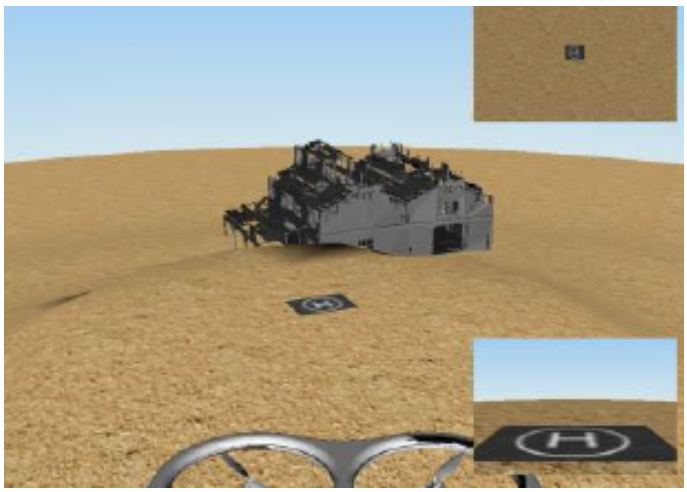
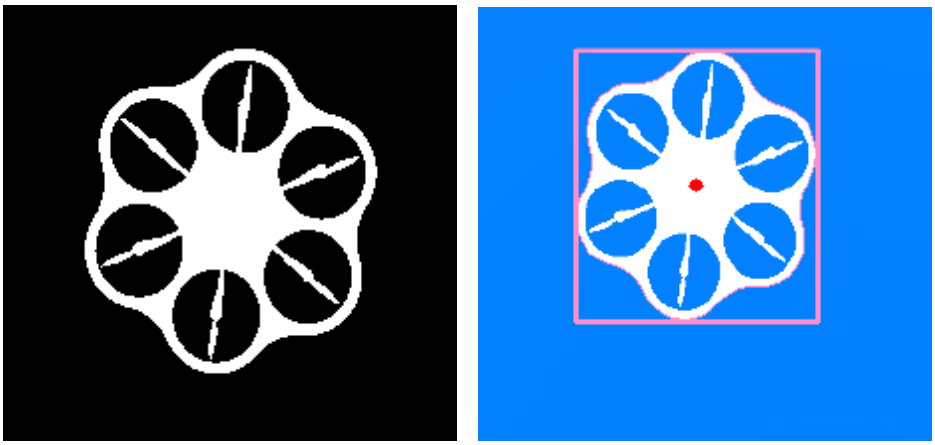


Figure 5.14: A general overview of the simulation environment. The helipad is set on top of the hill. The UAV starts far from the helipad having to rely on its GPS to navigate. Once the UAV is close to the helipad, the camera on the ground should detect the it and the landing manoeuvre can start.

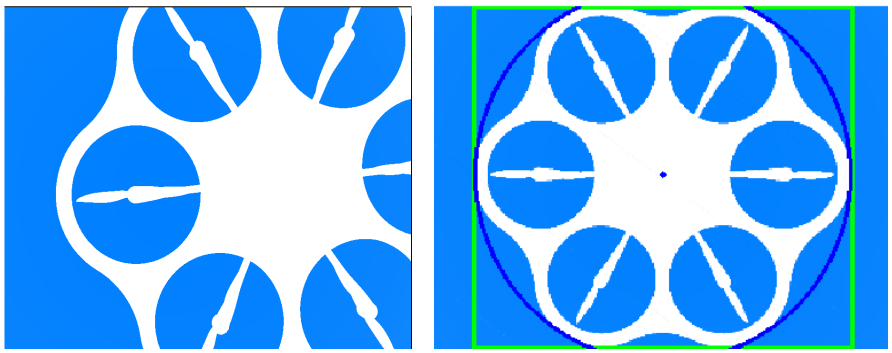
5.5.1 Discussion

The control module proved to be successful in the safe landing of the UAV, in a moderate amount of time, with an average final distance from the helipad's centre of 24 cm. As expected the same problem that occurs in real footage, occurred in the simulation. In the final centimetres of the descent, the UAV does not fit totally in the image, resulting in an excessive error in the UAV's position computation, which makes the task of landing close to the helipad's centre considerably harder.



a) b)

Figure 5.15: Simulated helipad's camera. a) MoG mask obtained from camera's footage. b) The sky in the simulation is a simple blue background with no clouds or planes, therefore a simpler version of the detection algorithm is run.

