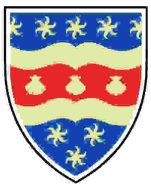
This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognize that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's prior consent.



UNIVERSITY OF
PLYMOUTH

Autonomous Drone Landings on an Unmanned Marine Vehicle using Deep Reinforcement Learning

by

Riccardo Polvara

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Engineering

July 2018

Abstract

Autonomous Drone Landings on an Unmanned Marine Vehicle using Deep Reinforcement Learning
Riccardo Polvara

THIS thesis describes with the integration of an Unmanned Surface Vehicle (USV) and an Unmanned Aerial Vehicle (UAV, also commonly known as drone) in a single Multi-Agent System (MAS). In marine robotics, the advantage offered by a MAS consists of exploiting the key features of a single robot to compensate for the shortcomings in the other. In this way, a USV can serve as the landing platform to alleviate the need for a UAV to be airborne for long periods time, whilst the latter can increase the overall environmental awareness thanks to the possibility to cover large portions of the prevailing environment with a camera (or more than one) mounted on it. There are numerous potential applications in which this system can be used, such as deployment in search and rescue missions, water and coastal monitoring, and reconnaissance and force protection, to name but a few.
The theory developed is of a general nature. The landing manoeuvre has been accomplished mainly identifying, through artificial vision techniques, a fiducial marker placed on a flat surface serving as a landing platform. The *raison d'etre* for the thesis was to propose a new solution for autonomous landing that relies solely on onboard sensors and with minimum or no communications between the vehicles. To this end, initial work solved the problem while using only data from the cameras mounted on the in-flight drone. In the situation in which the tracking of the marker is interrupted, the current position of the USV is estimated and integrated into the control commands. The limitations of classic control theory used in this approached suggested the need for a new solution that empowered the flexibility of intelligent methods, such as fuzzy logic or artificial neural networks. The recent achievements obtained by deep reinforcement learning (DRL) techniques in end-to-end control in playing the Atari video-games suite represented a fascinating while challenging new way to see and address the landing problem. Therefore, novel architectures were designed for approximating the action-value function of a Q-learning algorithm and used to map raw input observation to high-level navigation actions. In this way, the UAV learnt how to land from high latitude without any human supervision, using only low-resolution grey-scale images and with a level of accuracy and robustness. Both the approaches have been implemented on a simulated test-bed based on Gazebo simulator and the model of the *Parrot ARDrone*. The solution based on DRL was further verified experimentally using the *Parrot Bebop 2* in a series of trials. The outcomes demonstrate that both these innovative methods are both feasible and practicable, not only in an outdoor marine scenario but also in indoor ones as well.

Contents

List of Figures

List of Tables

Acknowledgements

This thesis is a result of a journey that lasted a bit less than three years but which could not be possible without the support and the wisdom of many people I met during my stay at the University of Plymouth.

First of all, I wish to thank my supervisory team, comprising Dr Sanjay Sharma, Dr Jian Wan, Dr Andrew Manning and Prof Robert Sutton. I recognise I was not the traditional quiet PhD student everyone thinks about, completely devoted to the cause and ready to do whatever he is asked for. I always consider myself more like a young rebel scientist with his own ideas and some difficulties to reach a compromise with. And this is why I am really grateful that my supervisory team aknowledged the working freedom I needed to express my full potential as a researcher. I am sure that the quality and the amount of work I achieved could not be possible if I was forced to follow a path I was not interested to explore. I would also like to express my gratitude to my thesis committee for accepting my invitation to read this work and providing useful feedbacks on the research I conducted.

On a more personal side, I am truly grateful to a large group of people that made my time in Plymouth enjoyable and unforgettable. The first and more important people I would like to mention are definitely the members of the Alps Ibex group, more precisely Valerio Biscione, Andrea Ferrario, Massimiliano Patacchiola, Luca Stirniman, Vincenzo Motta, Giovanni Carmantini, Luca Surace and Salvatore Lentini. In particular, Valerio was the guy that opened me to the social life in Plymouth and allowed me to meet some of the most important people of this journey. My first week in the city was rather boring, living with a girl that was never at home to chat with and working in an empty office. I will never forget that 5th November 2015 in which I decided to drop him a message to go and see the bonfire together. In that occasion, he not only showed himself like an exquisite person to talk with about every imaginable topic, but he also introduced me to many people which I will be bonded with for life. Andrea, instead, has been more than a brother for me. We shared the same house for more than two years, spending uncountable hours in the most diverse activities, from playing video games and watching movies, to talk about statistics, ethics, religion, passing by our weekly journey to Sainsbury's for groceries. I found in him the twin I have never had and I am confident our friendship will last for many years to come. Massimiliano has always been an inspiration for me, not only because older and wiser than me. His passion and dedication to work are unmatchable by anyone else I know. Strictly working with him for more than an year, and observing his finicky scientific approach, made me a better researcher indeed. The remaining members of the group, even though not present in Plymouth for all the length of my PhD, still represent a solid stone in this path and good friends always ready to offer me a suggestion or a laugh.

Then, it is the turn of Yogang Singh, the other PhD candidate of the research group I was part of. He has been an amazing mate during these three years and he was

always there, sit at the desk close to mine when I had doubts about my future. He was probably the first person to believe in me in Plymouth, always ready to cheer me up when I was sad because my code was not working or my submissions were rejected. We established a strong friendship that, I am sure, will last for a while, even now he moved across the Atlantic Ocean.

What follows is another group of people, known as the Fratercula Arctica. Its members comprise Clara Cutello, her boyfriend Francois Foerster and my longtime friend Marta Romeo. With them I probably spent the most fascinating week on my life, driving around the entire Iceland, taking numerous beautiful pictures of waterfalls, glaciers and volcanoes. It was an incredible adventure that made our friendship even stronger than before. I found in them three kind people ready to listen, assist and encourage me every time I need it. I will always choose you all as my favourite mates for the next trip to come.

The list of people I should thank is long, too long to dedicate a paragraph to every single individual that, for one reason or another, made me smile at least one day. For this reason, I am just listing all of them now, in a sparse order. I hope no one is getting offended because his/her name appears after someone else's one. Therefore, I want to thank the people living with me at 90 North Road East, namely Martina Fiori, Ilaria Torre, Frank Loesche, Emmanuel Senft, Samuele Vinanzi and Samuel Richardson, few colleagues at the Marine Building such as Carlos Perez-Collazo, Pierre Henry "Pilou" Oust and Alessandro Antonini, the basketball mates which made me leave my working desk on Thursday evening, Prof Angelo Cangelosi for his support and trust in me, all his students such as Debora Zanatto, Alexandre Antunes, Bahar Irfan, Mina Marpena, Ricardo de Azambuja, Frederico Klein, Gabriella Pizzutto, Baris Sehran, Mohammad Thabet, Pontus Loviken, Gabriel Noury and Daniel Hernandez Garcia.

Finally, my family with my mother Rosy, my father Luca, sister Federica, my cat Pablo and my beloved cocker Aaron. They gave me affection and unconditional encouragement, they have always been there, on the phone or physically, to support me and believe in me also when I was completely lost and ready to give up.

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award. Work submitted for this research degree at Plymouth University has not formed part of any other degree either at Plymouth University or at another establishment.

This work has been carried out by Riccardo Polvara under the supervision of Dr. Sanjay Sharma, Dr. Jian Wan, Dr. Andrew Manning and Prof. Robert Sutton.

A programme of advanced study was undertaken which included participation in multiple courses organised within the Research Development Programme of the University of Plymouth Graduate School, participation and successful completion of the PGCAP - A Postgraduate Certificate in Academic Practice (Intensive), participation in the Introduction to CUDA, Optimisation steps for neural networks with CUDA and Deep Learning with DIGITS, participation in the ERL Emergency/TRADR summer school 2017, participation in the MarineUAS winter school 2018, and participation in the supervision of Krishna Chouthankar during his project focused on UAV pose estimation during his visiting period at the University of Plymouth.

Relevant scientific seminars and conferences were regularly attended at which work was often presented. External institutions were visited for consultation purposed and several papers prepared for publication.

Regarding the content of this thesis, I must point out that the dynamical model of the drone used in this research has not been implemented from scratch, but I relied on the one realised by the Technical University of Munich. Also, the Extended Kalman filter used was already offered as standalone package in the ROS framework. As last, I used the help of Massimiliano Patacchiola to design and implement the Deep Q-Network used in one of the work here presented.

Word count for the main body of this thesis: **27589**

Signed: _____

Date: _____

Publications

In pursuing the research objectives, the finding of the studies conducted have and are in the process of being disseminated in a series of publications including journal

papers and conference proceedings. The following comprises a list of papers published or submitted for publication.

Parts of this thesis have been published by the author:

Abed W, **Polvara R**, Singh Y, Sharma S, Sutton R, Hatton D, Manning A, Wan Y (2016). Advanced feature extraction and dimensionality reduction for unmanned underwater vehicle fault diagnosis. *UKACC 11th International Conference on Control (CONTROL)*, Belfast, UK. DOI: 10.1109/CONTROL.2016.7737596

Polvara R, Sharma S, Sutton R, Wan J, Manning A (2016). Toward a Multi-agent system for Marine Observation. *Advances in Cooperative Robotics: Proceedings of the 19th International Conference on (CLAWAR)*, London, UK. DOI: 10.1142/9789813149137_0028

Singh Y, **Polvara R**, Sharma S, Hatton D, Wan J, Sutton R. (2016). Design of a Variable Buoyancy Engine for Small Scale Underwater Vehicle. *International Conference on Advances in Subsea Engineering, Structures and Systems (ASESS-2016)*, Glasgow, UK. http://hdl.handle.net/10026.1/8481

Polvara R, Sharma S, Wan J, Manning A, Sutton R (2017). Towards autonomous landing on a moving vessel through fiducial markers. *IEEE European Conference on Mobile Robotics (ECMR)*, Paris, France. DOI: 10.1109/ECMR.2017.8098671

Szyrowski T, Bogdan P, Khan A, Pemberton R, Sharma S, Singh Y, **Polvara R**. (2017) Range Extension for Electromagnetic Detection of Subsea Power and Telecommunication Cables. *International Conference on Marine Electromagnetics (MARELEC 2017)*, Liverpool, UK. http://hdl.handle.net/10026.1/12232

Polvara R, Sharma S, Wan J, Manning A, Sutton R (2017). Obstacle Avoidance Approaches for Autonomous Navigation of Unmanned Surface Vehicles. *Journal of Navigation*. 71(1), 241-256. DOI: 10.1017/S0373463317000753

Polvara R, Sharma S, Wan J, Manning A, Sutton R (2018). Vision-based Autonomous Landing of a Quadrotor on the Perturbed Deck of an Unmanned Surface Vehicle. *Drones*. 2(2):15. DOI: 10.3390/drones2020015

Polvara R, Patacchiola M, Sharma S, Wan J, Manning A, Sutton R, Cangelosi A. (2018). Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning. *International Conference on Unmanned Aerial System (ICUAS)*, Dallas, USA. DOI: 10.1109/ICUAS.2018.8453449

Polvara R, Sharma S, Wan J, Manning A, Sutton R (2019). Autonomous Vehicular Landings on the Deck of an Unmanned Surface Vehicle using Deep Reinforcement Learning. *Robotica*, 1-16. DOI: 10.1017/S0263574719000316

Parts of this thesis have been submitted to peer-review journals and are currently under review:

Polvara R, Patacchiola M, Sharma S, Wan J, Manning A, Sutton R, Cangelosi A. (under review). Autonomous Quadrotor Landing using Deep Reinforcement Learning. *https://arxiv.org/abs/1709.03339*

Chapter 1

Introduction

THIS chapter outlines the objectives of the research and presents an overview of the concepts that are developed through the thesis without delving into technical details. The main contribution of the research and the structure of the thesis are included in this introductory chapter.

1.1 Motivation

The motivation behind this thesis was to investigate the limitations affecting the autonomous navigation of unmanned surface vehicles (USVs). Autonomous navigation is an even more interesting and challenging task for unmanned vehicles, in particular for a USV. While traversing, multiple obstacles can pose a risk for the own vessel. Examples are offered by different size boats, buoys, divers, floating objects but also shores and coasts. Multiple advancements have taken place in satellite navigation for USV deployed for various applications such as military surveillance or bathymetric survey of shallow waters. Despite the efforts started in the 1990s during the Gulf War, the development of a robust and optimal path planner for USVs is still a crucial aspect towards (semi-)fully autonomy.

In this sense, several solutions are present in the literature based on constraints imposed by the presence of obstacles, the USV's geometry and dynamics. However, the uncertainties related to the marine environments and sensors limitation make the path planning problem even more complicated.

From here the notion to adopt an unmanned aerial vehicle (UAV), equipped with a complete sensor suite, as an additional eye looking at the environment has been formulated

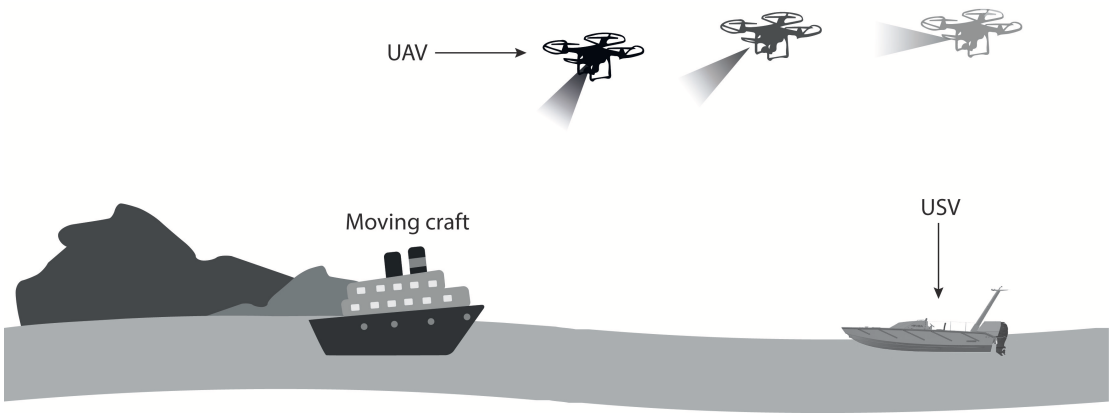


Figure 1.1: A UAV is integrated with an USV to increase the situation awareness of the environment.

to overcome the problems described above. This can be achieved exploiting the possibility to observe the world surrounding a USV from multiple perspectives and at long distances thanks to the possibility for the UAV to fly. In fact, the advantage represented by the deployment of a UAV is strictly linked with its ability to cover long distances in a short amount of time (its speed is usually much higher than that of a UAV) and being equipped with high-resolution cameras. Image data acquired from cameras can be used to map and better represent the environment in which the autonomous vessel is traversing. This representation of the world is then used to plan a collision-free path for the USV.

Therefore, it is possible to couple the USV and the UAV in a multi-agent system that can be deployed for multiple tasks, such as patrolling, water and coastal monitoring, or search and rescue scenarios. A graphical representation of the overall system is shown in Figure 1.1. Like in any other multi-robot system, one of the most delicate phases that needs to be addressed is the communication and the interaction among the vehicles involved. From the point of view of this thesis, the interaction is represented by landing the flying vehicle on the deck of the marine one, for example to recharge the batteries or just at the end of the task for which it has been used.

Owing to the possible adverse weather conditions affecting operation not only at sea but outdoor in general, it is essential to eliminate the reliance of a UAV on any external

signals such as GPS and UAV-to-USV communications. Hence the goal of this thesis, to give a contribution towards precise autonomous landing while relying only on the on-board sensor suite present on the UAV. Moreover, the recent interest and achievements obtained by deep learning techniques, in particular in the field of image classification, suggested the adoption of artificial neural networks for increasing the robustness of the image processing module whilst offering a flexible solution for controlling the UAV.

1.2 Contribution

The objective of this thesis was to create a controller for landing a UAV on the deck of a USV that relies only on on-board sensors, it is flexible, and proved to be robust to variable environmental conditions. In order to accomplish this goal, three steps have been planned in advance and summarised below:

1. Combining UAV's onboard sensors data and minimum inter-vehicle communications while exploiting computer vision, state estimation filters and classic control theory.

2. Proposing an end-to-end solution based on Deep Reinforcement Learning that directly maps raw observation inputs to high-level control actions.

3. Improve training process performed within a simulated environment in such a way to fill the reality gap existing when testing the algorithm with a real platform.

In the next section, these three phases are related to the structure of the thesis providing an overview of the work that has been done.

1.3 Outline of the thesis

This thesis comprises of 6 chapters organised similarly to the work-flow it was followed during the PhD programme. Most of the work has already been published in peer-reviewed conferences and journals, as reported the Author's declaration. This section will shortly describe the content of each chapter, explaining how they are linked together.

Following on from this introductory chapter, Chapter 2 provides a review of the available literature related to the topics of this thesis. It describes the most used path planning and obstacle detection techniques for a USV, pointing out the complexity of autonomous navigation, and suggests a multi-agents system UAV-USV as a solution to overcome the limitations of the existing methods. In particular, since this thesis is focused on the autonomous landing of the UAV on the deck of the USV, an overview of the existing methodologies for solving this task is offered.

Chapter 3 presents the material and the methods used in the thesis.

Chapter 4 and 5 form the core chapters of this research. They focus on the problem of autonomously landing on a platform, already discussed in Chapter 2. The first introduces briefly unmanned aerial vehicles, remarks the challenges posed in a marine environment and formally describes an approach based on artificial vision and classic control theory. However, this solution is not able to prevail over all the limitations of the platform used and understanding the reason for and overcoming these difficulties is the topic of the following chapter.

In Chapter 4, Deep Reinforcement Learning, a new trend in AI based on the combination of reinforcement learning and deep neural network, is introduced and applied for achieving autonomous landing for UAV without any human supervision or intervention.

Finally, the last chapter, Chapter 5, concludes the thesis discussing the research outcomes, drawing conclusions and providing suggestions for further work.

Chapter 2

Literature Review

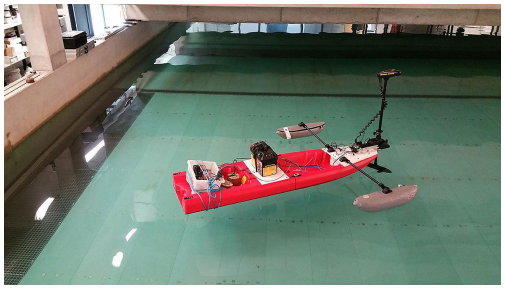
THE present chapter illustrates the state of the art methodologies regarding obstacle detection and path planning for an unmanned surface vehicle. Path planners are classified in local and global ones, and a special section has been allocated for intelligent methods. In the conclusive sub-section, the drawbacks and the limitations of the approaches described are reported and the purpose of this thesis is justified.

2.1 Introduction

Since their first development, the military community showed interest in unmanned surface vehicles (USVs) for their deployment in force protection and surveillance scenarios (Bertram 2008). Other examples are given by the *Spartan* produced by US Space and Naval Warfare System Center (SSC) San Diego, the *Delfim* developed by the Portuguese Dynamical Systems and Ocean Robotics (DSOR) Laboratory (Alves et al. 2006), and the *Springer* developed by the University of Plymouth (Naeem & Sutton 2009). Most of the vessels are dual-purpose vehicles, i.e. they can both be driven remotely and operated autonomously.

USVs usually adopt catamaran and kayak shapes (reported in Figure 2.1), for their roll robustness or ease of manufacture, and they are generally provided with a rudder-propeller system for propulsion and steering. Their kinematics are modelled ignoring pitch and roll, and considering an Earth-fixed frame and a body-fixed one (Bibuli et al. 2009). The first is used to express the position and the orientation $[x\ y\ z]^T$, while the second for surge u_r and sway velocities v_r relative to the water, yaw rate r and forces and moments. Linear and angular velocities are defined by these equations, expressed

(a) An example of catamaran.



(b) An example of Kayak.

Figure 2.1: Two example of autonomous surface vehicle developed at the University of Plymouth: a catamaran (a) and a kayak (b).

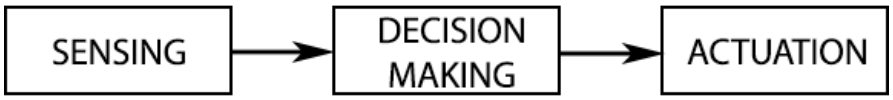


Figure 2.2: Architecture of an autonomous system.

in the Earth-fixed frame:

$$\begin{cases} \dot{x} = u_r cos\psi - v_r sin\psi + \dot{x}_C \\ \dot{y} = u_r sin\psi + v_r cos\psi + \dot{y}_C \\ \dot{\psi} = r \end{cases}$$

where $[\dot{x}_C\ \dot{y}_C]^T$ are the sea currents that are considered constant. Although in general, the marine environment introduces a number of serious disturbances which affect the control of a USV (Bandyophadyay et al. 2010). Examples are sea currents, tides, and sea-bed conformation. When transiting close to the coasts, the presence of tides dictates where a ship can go and how much time it will take to reach its destination. Despite the importance of weather conditions while traversing, there is only a small number of papers addressing this issue.

2.2 Collision avoidance solutions for USVs

The existence of obstacle detection and avoidance modules requires the combining the sensing and decision making components, in Figure 2.2, to navigate autonomously (Statheros et al. 2008, Tam et al. 2009, Hasegawa & Kouzuki 1987, Hasegawa 2009). The path planning problem has a long history in robotics, especially for Unmanned

Ground Vehicles (UGVs) (Fahimi 2008).A Path Planner (PP) is generally composed of a global path planner and a local one. The first (*deliberative*) must find a safe path connecting the actual position of the robot and its destination. The local (*reactive*) must react to immediate collisions, moving the vehicle on a different route. Referring to unmanned boats, obstacles are represented by other vessels, divers, buoys, rocks, harbours, islands and coasts.

The sensing module covers a key role towards full autonomy since it must identify the surrounding obstacles. When static, finding a safe path is easy and any graph search algorithm can be used. However, detecting collisions in time is still an open problem because of sensor limitations (e.g., false positive/negative due to noisy data). Moreover, the conformation of the environment can occlude the sensor's field of view, preventing a moving obstacle's detection. Therefore, autonomous navigation is a more complex task than connecting two points; multiple situations must be considered, especially under uncertainty. Hence, the scope of this paper is to offer an overview concerning the latest methods regarding collision avoidance for USVs and how to solve the problems described above.

The rest of the chapter is divided in the following way: Section 2.2.1 illustrates how to perceive the surrounding environment; in Section 2.2 the necessity of having a robust path planner is discussed, focusing on the difference between GPP and LPP in Subsection 2.2.2.2 and 2.2.2.3 respectively, and introducing intelligent methods in Subsection 2.2.2.4. In Section 2.3 concluding remarks are given.

2.2.1 Collision detection

To avoid obstacles, an accurate representation of the environment is required. To obtain it, data from sensors (often integrated with electronic charts) are combined to give a Two-Dimensional (2D)/Three-Dimensional (3D) model.

With regards to technology advancements, the methods for collision identification have dramatically changed in the last fifty years (Kemp et al. 2012) and sensors advantages and limitations are summarised in Table 2.1. In Almeida et al. (2009) a radar classifies

Table 2.1: Advantages and limitation of various sensors within USVs (Liu et al. 2016)

Sensors	Advantages	Limitations
Radar	1. Long detecting range; 2. Good velocity estimates; 3. Provides nearly all-weather and broad-area imagery; 4. High depth resolution and accuracy.	1. Skewed data from fast turning manoeuvres depending on the context; 2. Limited small and dynamic target detection capability; 3. Susceptible to high waves and water reflectivity.
LIDAR	1. Good at near range obstacle detection; 2. High depth resolution and accuracy.	1. Angular resolution both vertically and horizontally; 2. Sensitive to environment and USV motion.
Sonar	1. No visual restrictions; 2. High depth resolution and accuracy.	1. Limited detection and range in each scan; 2. Impressionable to the noise from near-surface.
Visual sensor	1. High lateral and temporal resolution; 2. Simplicity and low weight in practical application.	1. Low depth resolution and accuracy; 2. Challenge to real-time implementation; 3. Susceptible to light and weather conditions.
Infrared sensor	1. Applicable for dark conditions; 2. Low power consumption.	1. Indoor or evening use only; 2. Impressionable to inference and distance.

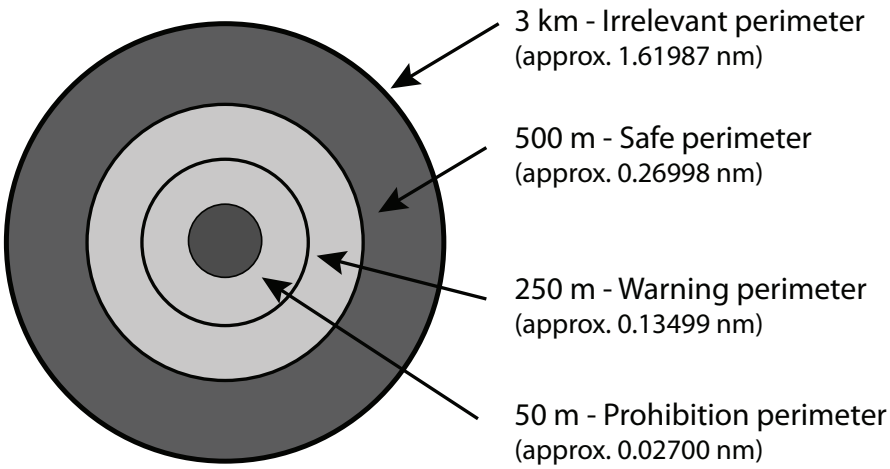
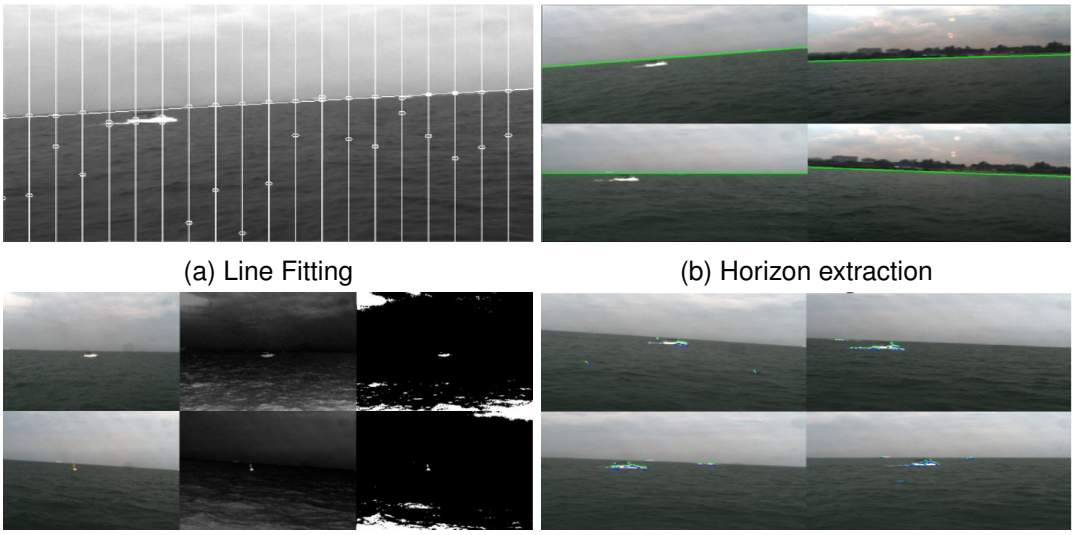


Figure 2.3: The proximity of the USV (Almeida et al. 2009).

targets in terms of a collision threat. A set of perimeters (Figure 2.3) is defined around the USV: irrelevant (3 km), safe (500 m), warning (250 m) and prohibition perimeter (50 m). Based on the Closest Point of Approach (CPA), the estimated shortest distance between the detected object and the USV, targets are classified as no threat, low threat, potential threat and dangerous. Low-cost radars are also used in Schuster et al. (2014) to compensate for the lack of an automatic identification system. Despite the fact that



(a) Line Fitting (b) Horizon extraction

(c) Saliency detection (d) Corner extraction

Figure 2.4: Monocular obstacle detection (Wang et al. 2011).

Figure 2.5: The obstacle is tracked among successive frames (Wang et al. 2011).

radar has been extensively used for far-field obstacle detection, its accuracy is low for short distances. It is then difficult to detect changes in course in time. Moreover, high waves and water reflectivity still represent a major challenge.

Other approaches use vision methods for recording obstacles near the vessel. For example, Wang et al. (2011, 2012) use two parallel cameras. In the first monocular phase, the horizon is extracted using the Random Sample Consensus (RANSAC) (Fischler & Bolles 1981) method (Figure 2.4a), a binary mask is built (Achanta et al.

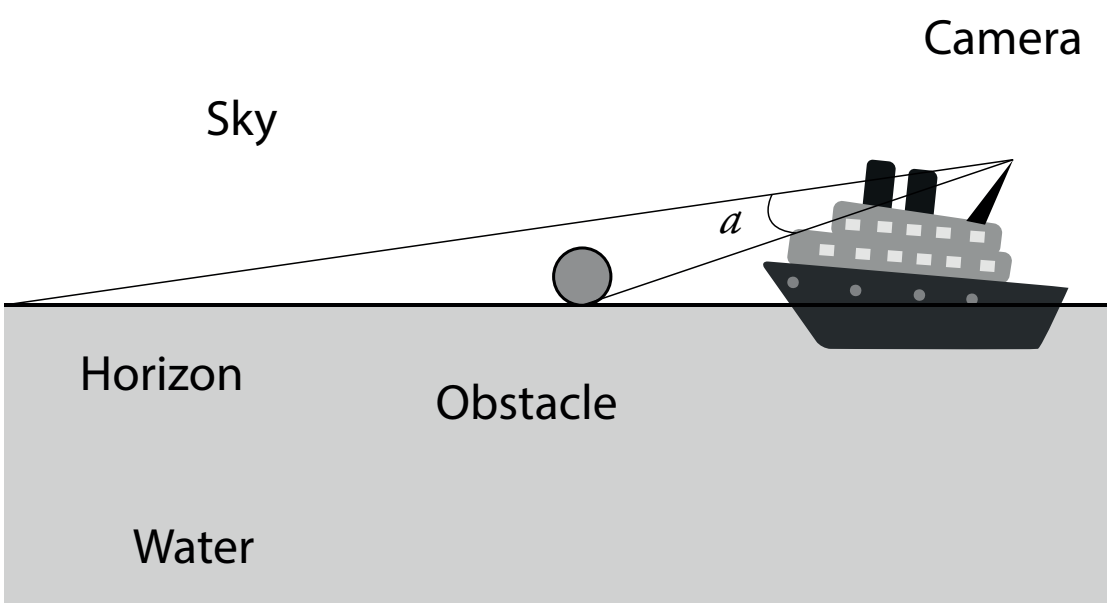


Figure 2.6: Estimation of the USV-obstacle distance (Azzabi et al. 2014).