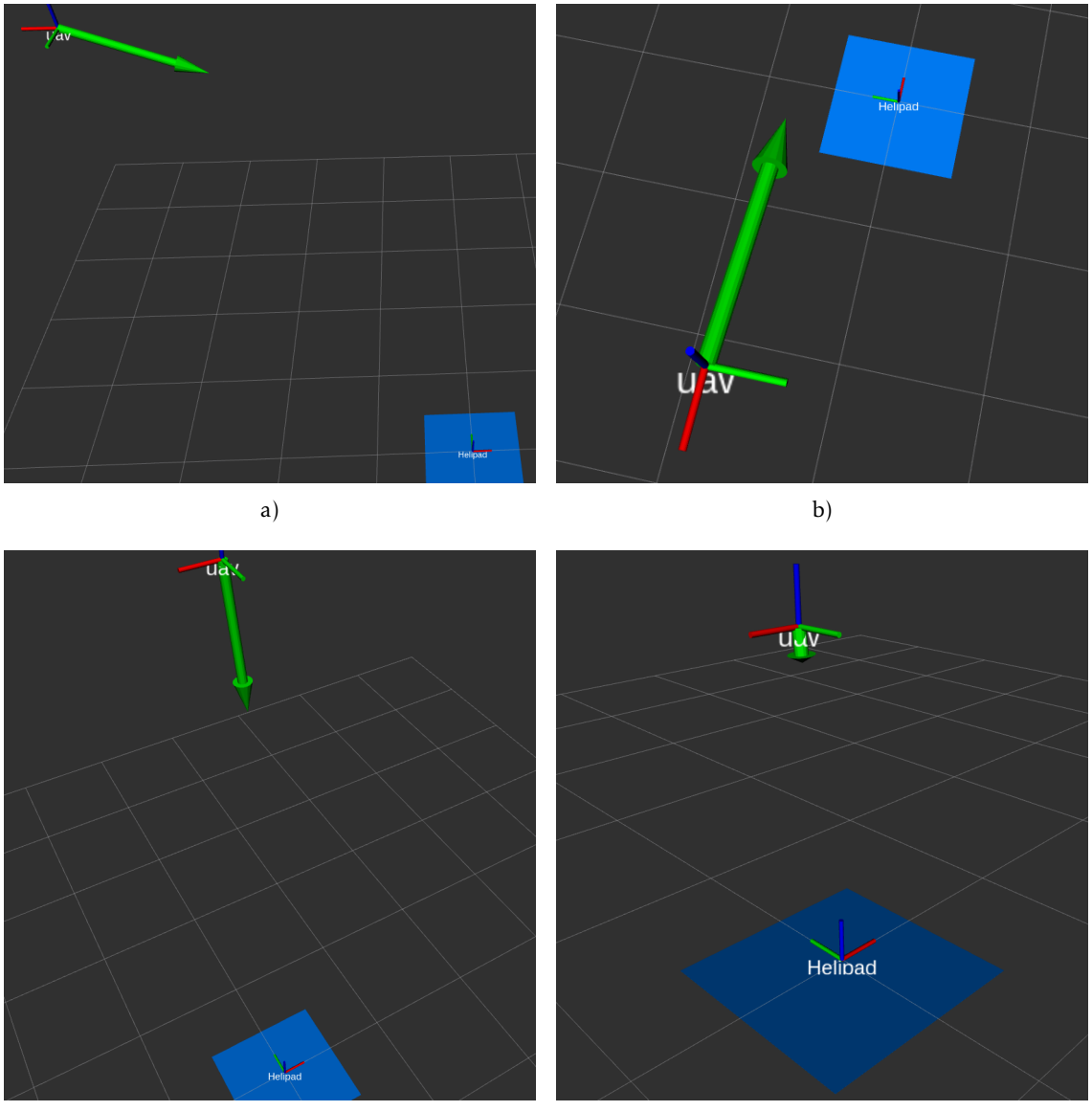


a) b)