2009) and detected salient features are considered of potential interest (Figure 2.4c). Surface obstacles are distinguished (Harris & Stephens 1988, Bouguet 2000) (Figure 2.4d) and interesting one are tracked in consecutive frames to validate them as potential obstacles (Figure 2.5). Stereo correspondence was then realised applying an epipolar constraint to reduce the search space. This approach showed good results up to 100 m from the USV using a low-resolution image. Monocular greyscale images are used in Azzabi et al. (2014). The Sobel operator and the Hough transform are applied to extract the edges (Maini & Aggarwal 2009, Hough 1962), then the horizon is identified and moving objects are detected using optical flow estimation. Finally, the distance to obstacles is estimated using geometric relationships (Figure 2.6). Using greyscale images, the authors reduce the computational effort required by other algorithms (e.g., Ettinger, Nechyba, Ifju & M (2003)).

Other systems use Infra-Red (IR) cameras due to their ability to discretise temperatures. The main advantage of IR cameras is the possibility to overcome the impact of various light conditions (darkness or fog), allowing operations in day and night to take place with little deterioration in performance. In Borghgraef et al. (2010), floating mines were identified in the Persian Gulf. Two background subtraction algorithms, the

Visual Background extractor (ViBe) (Barnich & Van Droogenbroeck 2011) and the behavioural subtraction (Jodoin et al. 2008), were implemented. Without specifying any details about the target object, this approach detects every anomaly in respect to the background. IR data are also used in Park & Jeong (2012), where a multi-step algorithm extracts the shape of a floating object. Vertical and horizontal edges are initially extracted to find candidate objects. Then, the background is removed and a logical AND operation with the original frame is performed to highlight the object region. The limit shown by the previous approaches relies on detecting small obstacles at long distances but, unfortunately, IR applications are still very limited. For this reason, it has not been proved that marine situations such as sea fogs, wave occlusions, variations of lighting and weather, and object reflections do not affect IR cameras' performances like common video cameras (Liu et al. 2016).

From the consolidated use with UGV, Lebbad & Nataraj (2015) adopt a LIDAR (Laser Direction and Ranging) because a laser entering the water dissipates itself, segmenting any object present there. Sensor data are combined with that of a camera and the object area is then identified. The authors discovered that the vessel's motion can distort the LIDAR image and some of the targets may appear tilted. A similar approach was proposed in Halterman & Bruch (2010) where sensor data were analysed with regards to different parameters (i.e., scanning rate and density, coverage in azimuth, elevation and range etc.). It was noticed that most of the obstacles appear clearly in the returned data while the sea surface does not provide any return. On the other hand, problems are posed by the presence of salt spray, winds, waves, currents and tides that produce image blurring and vibrations (Gal & Zeitouni 2013).

2.2.2 Path planner

After perception, a representation of the world is realised. This can be done in a 2D or 3D space but the first option is often preferred because it is less computationally demanding (even though the depth dimension can also be useful for navigation purposes). As stated in Section 1, the Path Planner (PP) is usually subdivided: the global

PP identifies a free path from the actual position of the robot to a destination, while the local PP tries to avoid close moving obstacles.

In the following subsections, the most recent techniques used in marine robotics are described. Before that, certain navigation rules are introduced.

2.2.2.1 The International regulations for avoiding collisions at sea

While underway, vessels must obey rules designed to avoid any kind of collision. Despite being manned or not, the rules must be obeyed by any kind of vessel operating at sea (Wilson et al. 2003, Lee & Kim 2004, Kemp 2002, Belcher 2002). Differing, confused behaviour can lead to collisions with other marine craft. The COLREGS (Commandant 1999) an abbreviation for the International Regulations for Preventing Collisions at Sea, are divided into three parts. Part A defines vessel and authority responsibilities, Part B regulates the conduct of vessels in an encounter, and Part C establishes communication protocols. Rules contained in Part B are defined as *Steering and Sailing Rules*, and the more important are reported in Table 2.2.

The rapid increase in the number of USVs posed interesting questions about their responsibilities while underway (Allen 2012). This is still an open problem, the US Navigation Safety Advisory Council seems inclined to recognise their status as 'vessels' and, therefore, their obligations in an encounter.

2.2.2.2 Global path planner

A global path planner must continuously adapt the existing path to new long-range obstacles. Larson et al. (2007a,b), discretise the environment in a bi-dimensional grid. Stationary obstacles are provided by a chart server, and moving ones by the radar. The A* algorithm (Hart et al. 1968) was chosen as the search technique and a proximity cost added to prevent the USV going close to obstacles. To avoid the moving ones, safe velocity ranges are determined using the Velocity Obstacles (VO) method (Figure 2.7): a velocity space $v - \theta$ grid (where v identifies the vehicle's speed and θ its heading angle) is built where the obstacle's size is increased and the robot treated as a point.

Table 2.2: Selected Rules from the International regulations for avoiding collisions at sea (Commandant 1999)

Number	Situation	Action
13	Overtaking	Any vessel overtaking any other shall keep out of the way of the vessel being overtaken.
14	Head on	When two power-driven vessels are meeting on a reciprocal course so as to involve risk of collision, each shall alter her course to starboard so that each shall pass on the port side of the other.
15	Crossing	When to power-driven vessels are crossing so as to invoke risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel.
16	Give way	Every vessel which is directed to keep clear of another vessel shall, so far as possible, take early and substantial action to keep well clear.
17	Stand on	Where one of two vessels is to keep out of the way, the other shall keep her course and speed. The latter vessel may, however, act to avoid a collision by her manoeuvre alone, as soon as it becomes apparent to her that the vessel required to keep out of the way is not taking appropriate action in compliance with these rules.

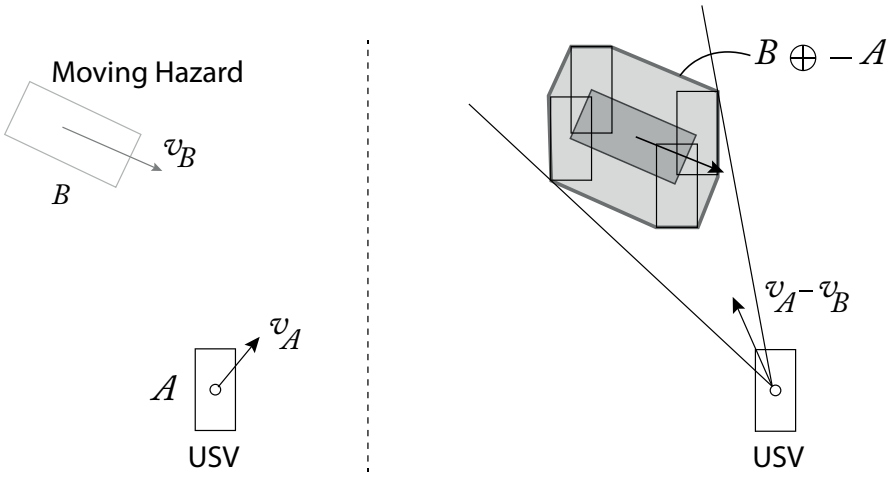


Figure 2.7: To avoid the collision, the relative velocity $\vec{v_A}$ - $\vec{v_B}$ has to be outside the cone defined by the robot centre and the expanded obstacle $A \oplus B$.

To avoid any collision, the robot's velocity must lie outside the VO. If a collision cannot be avoided, a Projected Obstacle Area (POA) (Figure 2.8), the future occupied area, is created for each obstacle and a new safe route is found. The authors also implemented the COLREGS, in particular rules 13, 14 and 15.

Casalino et al. (2009) suggest an approach based on the Visibility Graph (VG) concept (Figure 2.9): for each couple of inter-visible points, a straight line connecting them and not passing into an obstacle was drawn. After transforming obstacles into polygons,

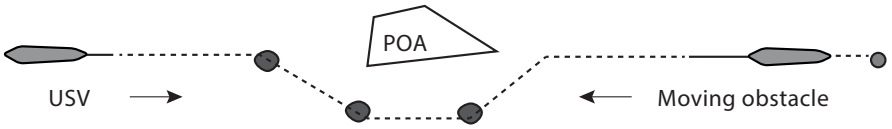


Figure 2.8: The projected obstacle area of a moving obstacle.

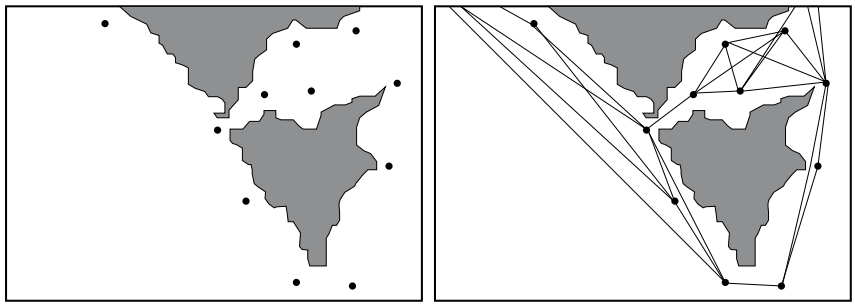


Figure 2.9: A visibility graph.

the VG is built and Dijkstra's Algorithm (Dijkstra 1959) was applied to find a safe path.

A different solution has been proposed by Xie et al. (2014)who modified the Artificial Potential Field (APF) (Khatib 1985) method (Figure 2.10) in a way that, in the presence of an obstacle, attractions decrease as a linear function and repulsions as a higher-order function. Chen et al. (2013) propose to adapt the scanning angle of a sonar to the distance from the obstacle using fuzzy logic to make the strategy timely and effective. Four scan range levels are defined: if no obstacle is detected in a cycle, the scan range will jump to the upper level, otherwise, it will move down to the nearest level bigger than the distance existing between the sonar mounted on the vessel and the identified obstacle. At this point, the type of the obstacle is determined and the slope is calculated. The only limitation is given by the low precision of the Inertial Measurement Unit (IMU) used, which compromised the reaction accuracy.

Chang et al. (2003) extended the maze routing algorithm developed by (Lee 1961) to general λ-geometry for $\lambda \geq 4$. A route is defined as a data structure composed of the vessel path, the source cell, the destination cell and the speed. The vessel path is a list of all the cells traversed, characterised by their coordinates, the arrival time, and a

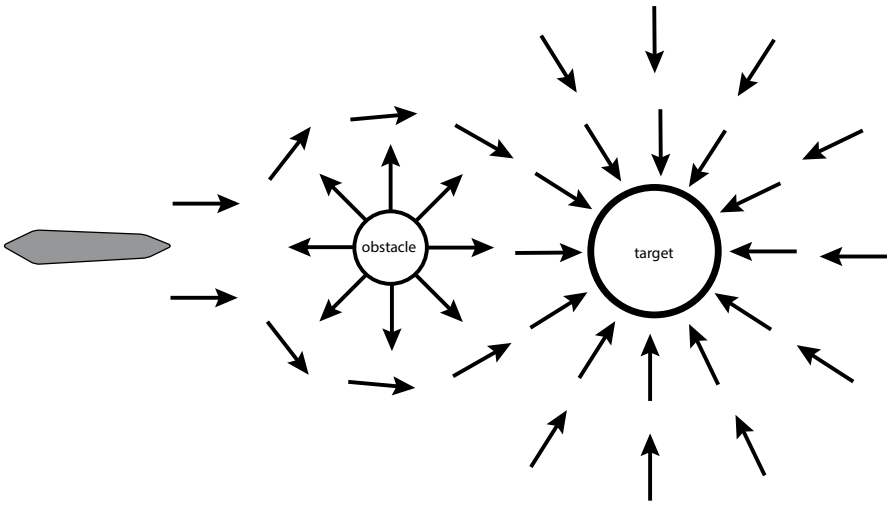


Figure 2.10: The APF model.

link to the next one. Unfortunately, the shortest path on a raster grid is not always the optimal one since it could contain too many course alterations. Moreover, a change in speed often corresponds to a longer passage time. This problem has been addressed by Szlapczynski (2006), which found the shortest path with bend penalisations. The data structure of the previous work is modified to account for the cost of all course alterations.

2.2.2.3 Local path planner

One example of an LPP is given by Kuwata et al. (2014), where the COLREGS and VO are considered. Initially, the CPA for every obstacle was calculated and the most suitable rule is applied. A cost for every velocity v_j and heading angle ω_j admissible was generated and the (v_j, ω_j) pair with the minimum cost sent to the controller. Similarly, Leng et al. (2013) integrated VO with mixed linear integer programming after linearizing the USV's properties, sensor data and uncertainty about the environment. The collision risk is checked by calculating the CPA and its distance from the vessel.

In Larson et al. (2007a,b) the Morphin algorithm (Simmons & Henriksen 1996) is applied, drawing multiple arcs in front of the vehicle over the local occupancy map. Thus, all the safe paths are considered, covering every free cell of the map with at least one arc. A weight is assigned to each arc depending on its vicinity to obstacles. The votes

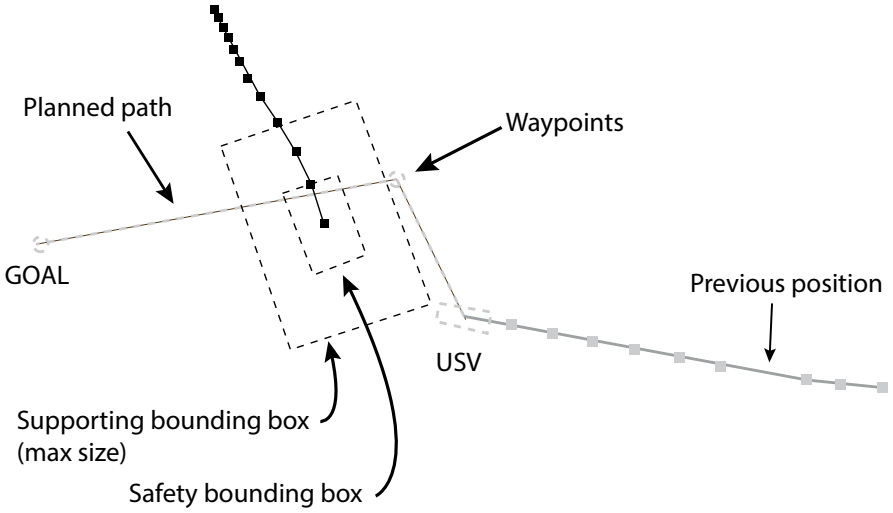


Figure 2.11: The bounding boxes of a moving obstacle (Simetti et al. 2014).

are then scaled from 0 to -1 and combined with those coming from other navigation behaviours.