Figure 5.16: a) When the UAV is close to the helipad, it does not fit completely in the image, making the computation of the UAV's centre prone to error. b) With an added offset between the camera and the helipad's surface, even with the UAV sitting on top of the helipad, it still fits into the camera's image completely.

To minimize this problem one of the solutions presented in 5.4 was applied. A second scenario was established identical to the first one but with the difference that now the helipad is partially transparent and the camera is positioned a few centimetres below it (see 5.4). This way, when the UAV is close to the helipad's surface it will still fit into the camera's FoV, Fig. 5.16. In this second scenario the results improved significantly with the UAV being able to land closer to helipad's centre. The landing was able to be performed in a shorter amount of time with the maximum distance from the helipad, from all the trials performed, registered at 13 cm.

Some moments during the landing manoeuvre, as well as a graphical representation of the velocity command applied in each one of them, can be seen in Fig. 5.17.



a) b)

c) d)

Figure 5.17: Velocity commands example in 3D perspective obtained via simulator. Blue square represents the helipad. The green arrow represents a v_c, with certain orientation and magnitude. a),b) When the UAV is far from the helipad the v_c should have a large magnitude to quickly guide the UAV to the helipad top. c),d) when the UAV is directly above the helipad the v_c should point downwards, with its magnitude decreasing as the altitude gets lowers.

Chapter 6

Conclusions and Future Work

In this chapter, a summary of the results achieved in this dissertation is given, as well as some directions for future research and possible improvements to the system.

6.1 Conclusion

This dissertation proposes a cooperative approach to perform autonomous landing of MR-VTOL UAVs. It introduces an additional robotic element in the scene: a "smart" helipad which is an active element capable of communication and do sensory data gathering. Resorting to a camera placed at its centre with the optical axis pointing up towards the sky, the helipad provides additional information about the UAV's relative position. The system is designed to either work by itself, being the primary method to guide the UAV during its descent, or being part of a more complex system with additional queues providing redundancy and increased overall robustness.

Due to its relatively motionless aspect the sky can be interpreted as a background, and used to ease the extraction of foreign objects in the footage from the helipad's camera, using a BS algorithm, in this particular instance MoG. By tuning the algorithm to be very fast to adapt to changes in the background, and with some additional image processing, most of the cloud movement can be removed. The algorithm proved to be efficient, being capable of outputting a binary mask containing only the foreign objects in the image i.e. planes, UAVs and birds, in around 93,44% of the analysed frames.

To perform the distinction between the different objects present in the image, Farneback optical flow algorithm was applied to extract the motion patterns of each object. These patterns are then analysed using two different criteria: entropy and cosine similarity. Since the UAV presents a significantly more chaotic and irregular motion pattern, using

these two criteria it becomes possible to create a binary classifier that distinguishes between objects that are likely to be UAVs and which are not. Since frame-by-frame analysis can be prone to error a tracking algorithm using KF as basis was employed. An weighted average of the UAV's past position is computed as well, along with its estimated velocity to minimize errors and compute the centre with better precision. This heuristic working on top of KF showed some improvements when compared to the KF output, especially during frames where the UAV is moving slowly, with an average tracking error of 6.6%.

Finally, a high-level control system with the goal of landing the UAV on the landing pad was proposed. The control system takes into account the relative position between the UAV and the helipad, their altitude differences and horizontal distances, issuing sharper commands the farther the UAV is from the goal and softer the closer it becomes. The controller uses the concept of landing zone based on a multiplicative inverse curve. The landing zone acts as an area where the UAV should be to safely perform the landing. The magnitude of the movement follows a negative Gompertz function: at higher altitudes the magnitude is higher, slowly decreasing as the altitude gets lower and becoming close to zero in the final centimetres of the descent.

As seen by the experimental results presented in chapter 5, the vision modules were validated against a diverse set of videos with both cloudy and clear sky footage, at different times of the day and with multiple UAV, with distinct morphologies, being used as test subjects. The method showed to be capable of handling most environments with acceptable results, detecting and extracting the UAV's position with an average of centimetres of 6,2 cm of error in best case scenario. The method has also showed to be able to recover from momentary failures in its modules caused, for example, by sudden luminosity variations in the video footage.

The main limitation of the system occurs when the sun is high in sky, effectively, creating saturated zones in the image, from where it is impossible to extract any information. This can effectively hide the UAV in the image making its detection downright impossible. One way to minimize the problem is adapting the camera lens to work in such situations. Yet another problem, can arise from the fact that the UAV may not fit in the helipad's camera FoV during the final centimetres of the descent. As mentioned in chapter 5 when part of the UAV is outside of the image tracking error increases considerably. A solution to this problem is also presented in chapter 5, and it works by adding an offset between the camera's point of view and landing surface.

Finally, the proposed model contributes to the research of autonomous landing systems which are a requirement to the fulfilment of truly autonomous UAV operations. By offering a different approach from traditional methods, this dissertation seeks to bring a new perspective on the entire problem. The proposed method can work together with the classic approaches, overcoming some of their shortcomings, while providing a more robust system capable of handling a wider array of environmental conditions.

6.2 Future Work

There are some possible areas where the method could be further developed. The first is obviously the testing of the control system on a real UAV to validate the simulated environment results. Real-life scenarios bring additional challenges such as the presence of wind that can have a significant impact on the performance of the control system.

Yet another possible way to overcome the difficulty of computing with precision the UAV's centre when it is close to the camera is through the use of Light Detection And Ranging (LIDAR)s. One or multiple LIDARs can be positioned on to the helipad's surface, sweeping the area right above it. When the UAV is close to the helipad their data can be cross-referenced originating a point cloud, which can be used to more precisely compute of the UAV's position during final centimetres of the landing.

A further study on the effects of varying some parameters have on the tracker's overall performance can also be conducted. Being able to dynamical adjust parameters, depending on the footage characteristics, could also be of interest. For instance, in situations where there is little or close to no cloud presence in the sky, the BS history, h_{bg} could have higher values since there is no need to remove cloud movement from the foreground mask. This in turn, could allow for a more accurate extraction of the UAV position and size, since the UAV wouldn't be so "fragmented"in the binary mask.

The object identification classifier can be further developed, by adding additional criteria and metrics to be evaluated, and applying Machine Learning (ML) concepts, such as neural networks and support vector machines, which can be actively trained to improve identification.

Another interesting development would be the use of up-to-date Graphical Processing Unit (GPU)s, as they are more efficient and faster at performing calculations involving matrices and vectorial operations than their counterparts Central Processing Unit (CPU). They possess parallel hardware capabilities making them more efficient and faster when there is the possibility to process data in parallel, which is especially useful for some vision algorithms. Therefore they could be exploited to greatly improve computation, freeing more CPU resources for other tasks.

The development of a "smart helipad"capable of guiding UAV on their landings could also be interesting. Since the proposed method is model-free and can work without requiring the UAV to perform any specific processing, the helipad is in complete charge of the landing. The helipad could be a deployable element capable of communicating in a standard protocol with UAVs, becoming the only asset necessary for the autonomous landing of any kind of MR-VTOL UAVs.