Casalino et al. (2009) describe a method based on the Bounding Box (BB) concept, defined as the rectangular area the vessel should avoid. The algorithm integrates the BB vertices with the vehicle position S and the goal G into a graph. The solution is any path from S to G, identified using A*. Since kinematic constraints are neglected, the work suffers from sub-optimality. In successive work (Simetti et al. 2014), a safety BB is added and all the computations are now performed against it; after entering the safety box, the USV must leave it without intercepting the main diagonals and ensuring in this way the collision BB is not crossed. To address uncertainty, a supporting BB is adopted (Figure 2.11), whose dimensions may vary from coinciding with the safety ones to be identical to the maximum bounding box's ones. Thus, the path's robustness to changes in speed and heading of the obstacle is increased.

An approach based on lane-constrained trajectory is proposed in Tan et al. (2010). Here, while avoiding obstacles, the vessel must meet several objectives such as maintaining a minimum distance from each obstacle, respecting the COLREGs, keeping as close as possible to the intended path and complete the trajectory in the shortest time possible. Initially, the platform's motion is forward simulated for a fixed number of time

steps using simple models of a manoeuvre tracker and the vessel. Then, objectives are ordered and an elimination multi-stage process takes place. Regrettably, the algorithm may sometimes be unable to find a solution since only a sample of possible manoeuvres is generated.

Blaich et al. (2015) modified the A* algorithm to allow velocity variations and different turning circles. The cost function is adapted adding a penalty representing the amount of path skipped during the evasion manoeuvre. In this way, the USV is led back to the original path after the deviation. Velocity and time are added to the search space to guarantee the feasibility of the trajectory.

Tang et al. (2012) developed a new method called Obstacle Avoidance Algorithm Based on a Heading Window (OAABHW). To obtain the best navigation angle λ_{out}, the heading yaw is considered as an optimisation objective, with the heading window and the set of infeasible heading angles as constraints. From λ_{out} and the current heading angle λ_{USV}, it is possible to obtain the avoidance rotational velocity ω_{out}. Zhang et al. (2014) illustrated a new adaptive collision avoidance algorithm based on State-Action-Reward-State-Action (SARSA) on-policy reinforcement learning (Sutton & Barto 1998) to face changing in trajectory due to external conditions. It is composed of two modules: the Local Obstacle Avoidance Module (LOAM) and an Adaptive Learning Module (ALM). LOAM focuses on the avoidance manoeuvre, ignoring external disturbance factors, dealt with instead by ALM while searching for a course compensation angle. LOAM calculates the guidance angle with OAABHW, which is forwarded as input to the ALM together with sea winds and currents.

2.2.2.4 Intelligent path planner

Differing from the deterministic methods proposed before, some research groups have developed intelligent algorithms based on Fuzzy Logic (FL), Evolutionary Algorithms (EAs) and Artificial Neural Networks (ANNs). These approaches offer multiple advantages such as lower computation effort and the ability to learn the near-optimal solution. On the other hand, they can fail by getting trapped in local minima or fail to find a so-

lution at all. To overcome these problems, they are often combined in 'hybrid systems' (Campbell et al. 2012).

Hwang (2002) used a knowledge base and an fuzzy inference engine. Adopting a circular ship domain, the author solves a multi-ship collision one by one, suggesting the manoeuvres as a visual aid or sends them to an autopilot. Liu et al. (2006) coupled FL with an ANN in a three modules algorithm. The first module classifies the collision risk given CPA, course, and distance of the incoming vessel and provides an avoidance manoeuvre. The second calculates the membership function of the speed ratio. The last one generates magnitude and action time. Kao et al. (2007) addressed port navigation safety using FL to calculate a Guarding Ring (GR) and a Danger Index (DI). The radius of the GR is established starting from the length of the ship, its speed, and the sea state. When two GRs start to overlap, the collision alert system is activated. Perera et al. (2012, 2015) and Lee et al. (2004) provide an example where FL has been combined with a Bayesian network and an APF, respectively.

EAs are heuristic approaches developed to find a close-to-optimal solution in large search spaces (Holland 1975). They follow the Darwinian principle that only the best elements of every generation survive a process of variation and selection (Darwin et al. 1996). Smierzchalski (1999) and Smierzchalski & Michalewicz (2000) use genetic mutation operations to modify a ship's speed. COLREGS and the time variable are incorporated during the generation of a new population of trajectories and the solution is searched based on a cost function combining space, time and smoothness. In this model, based on (Ito et al. 1999), each chromosome has a different number of genes representing the coordinates of the turning points, the ship's speed, and the interconnections among genes of the same chromosome. A more realistic approach is explained by Zeng (2003*b*),where a single gene is characterised by some noise parameters, such as wind, wave and sea current. The length of each chromosome depends on the navigation (Figure 2.12). The same approach is also used by Tsou (2010). The selection of the best route is made using the roulette selection method, after ordering

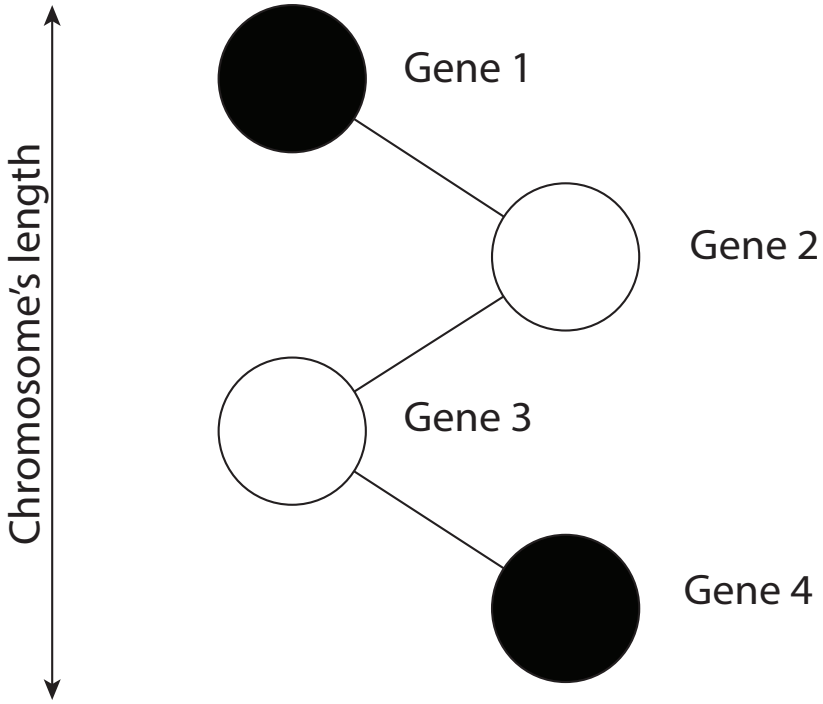


Figure 2.12: The route as a chromosome (Zeng 2003*a*).

all of them per a linear fitness function. In Szlapczynski (2013, 2015), evolutionary algorithms were used to find a set of safe trajectories for all ships involved in a collision. Six types of violations were identified and corresponding penalties introduced. Ship tracks, modified in course and speed, are therefore evaluated with a fitness function inversely proportional to penalties.

In contrast, in Lazarowska (2015) the path planning problem is solved as an Ant Colony Optimisation (ACO) problem. ACO is inspired by the indirect communication mechanism observed among ants. A graph considering all the constraints and the ship position is built. At this point, every ant constructs its path from the starting vertex until the ending one: at each step, the ant chooses its next position based on the value of the pheromone trail. After all the ants complete the task, the pheromone trail is updated penalising those vertices with a constant value and increasing the others. The algorithm terminates when a maximum number of iterations is reached or after a maximum

computational time. The path selected is shown to be the shortest.

2.3 Discussion

The review presented at the beginning of this chapter began by comparing different sensors employed for acquiring information about the environment. Classic sensors used in the marine environment such as sonar and radar, as well as those mounted on a ground vehicle, i.e. cameras and LIDARs, have been evaluated to establish how good they are in identifying static and moving obstacles. Studies involving them are reported and their respective advantages and drawbacks have been shown. It is concluded that the radar, despite being the sensor most used for preventing a collision at sea in the past, suffers from a lack of accuracy while trying to identify objects at close distance. Vision-based (mono and stereo) approaches can achieve better results than radar despite maritime phenomena like lightning changes and waves still posing a big challenge in the field. Infra-red cameras can offer a partial solution to the previous problems but only a few works have employed them thus far. Many efforts have been made to adopt the use of LIDAR from unmanned ground vehicles but, despite calibration problems now being solved, the same issues for visual sensors still hold. Therefore, it is possible to conclude that the obstacle detection problem remains open and a definitive robust solution is still missing.

Multiple path planners have been reported, classifying them as global or local ones. A stand-alone section has been dedicated to the so-called 'intelligent' methods, based on soft computing techniques. ANNs have been applied to this problem due to their non-linear mapping, learning ability and parallel processing. Fuzzy logic, instead, can simulate the human thinking represented by knowledge-based conditional rules. All the methods described must face the difficulty of addressing complex encounter scenarios that would require human-like experience to choose the best action. Considering that most of the time they work as a Black-box (e.g., ANN), the convergence to the optimal solution is not guaranteed. In addition, these techniques are computationally time-consuming. Therefore, they should be discarded while considering real-time navigation

but they can be adopted in different tasks where they have already proved to be efficient (e.g., convolutional neural networks for identifying possible obstacles in front of the vehicle). A major problem with most of the solutions reported is that they have been tested only in the simulation while proving their validity in the real world still has to be done. Moreover, their reliability is limited by the results that can be incomplete due to a small subset of scenarios tested, or small for a proper probabilistic analysis (Niu et al. 2016). It is therefore plausible to affirm that there is no guarantee of avoiding obstacles in all conditions with the methods above. In addition, three serious shortfalls have been identified:

- while defining the path, most of the methods reported do not consider the vessel's dynamics;
- others ignore the COLREGS despite this being a mandatory requirement while underway in open waters;
- almost all of them do not take into account weather and sea conditions.

Taking these points in order, most of the path planners previously described produce un-executable paths. Treating the USV as a point and ignoring its dynamics (e.g., the minimum turning radius of the vehicle), they generate way-point paths in which two points are generally connected by a straight line. This results in a trajectory characterised by high turning rates with the possibility of damage to the vessel's actuators. To avoid this problem, the solution is to incorporate the vessel's dynamics (in terms of velocity, drag and heading angle) within a cost function in the formulation of the path to making it suitable for the USV's turning abilities.

The second main point to address is the implementation of the COLREGS. This is an important requirement for all types of USVs since they must show intelligent behaviour for understanding and executing the standard rules for marine navigation. There is not a universally agreed solution to incorporate them and most of the existing works either disregard them or adopt different safety domains to emulate them. In a few cases,

they are included in the solution space but the optimality is not guaranteed. Therefore, to simplify the decision-making process in an encounter, it has been decided that an autonomous vessel always has to give away if a doubt arises while meeting a manned vehicle.

Finally, future work should address environmental disturbances and uncertainties. USV path planning is dependent on the combined effect of wind, waves and currents, where wind and wave effects can be neglected for larger vessels (Lee et al. 2015). High speed currents can deviate the vehicle from the planned path instead. The incorporation of such environmental effects within the planning process, while improving the overall understanding of the surrounding world increases the computational complexity for the optimal path planners. In the same way, varying environmental conditions (e.g, fog, lighting, rain, wave occlusions, sophisticated background) as variations in the view angle and range, still represent an issue for real-time vision-based perception.

In this sense, the work presented in this thesis can play an important role. UAVs are becoming even cheaper, attracting a lot of interest from researchers all around the world. The applications involving UAVs range from scientific exploration and data collection (Shim et al. 2005, Bourgault et al. 2004, Neumann et al. 2013), to commercial services, military reconnaissance and law enforcement (Murphy & Cycon 1999, Colorado et al. 2017), search and rescue (Tomic et al. 2012, Kruijff et al. 2014), patrolling (Minaeian et al. 2016) and even entertainment (Kim et al. 2018). Some works have already been done in combining them with unmanned ground vehicles in multi-agent system to solve a wide range of operation, but only a couple of projects involve their presence in the maritime environment, where they can be deployed for multiple applications such as disaster monitoring (Murphy et al. 2008), coastal surveillance (Pereira et al. 2009, Pinto et al. 2013) and wildlife monitoring (Linchant et al. 2015, Watts et al. 2012). The innovation of using a UAV is given by its flexibility because it can look far ahead from the USV: in this way, obstacles moving far away from the vessel can be detected and a new path for the USV can be planned a second or multiple times while

considering new piece of informations. From a practical point of view, in situations in which the UAV is unable to operate, the cameras mounted on the vessel allow it to continue the task even in the absence of the GPP, acquiring the proper data for creating a local area model used by the LPP. Data acquired from these sensors do not represent only a backup plan but they are also used during all the navigation phase together with those coming from the UAV (e.g. an infra-red camera can allow the vessel to navigate also in foggy conditions), in order to increase the awareness of the environment. Nevertheless, these shortcomings offer interesting challenges which need to be overcome in the future.

Flying a UAV in the marine environment encounters rough and unpredictable operating conditions due to the influence of wind or wave in the manoeuvre compare to land. Apart from the above, there are various other challenges associated with the operation of UAVs. For example, the inaccuracy of low-cost GPS units mounted on most UAV and the influence of the electrical noise generated by the motors and on-board computers on magnetometers. In addition to this, the estimation of the USV's movements is a difficult task due to natural disturbances (e.g. winds, sea currents etc.). This poses difficulty for a UAV to land on a moving marine vehicle with a low quality pose information.

This thesis represents an important step towards building the multi-agent system described above. In the following chapters, it will be described two novel solutions developed for achieving autonomous landing of a UAV, in particular on the deck of an unmanned surface vehicle under simulated marine conditions. To overcome the issues stated above, the camera mounted on the UAV and commonly used during surveillance mission (Stacy et al. 2002), can also be used to increase the accuracy of the relative-pose estimates between the aerial vehicle and the landing platform (Ettinger, Nechyba, Ifju & Waszak 2003). The adoption of fiducial markers on the vessel's deck is proposed as solution to further improve the estimate results.

2.3.1 Autonomous landing for UAV

Autonomous landing is so far one of the most demanded features by UAV pilots. Despite this, it is still one of the most challenging aspects of this kind of vehicle. In order to compare the proposed methods with the existing approaches, a brief background on the existing techniques is now offered.

Before describing the literature there are three aspects to take into account in landing: control, pose estimation, navigation. Control represents the low level layer and consists of integrating the data from the sensors (GPS, IMU, altimeter, etc) in order to keep the vehicle around the desired set-point. The pose estimation is strictly related to control and consists in using sensors and visual feedback to estimate the position of the UAV with respect to a reference frame, which is the landing pad in this case. Finally the navigation: once the pose of the vehicle has been estimated, it is necessary to create and adjust a trajectory for reaching the target point. For the purpose of this thesis, this section only focuses on literature concerning pose estimation and navigation.

In order to help the reader to understand the state of the art, the existing methods are grouped in three major areas: sensor-fusion, device-assisted, and vision-based. Combining different data from multiple sensors is a common expedient used to improve the performances. This is the main strategy used in sensor-fusion systems. In the recent work (Forster et al. 2015), the data from a downward-looking camera and an inertial measurement unit were combined in order to build a three-dimensional reconstruction of the terrain. Given the two-dimensional elevation map was possible to find a secure surface area for landing. In Saripalli et al. (2003) the authors combined vision with differential GPS for estimating the relative pose of the UAV with respect to the helipad. In this work, geometric invariant extraction is used to identify an H-shape landing pad. The same technique was also applied in a successive work (Saripalli & Sukhatme 2006) in which the UAV landed on a moving target using a Kalman filter. In Herissé et al. (2012) the measurements of optical flow, gyroscopes and accelerometers were combined to increase the robustness of a vision system for hovering and landing a

flying vehicle on a moving target. Additional research based on optical flow for tracking and approaching a landing area has been undertaken in Cesetti et al. (2009) and Ruffier & Franceschini (2015). A ground-based multisensor fusion system has been proposed in Zhou et al. (2015). The system included a pan-tilt unit, an infra-red camera and an ultra-wideband radar used to centre the UAV in a recovery area and guide it toward the ground. There are two main problems in sensor-fusion approaches. First the sensor noise, which may be extremely high in an unstructured environment. Second, the unavailability of navigation information and/or improper sensor feedback. For instance in remote areas it is not possible to use GPS to adjust the trajectory.

On-board and ground devices have been widely adopted to increase the precision of marker identification. A system based on infra-red lights has been used in (Gui et al. 2013). The authors adopted a series of parallel infrared lamps placed in a runway. The camera on the vehicle was equipped with optical filters for capturing the infra-red lights and the images were forwarded to a control system for pose estimation. A Chan-Vese approach supplemented through an extended Kalman filter has been proposed in Tang et al. (2016) for ground stereo-vision detection. In Kim et al. (2014) an omni-directional camera was used to enhance the field of view of a UAV and to identify a red marker. In Yakimenko et al. (2002) an on-board infra-red camera was used to identify three known ground points and estimate their distance to the UAV. Infra-red ground spots are another expedient used to help the detection of the marker. For example, in Wenzel et al. (2010) the authors used an algorithm for detection and tracking them with a downward looking camera. Other devices commonly used are distance measurement sensors, such as Laser Imaging Detection and Ranging (LIDAR). In Theodore et al. (2006) LIDAR data was fused with altitude measurements and feature-tracking data for estimating the UAV pose with respect to a fixed point in a selected landing area. Similarly, Scherer et al. (2012) used the LIDAR to identify and validate landing zones for a full-scale helicopter. The limits of these approaches are enumerable. First of all, the use of dedicated hardware both on the ground and on the vehicle is not always possible. Many commercial UAVs have limited hardware resources or limited payload

capacity, which means that it is not feasible to use specific on-board devices. Second, some of these devices are expensive (e.g. LIDAR) and it is not possible to justify their use in commercial products.

Vision-based systems primarily rely on camera images for pose estimation and planning. These methods identify the ground marker and extract relevant features which allow for estimating the vehicle's pose. This data is then used within a control loop for approaching the pad. A method based only on a monocular camera has been proposed in Lange et al. (2009). The system used a well defined target pattern, easy to identify at different distances. Having a series of concentric circles, it was proved to be possible to find the landmark also when partially occluded. A modified version of the international landing pattern has been used in Lin et al. (2017). The solution adopted used a seven-stages vision algorithm to identify and track the pattern in a cluttered environment and reconstruct it when partially observable. The use of AR-tag fiducial marker has been taken into account in Falanga et al. (2017) and Vetrella et al. (2018). In both cases a precise pose estimation has been done using only an onboard camera. In Lee et al. (2012) a vision-based visual servoing algorithm has been used to track a moving platform and to produce velocity commands for an adaptive sliding controller. In Sereewattana et al. (2014) the authors used Histogram of orient gradients (HOG) feature extraction and a pair of images taken at a known interval to estimate the distance between the vehicle and the marker. The adoption of a specific design for the pad (e.g. H-shape) allows performing corner detection and labelling after proper image binarization (Shi & Wang 2009). In Shakernia et al. (1999) the vision problem is addressed as an ego-motion estimation one, estimating the altitude of the vehicle in relation to a fixed planar surface. In this way vision is used in the feedback loop as a state observer for the landing controller. Vision-based systems tend to use only limited hardware for control and planning, this is advantageous for a commercial vehicle which rely only on limited resources. However, most of these approaches are not particularly robust to image noise and may have problems when the pad is distant, partially occluded or blurred.

The methods previously described demonstrate different limitations and these are now discussed. Sensor-fusion methods rely on information coming from expensive sensors that most of the time are not present on-board of low-cost UAVs. In addition, they mainly use a GPS signal that may be unavailable in real-world scenarios. The device-assisted methods usually provide an accurate estimate of the UAV's pose. However, they rely on external sensors that may be expensive and not always available. Finally, vision-based approaches use images acquired by on-board sensors, namely cameras, for control purposes. Even though this represents an invaluable advantage over other approaches, they may fail when trying to identify low-level features in distorted images. The present work tries to solve the aforementioned problems.

2.3.2 Deep reinforcement learning

The existing work in this field is based on hand-crafted features extraction and external sensors to identify the landing pad, usually represented by an high-contrast marker. In this thesis, a completely different solution is described, based on the recent interest born around the Deep Reinforcement Learning (DRL) and the Deep Q-Network (DQN) architecture (Mnih et al. 2015).

The use of reinforcement learning in robotics is not novel, for example it has been widely used to achieve a robust walking gait (Peters & Schaal 2008) and in manipulation tasks (Cui et al. 2017). For a recent survey please refer to Kober et al. (2013).

However, work using reinforcement learning for UAV control are quite limited. The main problem is related to an extremely large continuous control space and the unrealistic assumption of a noise-free state-space; however, the present literature offers some successful applications and these will be discussed. For example, in Bagnell & Schneider (2001) a policy search method has been used for controlling a small helicopter. In a similar way, Ng et al. (2006) taught a mini helicopter to perform acrobatic manoeuvres, such as flying in a reverse status. In Zhang et al. (2016) reinforcement learning is combined with model predictive control to train an obstacle avoidance policy which is allowed to access only sensor readings and not the full state of the system. In

all these cases, the policy was intended to be a low level closed loop controller instead of the high level navigator considered in this work.

The possibility to couple deep neural networks with reinforcement learning has recently opened new insights in a variety of domains. Since 2011, deep neural networks (DNNs) represent the state of the art in a variety of tasks, such as speech recognition (Fayek et al. 2017), head pose estimation (Patacchiola & Cangelosi 2017), and scene segmentation (Pavel et al. 2017) (for a review see (Schmidhuber 2015)). The use of DNNs as Q-function approximators allowed extending reinforcement learning to a new series of problems. In Levine, Finn, Darrell & Abbeel (2016) a guided policy search method has been used to train a convolutional neural network for tasks that require close coordination between vision and control, such as screwing a cap onto a bottle. A deep network has been used by Levine, Pastor, Krizhevsky, Ibarz & Quillen (2016) to learn hand-eye coordination for grasping in a robotic manipulator. Mnih et al. (2016) propose an asynchronous variant of the actor-critic architecture that obtained the best score in the Atari domain and applied to a variety of continuous motor control problems as well as on a new task of navigating random 3D mazes. More recently, improved algorithms such as Deep Deterministic Policy Gradients (Lillicrap et al. 2015), Trust Region Policy Optimization (Schulman et al. 2015) and Proximal Policy Optimization (Schulman et al. 2017) achieved even better scores on the classic benchmark suite. Despite the recent effort in extending DRL to different domains, at the moment there is still no consolidated literature on UAV control and planning.

2.4 Conclusion

This chapter offered a detailed overview of the current state-of-the-art techniques for path planning with a single unmanned surface vehicle. Planning a collision free path is definitely a complex task when a proper representation of the environment (with static/moving obstacles) is missing. Once this representation is obtained through classic mapping approaches, the global path planner identifies a collision free path connecting the actual position of the vehicle with its final destination. In the situation a possible

collision is identified, the local path planner must find an evasive manoeuvre to avoid the collision in the first instance and then lead the vehicle on track. Despite the efforts currently done in research, there are still some important problems to solve and an optimal solution is far from being found. The realisation of a multi-agent system involving a UAV to overcome the sensors' limitation of the USV is proposed as a potential contribution to the field of path planning for USV. In a scenario where a UAV is flying above the USV, mapping the surrounding environment and updating its representation, the landing of the first vehicle on the deck of the second represents the most challenging task due to the high precision required. Therefore, the following chapters describes two novel solutions for achieving autonomous landing of a UAV on a mobile platform, in particular the deck of an unmanned surface vehicle.

Chapter 3

Methods and Materials

THE following chapter provides an overview of the methods and the material at the base of the research presented in the rest of the thesis.

3.1 Unmanned aerial vehicles

In the last few years, significant interest has grown towards UAVs, as described in Kumar & Michael (2012). Among different UAVs topologies, helicopter flight capabilities such as hovering or vertical take-off and landing (VTOL) represent a valuable advantage over fixed-wing aircraft.

3.1.1 Vehicle characteristics and hardware overview

The quadcopter in this study is an affordable ($250 USD in 2017) AR Drone 2.0 built by the French company Parrot and it comprises multiple sensors such as two cameras, a processing unit, a gyroscope, an accelerometers, a magnetometer, an altimeter and a pressure sensor. It is equipped with an external hull for indoor navigation and it is mainly piloted using smart-phones and tablets through the application released by the producer over a WiFi network. Despite the availability of an official software development kit (SDK), the Robot Operating System (ROS) (Quigley et al. 2009) framework can be used to communicate with it, using, in particular, the *ardrone-autonomy* package developed by the Autonomy Laboratory of Simon Fraser University, and the *tum-ardrone* package (Engel et al. 2012*a*,*b*, 2014) developed within the TUM Computer Vision Group in Munich. The specification of the UAV is as follow:

- Dimensions: 53 cm x 52 cm (hull included);

- Weight: 420 g;
- Inertial Measurements Units (IMU) including gyroscope, accelerometer, magnetometer, altimeter and pressure sensor;
- Front-camera with a high-definition (HD) resolution (1280x720), a field of view (FOV) of $73.5^\circ \times 58.5^\circ$ and video streamed at 30 frame per second (fps);
- Bottom-camera with a Quarted Video Graphics Array (QVGA) resolution (320x240), a FOV of $47.5^\circ \times 36.5^\circ$ and video streamed at 60 fps;
- Central processing unit running an embedded version of Linux operating system;

The downward-looking camera is mainly used to estimate the horizontal velocity and the accuracy of the estimation highly depends on the ground texture and the quadcopter's altitude. Only one of the two video streams can be streamed at the same time. Sensors data are generated at 200Hz. The on-board controller (closed-source) is used to act on the roll (Φ) and pitch (Θ), the yaw (Ψ) and the altitude of the platform (z). Control commands u = (Φ, Θ,Ψ, z) $\in$ [-1,1] are sent to the quadcopter at a frequency of 100Hz.

3.1.2 Navigation, guidance and control

While defining the UAV dynamics model, the vehicle must be considered as a rigid body with 6 degrees of freedom (DOF) able to generate the necessary forces and moments for moving (Nonami et al. 2010). The equations of motion are expressed in the body-fixed reference frame $\mathscr{B}$ (Goldstein 1980):

$$\begin{cases} m\dot{V} + \Omega \times mV = F \\ J\dot{\Omega} + \Omega \times J\Omega = \Gamma^b \end{cases} \tag{3.1}$$

where $V = [u, v, w]^T$ and $\Omega = [p, q, r]^T$ represent, respectively, the linear and angular velocities of the UAV in $\mathscr{B}$. F is the translational force combining gravity, thrust and

other components, while $J \in R^{3\times3}$ is the inertial matrix subject to F and torque vector Γ^b.

The orientation of the UAV in the air is given by a rotation matrix R from $\mathscr{B}$ to the inertial reference frame $\mathscr{I}$:

$$\begin{aligned} R &= R_\psi R_\theta R_\phi \\ &= \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \end{aligned} \tag{3.2}$$

where $\eta = [\phi, \theta, \psi]^T$ is the Euler angles vector and $s.$ and $c.$ are abbreviations for $sin(.)$ and $cos(.)$.

Given the transformation from the body frame $\mathscr{B}$ to the inertial frame $\mathscr{I}$, the gravitational force and the translational dynamics in $\mathscr{I}$ are obtained in the following way:

$$\begin{cases} \dot{\xi} = v \\ m\dot{v} = RF^b - mge_3^i \end{cases} \tag{3.3}$$

where g is the gravitational acceleration and F^b is the resulting force in $\mathscr{B}$, $\xi = [x, y, z]^T$ and $v = [\dot{x}, \dot{y}, \dot{z}]^T$ are the UAV's position and velocity in $\mathscr{I}$.

The UAV's body frame follows a right-handed z-up convention such that the positive x-axis is oriented along the UAV's forward direction of travel. Both camera frames are fixed with respect to the UAV's body one but translated and rotated in such a way that the positive z-axis points out of the camera lens, the x-axis points to the right from the image centre and the y-axis points down. The USV's frame also follows the same convention and is positioned at the centre of the landing platform. Finally, it has been defined as a local frame fixed with respect to the world and initialised by the system at an arbitrary location. In Fig. 3.1 the coordinate systems previously described are depicted.

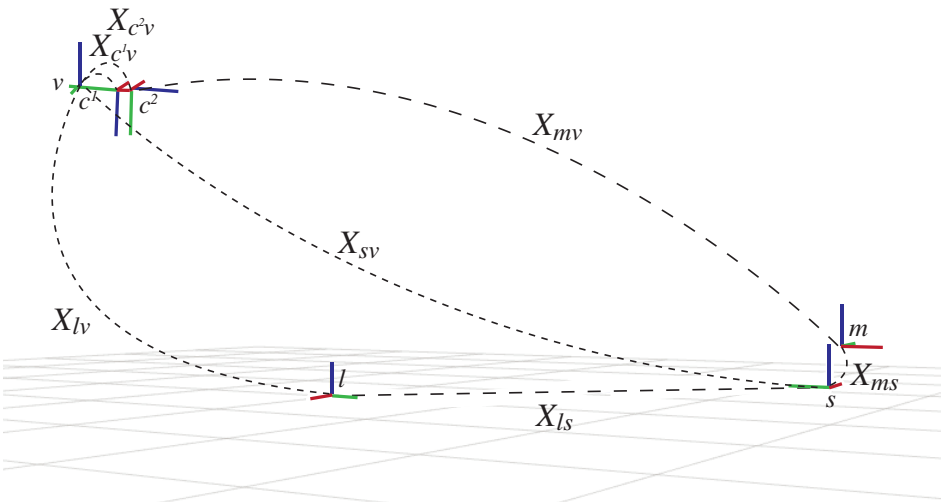


Figure 3.1: Coordinate frames for the landing systems. X_{lv} represents the UAV's pose with reference to the local frame and, in the same way, X_{ls} for the USV. X_{c^1v} and X_{c^2v} are the transformations between the down-looking camera and frontal cameras, respectively, and the vehicle's body frame. X_{mv} and X_{ms} define the pose from the visual marker to the UAV and to the USV, respectively. Finally, X_{sv} is the pose from the USV to the UAV.