6.3 Dissemination

Some of the concepts covered in this dissertation can be additionally viewed in the following publication, co-authored by the author:

- Prates, P.A., Mendonça, R., Lourenço, A., Marques, F., Matos-Carvalho, J.P. and Barata, J. (2018). Vision-based UAV detection and tracking using motion signatures. 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2018).

Bibliography

[1] J. F. Keane and S. S. Carr. "A brief history of early unmanned aircraft." In: *Johns Hopkins APL Technical Digest* 32.3 (2013), pp. 558–571.

[2] S. M. Adams and C. J. Friedland. "A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management." In: *9th International Workshop on Remote Sensing for Disaster Response*. 2011, p. 8.

[3] K Pratt, R. Murphy, S Stover, and C Griffin. "Requirements for semi-autonomous flight in miniature uavs for structural inspection." In: *AUVSI's Unmanned Systems North America. Orlando, Florida, Association for Unmanned Vehicle Systems International* (2006).

[4] E. Pinto, P. Santana, F. Marques, R. Mendonça, A. Lourenço, and J. Barata. "On the design of a robotic system composed of an unmanned surface vehicle and a piggybacked vtol." In: *Doctoral Conference on Computing, Electrical and Industrial Systems*. Springer. 2014, pp. 193–200.

[5] E. T. Steimle, R. R. Murphy, M. Lindemuth, and M. L. Hall. "Unmanned marine vehicle use at Hurricanes Wilma and Ike." In: *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*. IEEE. 2009, pp. 1–6.

[6] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar. "UAVs for smart cities: Opportunities and challenges." In: *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE. 2014, pp. 267–273.

[7] S. Banker. *Amazon And Drones – Here Is Why It Will Work*. Accessed: 2018-02-19. Dec. 2013. URL: https://www.forbes.com/sites/stevebanker/2013/12/19/amazon-drones-here-is-why-it-will-work/#10e48f055e7d.

[8] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler. "Sensor planning for a symbiotic UAV and UGV system for precision agriculture." In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1498–1511.

[9] G. Grenzdörffer, A Engel, and B Teichert. "The photogrammetric potential of low-cost UAVs in forestry and agriculture." In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 31.B3 (2008), pp. 1207–1214.

[10] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek. "Autonomous UAV surveillance in complex urban environments." In: *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*. IEEE Computer Society. 2009, pp. 82–85.

[11] Z. Li, Y. Liu, R. Walker, R. Hayward, and J. Zhang. "Towards automatic power line detection for a UAV surveillance system using pulse coupled neural filter and an improved Hough transform." In: *Machine Vision and Applications* 21.5 (2010), pp. 677–686.

[12] A. Puri. "A survey of unmanned aerial vehicles (UAV) for traffic surveillance." In: *Department of computer science and engineering, University of South Florida* (2005), pp. 1–29.

[13] Y. Zhao and H. Pei. "An improved vision-based algorithm for unmanned aerial vehicles autonomous landing." In: *Physics Procedia* 33 (2012), pp. 935–941.

[14] G. R. Bradski. "Real time face and object tracking as a component of a perceptual user interface." In: *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*. IEEE. 1998, pp. 214–219.

[15] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme. "Vision-based autonomous landing of an unmanned aerial vehicle." In: *Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on*. Vol. 3. IEEE. 2002, pp. 2799–2804.

[16] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme. "Visually guided landing of an unmanned aerial vehicle." In: *IEEE transactions on robotics and automation* 19.3 (2003), pp. 371–380.

[17] F. L. L. Medeiros, V. C. F. Gomes, M. R. C. de Aquino, D. Geraldo, M. E. L. Honorato, and L. H. M. Dias. "A computer vision system for guidance of vtol uavs autonomous landing." In: *Intelligent Systems (BRACIS), 2015 Brazilian Conference on*. IEEE. 2015, pp. 333–338.

[18] C. S. Sharp, O. Shakernia, and S. S. Sastry. "A vision system for landing an unmanned aerial vehicle." In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*. Vol. 2. Ieee. 2001, pp. 1720–1727.

[19] A. Borowczyk, D.-T. Nguyen, A. Phu-Van Nguyen, D. Q. Nguyen, D. Saussié, and J. Le Ny. "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle." In: *IFAC-PapersOnLine* 50.1 (2017), pp. 10488–10494.

[20] S. Lange, N. Sunderhauf, and P. Protzel. "A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments." In: *Advanced Robotics, 2009. ICAR 2009. International Conference on*. IEEE. 2009, pp. 1–6.

[21] A Alaimo, V Artale, C Milazzo, A Ricciardello, and L Trefiletti. "Mathematical modeling and control of a hexacopter." In: *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*. IEEE. 2013, pp. 1043–1050.

[22] R. Baranek and F. Šolc. "Modelling and control of a hexa-copter." In: *Carpathian Control Conference (ICCC), 2012 13th International*. IEEE. 2012, pp. 19–23.

[23] A. Alaimo, V. Artale, C. L. R. Milazzo, and A. Ricciardello. "PID controller applied to hexacopter flight." In: *Journal of Intelligent & Robotic Systems* 73.1-4 (2014), pp. 261–270.

[24] B Kada and Y Ghazzawi. "Robust PID controller design for an UAV flight control system." In: *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 2. 1-6. 2011.

[25] T. Sangyam, P. Laohapiengsak, W. Chongcharoen, and I. Nilkhamhang. "Path tracking of UAV using self-tuning PID controller based on fuzzy logic." In: *SICE Annual Conference 2010, Proceedings of*. IEEE. 2010, pp. 1265–1269.

[26] M. A. Olivares-Méndez, I. F. Mondragón, P. Campoy, and C. Martínez. "Fuzzy controller for uav-landing task using 3d-position visual estimation." In: *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*. Ieee. 2010, pp. 1–8.

[27] M. Collotta, G. Pau, and R. Caponetto. "A real-time system based on a neural network model to control hexacopter trajectories." In: *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*. IEEE. 2014, pp. 222–227.

[28] V Artale, M Collotta, G Pau, and A Ricciardello. "Hexacopter trajectory control using a neural network." In: *AIP Conference Proceedings*. Vol. 1558. 1. AIP. 2013, pp. 1216–1219.

[29] C. Phan and H. H. Liu. "A cooperative UAV/UGV platform for wildfire detection and fighting." In: *System Simulation and Scientific Computing, 2008. ICSC 2008. Asia Simulation Conference-7th International Conference on*. IEEE. 2008, pp. 494–498.

[30] J. Silva, R. Mendonça, F. Marques, P. Rodrigues, P. Santana, and J. Barata. "Saliency-based cooperative landing of a multirotor aerial vehicle on an autonomous surface vehicle." In: *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*. IEEE. 2014, pp. 1523–1530.

[31] M. Piccardi. "Background subtraction techniques: a review." In: *Systems, man and cybernetics, 2004 IEEE international conference on*. Vol. 4. IEEE. 2004, pp. 3099–3104.

[32] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. "Review and evaluation of commonly-implemented background subtraction algorithms." In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE. 2008, pp. 1–4.

[33] C. Stauffer and W. E. L. Grimson. "Adaptive background mixture models for real-time tracking." In: *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*. Vol. 2. IEEE. 1999, pp. 246–252.

[34] E. Hayman and J.-O. Eklundh. "Statistical background subtraction for a mobile observer." In: *null*. IEEE. 2003, p. 67.

[35] Y. Sheikh, O. Javed, and T. Kanade. "Background subtraction for freely moving cameras." In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 1219–1225.

[36] B. G. Lindsay. "Mixture models: theory, geometry and applications." In: *NSF-CBMS regional conference series in probability and statistics*. JSTOR. 1995, pp. i–163.

[37] P. W. Power and J. A. Schoonees. "Understanding background mixture models for foreground segmentation." In: *Proceedings image and vision computing New Zealand*. Vol. 2002. 2002.