The pose of frame j with respect to frame i is now defined as the 6-DOF vector:

$$x_{i,j} = [{}^i t_{i,j}^T, \Theta_{i,j}^T]^T = [x_{i,j}, y_{i,j}, z_{i,j}, \phi_{i,j}, \theta_{i,j}, \psi_{i,j}]^T \tag{3.4}$$

composed of the translation vector from frame i to frame j and the Euler angles ϕ, θ, ψ.

Then, the homogeneous coordinate transformation from frame j to frame i can be written as:

$${}^i_j H = \begin{bmatrix} {}^i_j R & {}^i t_i^T j \\ 0 & 1 \end{bmatrix} \tag{3.5}$$

where ${}^i_j R$ is the orthonormal rotation matrix that rotates frame j into frame i and is defined as:

$${}^i_j R = rotxyz(\Theta_i j) = rotz(\psi_i j)^T roty(\theta_i j)^T rotx(\phi_i j)^T \tag{3.6}$$

3.2 The Proportional-Integral-Derivative (PID) controller

In order to control the drone, a Proportional-Integral-Derivative (PID) controller like the one offered by the tum_ardrone package can be use.

A PID controller usually calculates an error as the difference between a measured variable and a setpoint, and then apply a correction to reduce the error. The correction consists of three terms, namely a proportional, an integral and a derivate one, hence the name. The overall control function is mathematically expressed as follows:

$$u(t) = K_p e(t) = K_i \int_0^t e(t')\,dt' + K_d \frac{de(t)}{dt} \tag{3.7}$$

In the work of Engel et al. (2014), for each of the four degrees of freedom (roll $\bar{\Phi}$, pitch $\bar{\Theta}$, the yaw rotational speed $\bar{\Psi}$ and the vertical velocity $\bar{\dot{z}}$), a separate PID controller is employed. Each of them is used to steer the quadcopter toward a desired goal position $p = (\hat{x}, \hat{y}, \hat{z}, \hat{\psi}) \in \mathbb{R}^4$ in a global coordinate system. The generated controls are then transformed into a robotic-centric coordinate frame and sent to the UAV at 100Hz.

3.3 Computer vision and augmented reality markers

The definition of Augmented reality (AR) markers can be found in (Azuma 1997), "in an AR, virtual-objects are super-imposed upon or composited with the real world. Therefore, AR supplements reality".

In Fig. 3.2, the markers are high-contrast 2D tags designed to be robust to low image resolution, occlusions, rotations and lighting variation.

In order to use this library, the camera calibration file, the marker's dimension and the proper topic's name must be defined inside a configuration file. The package subscribes to the specific video streaming publisher from one of the two fixed cameras. Pixels in the current frame are clustered based on similar gradient and candidate markers are identified. The Direct Linear Transform (DLT) algorithm (Hartley & Zisserman 2004) maps the tag's coordinate frame to the camera's one, and the candidate marker

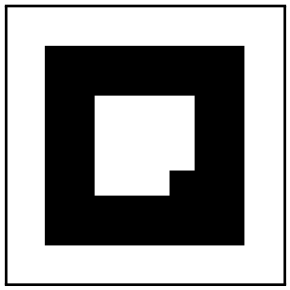


Figure 3.2: The augmented reality marker used in the experiments.

is searched for within a database containing pre-trained markers. The points in the marker's frame and camera's frame are respectively denoted as ${}^{M}P$ and ${}^{C}P$. So, the transformation from one frame to the other is defined as follow:

$$ {}^{M}P = {}^{M}_{C}H\,{}^{C}P = {}^{M}_{C}H^{-1}\,{}^{C}P = {}^{C}_{M}H\,{}^{C}P \tag{3.8} $$

where ${}^{M}_{C}H$ and ${}^{C}_{M}H$ represent the transforms from the marker to the camera frame and vice versa, respectively.

Using the camera's calibration file and the actual size of the marker of interest, the 6-DOF relative-pose of the marker's frame with respect to the camera is estimated.

3.4 Data fusion and state estimation filter

The Kalman filter (KF) (Kalman 1960) is a statistical recursive algorithm that uses a series of noisy measurements, collected in a given time, to generate a relatively accurate estimate of unknown variables. This result is obtained calculating the joint probability distribution over the variables at each time steps. Given this property, it is widely used to learn the internal state of linear dynamic systems. In the particular case in which the system studied is not linear, the extended Kalman filter (EKF) extension has been formulated.

The algorithm is commonly divided into two parts: prediction and update. In the prediction, the KF produce the estimates for the current state, together with the uncertainty. Once a new measurement is observed, the estimates in updated using a weighted av-

erage. Considering the sensor readings, the estimation process satisfies the following equations:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \tag{3.9}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} {\mathbf{F}_k}^T + \mathbf{Q}_k \tag{3.10}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} {\mathbf{H}_k}^T (\mathbf{H}_k \mathbf{P}_{k|k-1} {\mathbf{H}_k}^T + \mathbf{R}_k)^{-1} \tag{3.11}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}) \tag{3.12}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{3.13}$$

where k represents a discrete time instant, $\mathbf{F}_k$ is a kinematic constant velocity model, $\mathbf{H}_k$ is the observation model, $\mathbf{z}_k$ is the measurements vector, $\mathbf{I}$ is an identity matrix, $\mathbf{Q}_k$ is the process covariance matrix and $\mathbf{R}_k$ is the measurement covariance matrix.

3.5 Fuzzy Logic

Fuzzy logic (FL) differs from classical propositional logic because it aims to simulate humans ability of reasoning in a world made of uncertainty and partial information. FL is strictly related to the theory of fuzzy sets, defined as classes of objects without well-defined boundaries. As result, the membership to a fuzzy set is expressed as a continuous value in [0, 1], spanning from "definitely not in the set" through "partially in the set" to "completely in the set" (Zadeh 1965). The main advantage of using a fuzzy logic system is the fact that it does not require a precise model of the system to be controlled but it can solely rely on the empirical knowledge of an experienced user. This knowledge can be modelled by a set of "if [present conditions], then [action to be taken]" rules. The first fuzzy logic controlled was proposed by (Mamdani & Assilian 1975) and it is divided into three steps:

- Fuzzify all input values into fuzzy membership functions.
- Execute all applicable rules in the rule base to compute the fuzzy output functions.

- De-fuzzify the fuzzy output functions to get the correct output values.

3.6 Artificial Neural Network and Deep Learning

The first studies on the artificial neural networks (ANNs) can be dated at the beginning of the forties of the last century, when McCulloch & Pitts (1943) tried to replicate the human brain using a simple neural network based on electrical circuits. A significant theoretical contribution was given by Hebb (1949) who formulated the update rule of neural pathways, saying that they are strengthened each time they are used. This idea saw the light in the first real implementation of ANN, the perceptron of Rosenblatt (1958). This work was further improved by Widrow et al. (1960) who developed ADALINE, a model meant to recognise binary patterns in bits streamed over a phone line. The same authors were the first to propose the idea of a learning procedure, for adjusting the weights of the neural network, based on an error function. But the increasing popularity of ANN was rapidly inverted when Minsky & Papert (1969) proved the existing model, the perceptron, was nothing more than a linear classifier. As a result, the research community was discouraged by continuing investing this model. The interest towards ANN came back at the beginning of the eighties, supported by a joint US-Japan conference on Cooperative/Competitive Neural Networks. It followed that new studies on multiple layered networks started to be published. One of the most influencing ones was written by Rumelhart et al. (1988) and it proposed the *backpropagation* as a learning rule to distribute the error throughout all the layers of the network.

Figure 3.3 shows a graphical representation of an artificial neural network in its standard form, also known as a feed-forward neural network. It consists of multiple neurons arranged in layers and connected through weights. It is called in this way because each neuron in one layer is connected to every other neuron on the immediately next layer. Therefore, it is possible to say that the information is fed forward from the input to the output layer. The working principle is that the input vector x is multiplied by the weight matrix W, and finally summed with a bias vector b. The result z is then subject of a differentiable non-linear activation function $h(z)$ which generates the output of a given

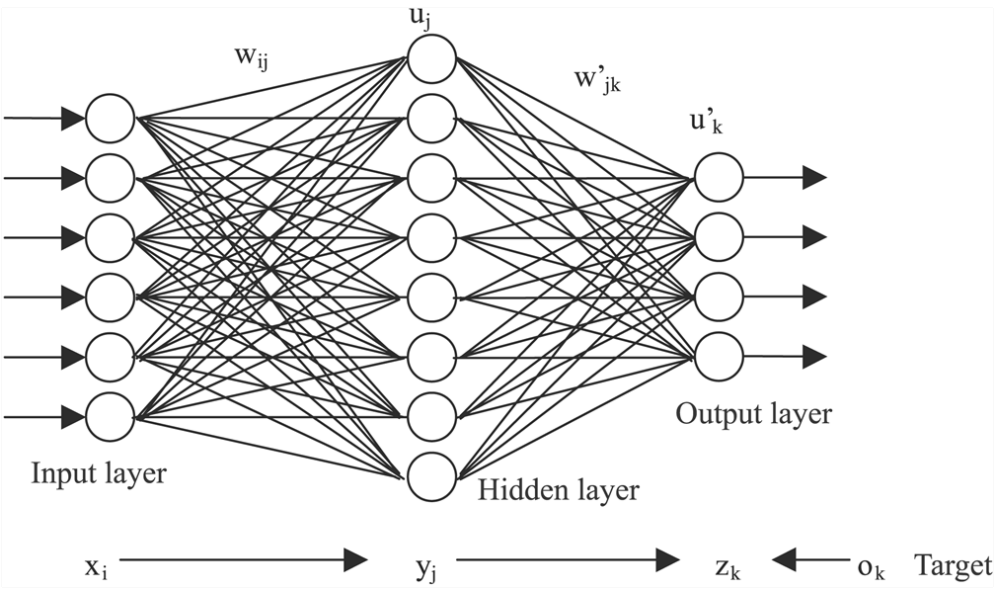


Figure 3.3: Graphical representation of a feed-foward network.

layer. This becomes the input the next layer in an iterative process until the final output of the network o is generated. This is usually compared with a desired target t and a differentiable error function $E(t-o)$ is computed. At this point, the backpropagation is applied to the error function: for each weight of each layer in the network, the respective gradient is calculated and used to move the weights in the direction of the negative gradient vector, in order to minimise the error.

3.6.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are nowadays very popular and their use spreads across many fields, such as autonomous driving (Chen et al. 2015), object detection (Ren et al. 2015), facial expression recognition (Yu & Zhang 2015) and many others (Schmidhuber 2015). But their first use can be dated at the end of the last century, when LeCun et al. (1998) introduced this architecture to recognise hand-written digits. Their popularity is becoming even more appealing given the price reduction involving Graphical Processing Units (GPUs). The use of such hardware can, in fact, speed up matrix and vectors operations which are the working principle of every neural network. Differently from the feed-forward architecture presented before, in a CNN the

units within a layer are grouped in a smaller area by connections called kernels (or filters). The input of the network is represented by a matrix X, whose dimensions are $m \times m \times c$ where m is the height and width of the matrix while c the number of channels. Each convolutional layer is defined by k kernels of size $n \times n \times q$, where $n < m$ and $q \leq c$. When performing a convolution, each element of X is multiplied with its local neighbours and weighted by the kernel W, generating k *feature maps* of size $n - m + 1$. Most of the time, a convolutional layer is thes followed by a pool operation that sub-sample the feature maps over a $p \times p$ region, with p in [2, 5]. The kernels are optimised and adjusted during the training phase using the backpropagation method. The error (or loss) function, also called $J(w)$, is minimised iteratively with gradient descent updating w at $t+1$ using the information ad time t, following the equation:

$$w_{t+1} = w_t - \alpha \nabla E|J(w_t)| + \mu v_t \tag{3.14}$$

The value α is called *learning rate* and defines the step of the algorithm in the direction of the gradient. The value μ is called *momentum* and is a technique for speeding up convergence. When working with CNN, the Rectified Linear Unit (RELU) has been proved to be the most effective activation function. It is defined as follow:

$$Relu(x) = max(0, x) \tag{3.15}$$

It became popular mainly because it helps speeding up the training, considering the simple computation step (all negative values are set to 0) and the gradient is always 0 or 1.

3.7 Reinforcement Learning

Sutton & Barto (1998) defines Reinforcement Learning as the discipline aiming to map situations to action in order to maximise a reward signal. The goal of the learning agent is to learn the best sequence of actions that maximise the long term reward. Like supervised and unsupervised learning, from which it differs because the absence of a

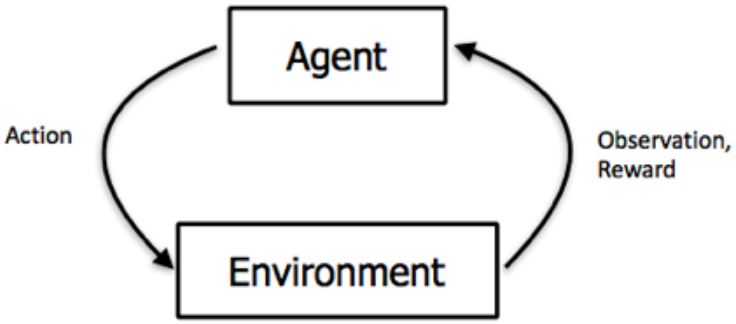


Figure 3.4: Graphical representation of a reinforcement learning cycle.

labelled training set or the objective to generalise to unseen data, RL is an independent subclass of machine learning. In particular, reinforcement learning can be defined by the following elements:

- world: it is the environment in which the agents moves and interacts with other entities, whose rules can be accessible or hidden to the agent.
- state: it represents the current environment configuration at a given time step.
- action: it defines how the agent moves or interact in the world.
- policy: it is the function learned by the agent that maps state to actions.
- episode: it is a series of transition that brings the agent from its starting state to a final one.
- reward: feedback that the agent receives from the world.

In a typical RL scenario, as shown in Figure 3.4, at each time step t the agent receives the state s_t, performs an action a_t sampled from the action space A, and receives a reward r_t given by a reward function $R(s_t, a_t)$. The action brings the agent to a new state s_{t+1} in accordance with the environmental transition model $T(s_{t+1}|s_t, a_t)$. The goal of the agent is to maximise the discounted cumulative reward called return $R = \sum_{k=0}^{\infty} \gamma^k r_{t+1}$, where γ is the discount factor. Given the current state, the agent can select an action from the internal policy $\pi = P(a|s)$. Reinforcement learning algorithms can be divided

in two groups depending on the availability of the transition model. In a simple tabular scenario, the transition model corresponds to a a table which maps to any pair state-action (s_t, a_t) a new state s_{t+1}. This may be read saying that performing action a_t in state s_t brings the agent into state s_{t+1}. The family of algorithms for which the transition model is available is called model-based algorithm. Otherwise, if the transition model is not available and the agent must learn how the environment evolves simply interacting with it, the algorithms are defined as model-free.