[38] A. M. McIvor. "Background subtraction techniques." In: *Proc. of Image and Vision Computing* 4 (2000), pp. 3099–3104.

[39] Z. Zivkovic. "Improved adaptive Gaussian mixture model for background subtraction." In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 2. IEEE. 2004, pp. 28–31.

[40] S. Dowling. *Dirichlet Process Mixture Models and their application in bioinformatics*. Accessed: 2018-03-16. URL: https://dirichletprocess.weebly.com/clustering.html.

[41] G. Farneback. "Fast and accurate motion estimation using orientation tensors and parametric motion models." In: *Pattern Recognition, 2000. Proceedings. 15th International Conference on*. Vol. 1. IEEE. 2000, pp. 135–139.

[42] F. Raudies. *Optic flow*. Accessed: 2018-03-15. July 2013. URL: http://www.scholarpedia.org/article/Optic_flow.

[43] B. Rokers. *Animated Example of the Aperture Problem Illusion*. Accessed: 2018-03-19. 2005. URL: http://psych.wisc.edu/vision/research.php.

[44] G. Farneback. "Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field." In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. Vol. 1. IEEE. 2001, pp. 171–177.

[45] G. Farnebäck. "Two-frame motion estimation based on polynomial expansion." In: *Scandinavian conference on Image analysis*. Springer. 2003, pp. 363–370.

[46] J. Rubinstein. *How does a Global Shutter Work?* Accessed: 2018-03-19. June 2013. URL: http://www.digitalbolex.com/global-shutter/.

[47] Q. Li, W. Lu, and J. Yang. "A hybrid thresholding algorithm for cloud detection on ground-based color images." In: *Journal of atmospheric and oceanic technology* 28.10 (2011), pp. 1286–1296.

[48] S. Dev, Y. H. Lee, and S. Winkler. "Systematic study of color spaces and components for the segmentation of sky/cloud images." In: *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 5102–5106.

[49] S. Dev, Y. H. Lee, and S. Winkler. "Color-based segmentation of sky/cloud images from ground-based cameras." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10.1 (2017), pp. 231–242.

[50] P. Santana, R. Mendonça, and J. Barata. "Water detection with segmentation guided dynamic texture recognition." In: *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1836–1841.

[51] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. 2nd. O'Reilly Media, Inc., 2013. ISBN: 1449314651, 9781449314651.

[52] javier. *UAV*. Accessed: 2018-03-20. URL: https://pngtree.com/freepng/uav_2663590.html.

ANNEX I

Dataset details

Table I.1: Complete details of dataset used in the testing of the vision algorithm.Avg. refers to the arithmetic average of all the videos from the dataset. Avg.Weighthed refers to the average weighted by the number of frames of each video.

Dataset Results				
Video ID	Nº of frames	Duration(s)	Detection(%)	Tracking Error(%)
1	709	62,2	92,81	7,8
2	1500	52,5	72,00	9,1
3	2334	81,7	79,09	13,7
4	1129	56,3	70,77	11,5
5	729	71,8	84,91	7,6
6	896	53,7	99,67	4,3
7	4108	163,8	95,62	8,9
8	1266	59,3	95,66	7,6
9	776	43,6	99,48	5,0
10	2110	136,6	91,18	6,1
11	2536	130,6	95,27	3,9
12	2638	113,3	97,57	4,8
13	2851	200,7	92,67	7,0
14	1130	103,3	97,26	7,4
15	661	56,5	99,39	7,5
16	3691	155,7	99,76	5,3
17	1399	95,4	99,14	6,0
18	1800	110,1	97,61	4,7
19	3289	166	96,50	7,3
20	2559	154	95,51	8,4
21	3172	172	99,24	7
22	1920	181	99,70	7
23	1354	59	99,34	7,5
Avg.	44557	2479,1	93,44	7,19
Avg. Weighted	44557	2479,1	94,15	7,18