3.7.1 (Deep) Q-learning

One of the most popular model-free algorithm existing in literature is *Q-learning*, used to learn a policy that defines which action to take under what circumstances. Differently from other algorithms, in Q-learning the prediction of the cumulative reward can be obtained through an action-value function $Q^{\pi}(s,a)$ (named Q-function) adjusted during the learning phase in order to approximate $Q^{*}(s,a)$, the optimal action-value function. The Q-function, given a state-action pair (s_t, a_t), return the reward signal expressing the quality of performing action a_t in state s_t. In simple cases, Q can be represented in tabular form. The tabular approach is generally inadequate to model a large state space due to combinatorial explosion. To solve the problem, a function approximator (e.g. artificial neural networks) can be used to represent the action-value function, hence the name deep.

The problem faced in DRL is to adjust the parameters θ, which are the weights of the network, minimising a loss function $L(\theta)$ through stochastic gradient descent. The loss function used in this work is the following:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(Y_i - Q(s,a;\theta_i)\right)^2\right] \tag{3.16}$$

where $D = (e_1, ..., e_t)$ is a dataset of experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ used to uniformly sample a batch at each iteration i. The network $Q(s,a;\theta_i)$ is used to estimate actions at runtime, whereas Y_i is the target that is defined as follows:

$$Y_i = r + \gamma \max_{a'} Q(s', a'; \theta_i^-) \tag{3.17}$$

the network $Q(s', a'; \theta_i^-)$ is used to generate the target and is constantly updated. The use of the target network is an expedient which improves the stability of the learning. The parameters θ are updated every C steps and synchronized with θ^-. In the standard approach, the experiences in the dataset D are collected in a preliminary phase using a random policy. The dataset D is also called *buffer replay* and it is a way to randomize the samples breaking the correlation and reducing the variance (WawrzyńSki & Tanwani 2013).

3.7.2 Double (deep) Q-learning

Another issue connected with the reward sparsity is a well known problem called overestimation Thrun & Schwartz (1993). A solution to overestimation has been recently proposed and has been called double DQN (Van Hasselt et al. 2016). The target estimated through double DQN is defined as follows:

$$Y_i^{\mathsf{d}} = r + \gamma\, Q(s', \operatorname*{argmax}_{a'} Q(s', a'; \theta_i); \theta_i^-) \tag{3.18}$$

It was found that the max operator in the original work of Mnih et al. (2015) was responsible for the overestimation affecting all the action but the trigger. Using the target of Equation 3.18 instead of the one in Equation 3.17 the divergence of the DQN action distribution is mitigated resulting in faster convergence and increased stability.

Chapter 4

Image-Based Visual Servoing for Autonomous Landing

In this chapter, a framework for making unmanned aerial vehicles (UAVs) landing on the deck of a ship is introduced. A simple damped spring controller is combined with augmented reality markers in order to facilitate the identification of the landing platform. In this way, it is possible to obtain the 6-DOF relative-pose between the UAV and the landing platform in an easy way. An extended Kalman filter is used to estimate the current ship's position when lost by the UAV's camera.[1]

4.1 Proposed method

4.1.1 Damped Spring Controller

In this work, in order to simplify the process of tuning the controller's parameters, a damped spring controller has been adopted. In the implementation, only two parameters, K_direct and K_rp, were used to modify the spring strength of the directly controlled dimensions (yaw and z) and the leaning ones (x and y). An additional one, $xy\_damping\_factor$, is responsible to approximate a damped spring and to account external disturbances such as air resistance and wind.

The controller inputs are variations in the angles of roll, pitch, yaw, and altitude, re-

[1] A video showing the working principle of this approach is at the following link: https://www.youtube.com/watch?v=J1ib9PIsr-8

spectively denoted as u_Φ,u_Θ, u_Ψ and u_z, defined as follows:

$$u_\Phi = -K\_rp(\hat{x} - x) + c\_rp(\hat{\dot{x}} - \dot{x}) \tag{4.1}$$

$$u_\Theta = -K\_rp(\hat{y} - y) + c\_rp(\hat{\dot{y}} - \dot{y}) \tag{4.2}$$

$$u_\Psi = -K\_direct(\hat{\psi} - \psi) + c\_direct(\hat{\dot{\psi}} - \dot{\psi}) \tag{4.3}$$

$$u_z = -K\_direct(\hat{z} - z) + c\_direct(\hat{\dot{z}} - \dot{z}) \tag{4.4}$$

where c_rp and c_direct are the damping coefficients calculated in the following way:

$$c\_rp = xy\_damping\_factor \cdot 2\sqrt{K\_rp \cdot droneMass} \tag{4.5}$$

$$c\_direct = 2\sqrt{K\_direct \cdot droneMass} \tag{4.6}$$

Therefore, instead of controlling nine independent parameters (three for the yaw, three for the vertical speed and three for roll and pitch paired together) the control problem is reduced to the three described above (namely K_direct, K_rp and $xy\_damping\_factor$).

The remaining controller parameters are platform dependent variables and they are kept always constant during all the trials. Ignoring $droneMass$ which does not require an additional description more than its name, max_yaw, max_gaz_rise and max_gaz_drop limit the rotation and linear speed on the yaw and z-axis, respectively. In the end, max_rp limits the maximum leaning command sent.

4.1.2 AR markers as landing spot

The AR markers previously introduced in Chapter 3, given their property of robustness to geometrical transformation and lightining variations, are considered suitable for a possible application in a marine scenario, where the landing platform can be subject to adverse conditions that can affect its direct observation.

The *ar_pose* ROS package (Dryanovsk et al. 2010), a wrapper for the $ARToolkit$ library widely used in human computer interaction (HCI) (Kato & Billinghurst 1999, Abawi et al. 2004), is used for achieving this task. It estimates the marker's pose at a frequency of

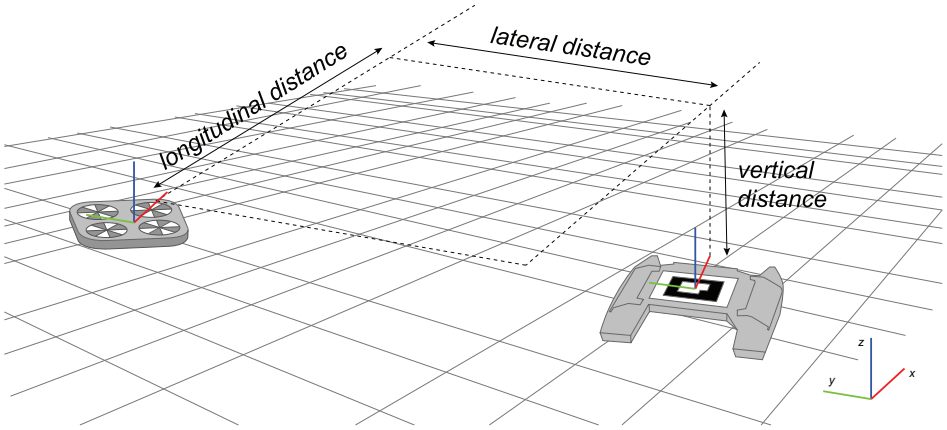


Figure 4.1: The image processing algorithm estimates the distances between the UAV and the visual marker.

Table 4.1: Maximum distance per marker's size (length of the side).

Marker Size (cm)	Maximum distance (m)
4	0.86
7	1.72
10	1.95
13	2.77
16	3.54
19	3.96
22	4.37
25	5.30
28	6.42
31	6.62

1 Hz as shown in Fig. 4.1. For the current and the last marker's observation, the time stamp and the transformation is recorded. This information is then used to detect if the marker has been lost and to actuate a compensatory behaviour.

In Table 4.1 and in Fig. 4.2 some results comparing the size of the marker and the maximum distance at which it can be seen are reported. It has been confirmed that increasing the size of the marker allows the UAV to identify it from a longer distance. On the other hand, a bigger marker cannot be used for precise landing because it would be impossible to keep within the camera's FOV.

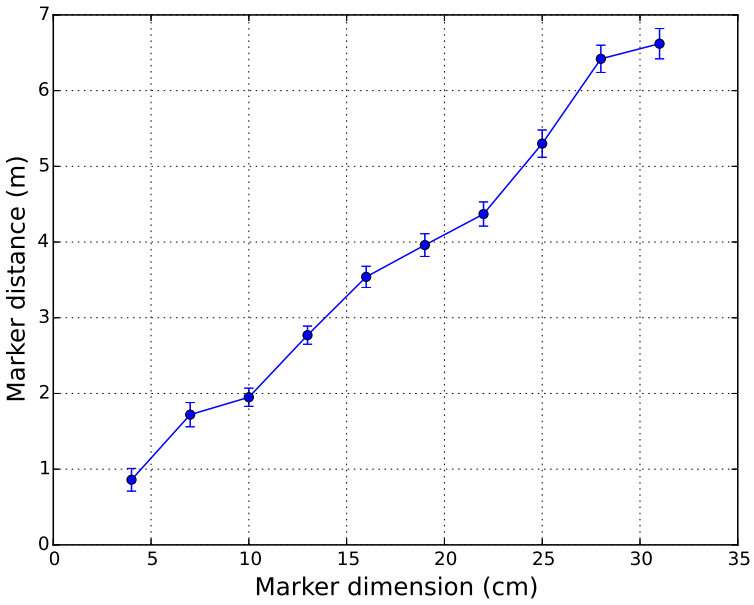


Figure 4.2: The relationship existing between the marker's size (side's length) and the distance at which it can be perceived.

4.1.3 The pose estimation filter

To increase the robustness and efficiency of the approach, an extended Kalman filter (EKF) has been adopted here for estimating the pose of the landing platform (Moore & Stouch 2016). For instance, the UAV could potentially lose the track of the fiducial marker while approaching and descending on it. In order to redirect the flying vehicle in the right direction, the EKF estimates the USV current pose that is then processed and forwarded to the controller. For estimation purposes, the odometry data coming from the landing platform and inertial data are fused together to increase the accuracy (Jetto et al. 1999, Chenavier & Crowley 1992). Adopting the mathematical formulation used in Chapter 3, the state vector is defined as $\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$, with x, y, z and $\dot{x}, \dot{y}, \dot{z}$ representing respectively the global positions and velocity, and ϕ, θ, ψ and $\dot{\phi}, \dot{\theta}, \dot{\psi}$ the attitude of the vessel.

The working principle of the EKF in this case is detailed below:

- the filter estimates the USV's pose at 50Hz and its encoding is saved in an hash

table using the time-stamp as the key;

- when the UAV loses the track, the hash table is accessed to retrieve the last record inserted (the most recent estimate produced by the filter) together with the one having as key the time-stamp of the last recorded observation;
- the deck's current position with reference to the old one is calculated using the geometric relationship;
- the controller commands are updated including the new relative position;

The procedure described above is iterated until the UAV is redirected above the visual marker and can perceive it through its bottom camera.

4.1.4 Methodology

Algorithm 1: Landing Algorithm

```
1:  while not landed do
2:      last_known_pose =NULL
3:      if marker_visible then
4:          last_known_pose ← detect_landing_marker()
5:          if last_known_pose < user_defined_threshold then
6:              controller.send_commands(land)
7:              landed ← true
8:          end if
9:      else
10:         usv_pose ← ekf.estimates_pose()
11:         last_known_pose ← last_known_pose + usv_pose
12:     end if
13:     trajectory ← calculate_trj(last_known_pose)
14:     attitude_cmd ← controller.calculate_cmd(trajectory)
15:     controller.send_commands(attutude_cmd)
16: end while
```

The following section explains how the algorithm 1 works. The code is publicly available on the repository [2].

The quadcopter flies using its fixed non-tilting frontal camera, approaching the landing site on the USV's deck identified only by a fiducial marker. This, which scope is to out-

[2]Github repository: https://github.com/pulver22/ardrone_tf_controller

line the landing area, has to be perceived during the entire landing manoeuvre. This is a requirement for precise landing despite the state estimator can compensate interruption in observation. When a visual marker is detected, the image processing library computes the 6-DOF relative-pose between the marker itself and the UAV. The result is used to make the quadcopter approaching the marker with the right orientation. To obtain this result, a damped spring controller reduces the error on the $x-$, $y-$ and $z-$axis and on the quadcopter's yaw. On attaining close proximity to the marker, the marker leaves the field of view of the frontal camera. This is due to hardware limitation of fixed non-tilting cameras. To overcome this problem, the video stream from the frontal camera is interrupted and acquired from the one located under the UAV and downward-looking. The quadcopter continues the landing manoeuvre keeping the marker at the centre of the second camera's FOV. Otherwise, a compensatory behaviour is adopted: the EKF estimates the actual position of the USV and the drone is redirected close to it while increasing its altitude. Increasing the altitude allows to enlarge the field of view of the bottom camera, that is quite limited. In this way, it is guaranteed that the marker will be soon perceived and centred by the aerial vehicle. The drone lands safely when an experimentally defined distance from the marker is reached. This distance depends on the side length of the marker used. In fact, with a smaller marker it would be possible to decrease this value but it would become impossible to perceive the marker at longer distance. It was found that a marker side length of approximately 0.30 m represents a good trade-off for making the marker visible at long and close distance at the same time. As a consequence, we decide to use 0.75 m as distance for starting the touch-down phase of the descending manoeuvre, during which the power of the motors is progressively reduced until complete shut-down. The use of visual markers allows the estimation of the full 6-DOF pose information of the aerial and surface vehicles. In this way, landing operations in rough sea condition with a significant pitching and rolling deck can still be addressed.

4.2 Results

All the experiments have been conducted inside a simulated environment built on Gazebo 2.2.X and offering a 3D model of the AR Drone 2.0. To the scope of this work, the existing simulator has been partially rewritten and extended to support multiple different robots at the same time.

Two different series of simulated experiments have been performed.

In the first series of experiments, the Husky A200, produced by Clearpath Robotics, has been used as a mobile base. The Husky was chosen because known to and used by a number of research groups involved in mobile robotics, and at the same time because it is equipped with a plane representing a versatile deck for UAVs of small dimension. In the second series the Kingfisher USV, still produced by Clearpath Robotics, has been used as a floating base. It is a small catamaran with a dimension of are 135 x 98 cm, that can be deployed in an autonomous or teleoperated way.

Both vehicles present a flat plane usually used for mounting various sensors. On this surface a square visual marker is placed. Previous research demonstrated a linear relationship is existing between the side length of the marker and its observability. Therefore, it was opted for a side length of 0.3 m that represents a good compromise, making the marker visible in the range [0.5, 6.5] m.

4.2.1 Landing on a mobile ground robot

The controller parameters for the experiments presented here are reported in Table 4.2. Different poses, expressed as position and orientation, have been set to the UAV with reference to the landing platform.

The approach here reported has been tested either with a static landing base and a moving one. In the first scenario the mobile base remains in the same position for all the length of the flight. The main scope of this experiment is to show that the UAV is able to find the proper orientation while approaching the visual marker and then land safely on it with high precision. On the other hand, when the mobile base is actually

Table 4.2: Controller parameters for the static landing platform experiment.

Parameter	Value	Parameter	Value
K_direct	5.0	K_rp	0.3
droneMass [kg]	0.525	max_yaw [rad/s]	1.0
xy_damping_factor	0.65	max_gaz_rise [m/s]	1.0
max_gaz_drop [m/s]	-0.1	max_rp	1.0

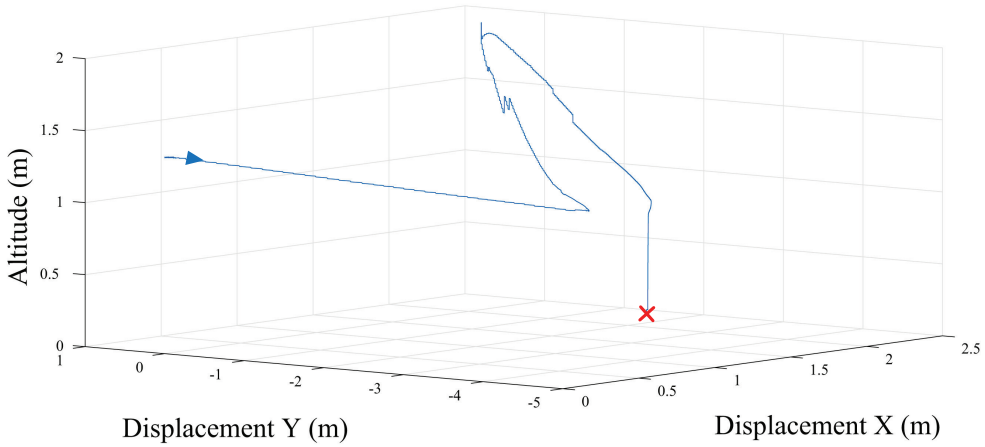


Figure 4.3: Quadcopter trajectory in three-dimensional space and top view during the experiment with a static platform.

moving in a straight line (a velocity command is continuously sent to the Husky), the interest is on tracking the marker and in the time required by the UAV to approach it.

4.2.1.1 Static platform

In this subsection, the attention is focused on the precision required by the unmanned aerial vehicle to align with a visual marker. For this reason, the time required to complete the landing manoeuvre is not considered as key-factor. The controller's parameters are shown in Table 4.2. The K_rp parameter, responsible to control the roll and pitch behaviour, is kept small in order to guarantee smooth movements along the leaning dimensions. In the same way, max_gaz_drop has been reduced to a value of 0.1 for decreasing the descending velocity. On the other hand, the max_yaw parameter, used to control the yaw speed, has been set to its maximum value because the drone must align with the base in the minimum amount of time possible.

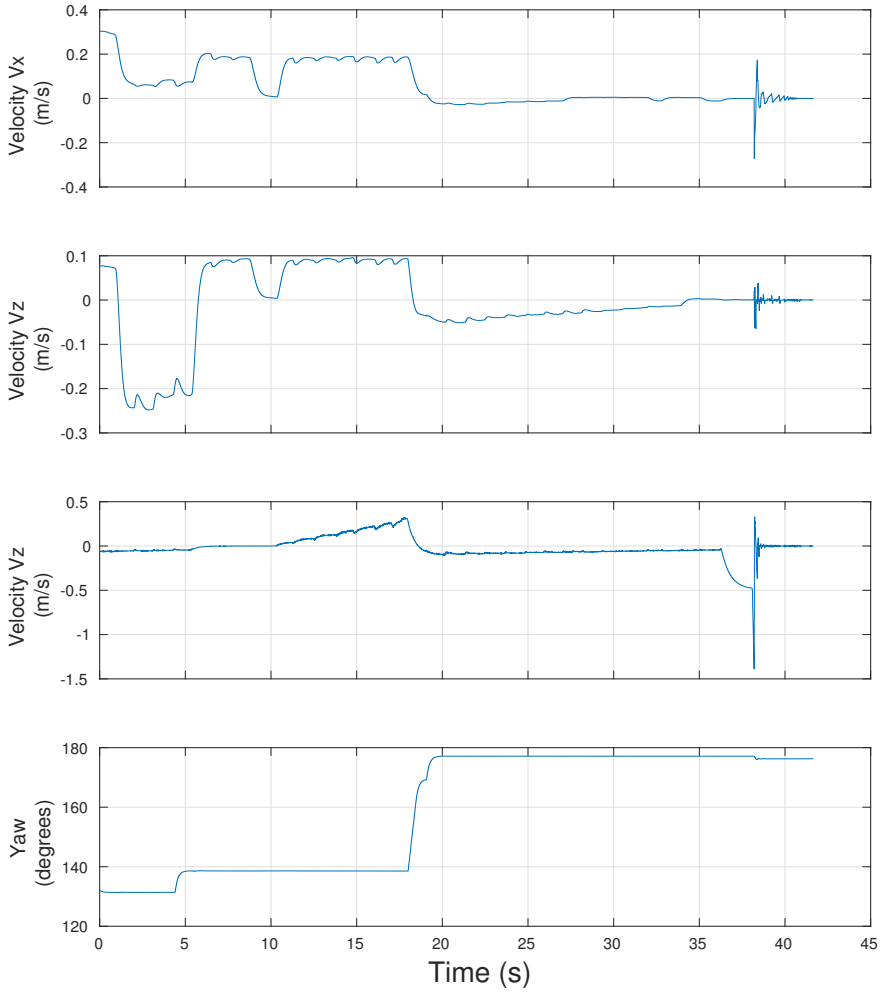
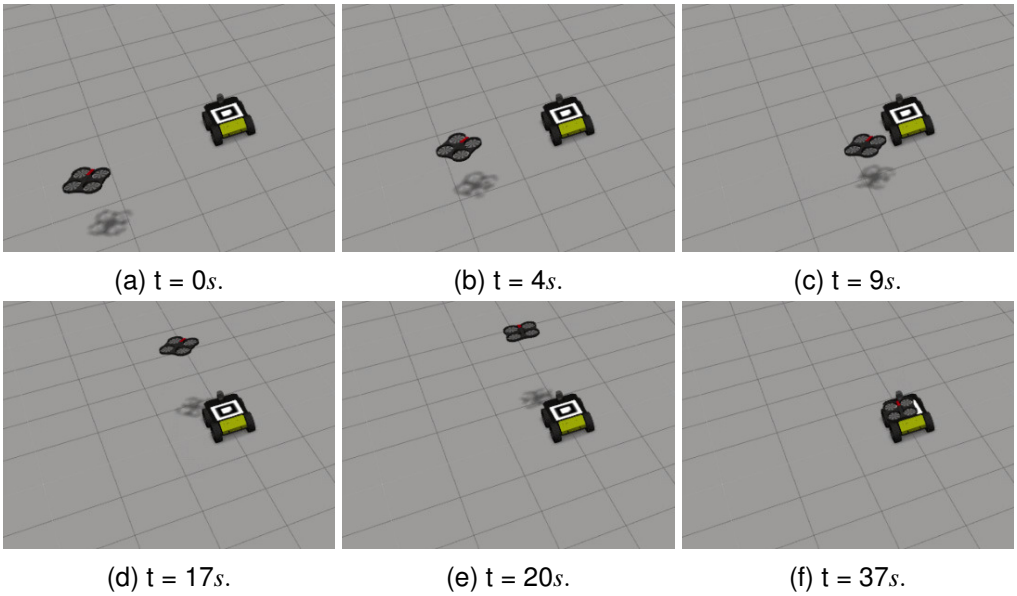


Figure 4.4: Quadcopter's velocity and yaw profile.

The results are reported in Fig. 4.3 and Fig. 4.4, where trajectory and state profile of the UAV are illustrated. In the second graph, a Hue Saturation Brightness (HSB) colour-map is used to dicretise different altitudes. The red colour identifies the maximum and minimum altitudes reached by the UAV, while intermediate values are reported in green an blue.

The marker has been successfully recognised at a distance of 2.16 m in front of the UAV, and at 1.93 m on its right. As depicted in Fig. 4.5a, the displacement on the



(a) t = 0s. (b) t = 4s. (c) t = 9s.

(d) t = 17s. (e) t = 20s. (f) t = 37s.

Figure 4.5: Landing manoeuvre of a VTOL UAV on a static base.

$z-$axis, used as the reference for the altitude, was 1.37 m instead. The UAV, with the parameters reported in the previous table, has been able to complete the landing in 37 s.

The quadcopter approaches the landing base trying to keep it at the centre (in a range of $\pm$ 10 degrees) of its camera's FOV. When the marker leaves this interval of tolerance, the UAV rotates around its $z-$axis in order to centre it again. This is what happens at $t = 4s$ and illustrated in Fig. 4.5b, when the quadcopter changes its orientation and continues the manoeuvre. It is interesting to notice the deviation of almost 2 m on the $y-$axis that occurs between $t = 4s$ and $t = 8s$. This is caused by leaning effects caused by commands generated by the controller when the quadcopter still had the old orientation. As soon as these effects are compensated by the new commands, the UAV approaches the visual marker with the current orientation. This happens until the UAV's low altitude prevents the marker to be seen from the frontal camera, as shown in Fig. 4.5c ($t = 8s$). At this point, the video stream is switched from the frontal camera to the one located at the bottom of the quadcopter and looking down, and new commands are generated and sent. The UAV is instructed to move towards the last known position of the landing platform but increasing its altitude in order to enlarge the area covered

by its bottom camera. At $t = 17s$, as represented in Fig. 4.5d, the UAV is located exactly above the marker and it can now complete the landing phase: it therefore adjusts its orientation accordingly ($t = 20s$) and descends while trying to keep the marker at the centre of its FOV, as shown in Fig. 4.5e. Small velocity commands are sent on the leaning direction (x and y, respectively) in order to approach the final position with high accuracy. Finally, at $t = 33s$ the UAV reaches the minimum altitude (z = 0.75 m) required to shut-down its motors and land on the platform (Fig. 4.5f). The oscillations possible to see in Fig. 4.4 at this instance of time are caused by the quadcopter landing only due to the gravity force and hitting the platform.

The commands generated from the relative-pose between the UAV and the landing platform's frame are illustrated in Fig. 4.6. Here, the controller's commands are plotted against the perception from the camera. As it is possible to see in the figure, for the main part of the travel the two curves of the commands and of the observations overlap perfectly. The only portion in which the two curves are not aligned is between $t = 11s$ and $t = 18s$. Here, the marker is lost and the UAV is actuating the compensatory behaviour: it is instructed to keep its orientation constant and to proceed in the direction of the last observation. In this way it is possible to justify why the pitch, roll and yaw commands do not change meanwhile the UAV's altitude increases.
A dedicate analysis is reserved for the altitude's data between $t = 8s$ and $t = 10s$, and the yaw's ones between $t = 20s$ and $t = 33s$. In this case, the offsets are below a threshold chosen by the user at the beginning and a respective command equal to 0 is sent instead. The use of a threshold has been introduced for speeding up the landing phase: while testing the controller it was noticed the UAV spent a lot of time while trying to align perfectly on the three axes with the centre of the landing phase, sometimes without any success. This has been identified as a limitation of controllers with fixed values parameters and a new more versatile solution is already planned as future work.

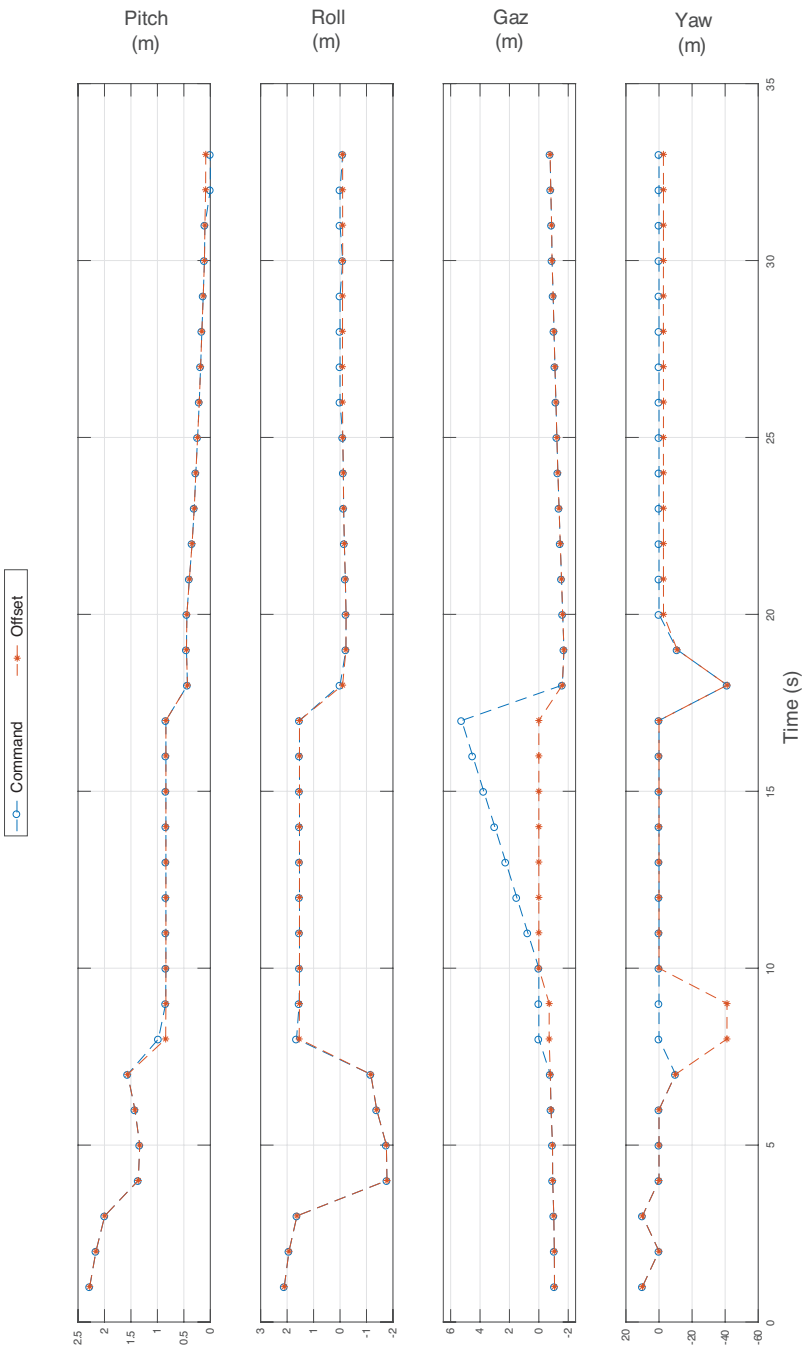


Figure 4.6: Controller commands and visual offsets in the experiment with static landing platform.

Table 4.3: Controller parameters for the moving landing platform experiment.

Parameter	Value	Parameter	Value
K_direct	5.0	K_rp	0.5
droneMass [kg]	0.525	max_yaw [rad/s]	1.0
xy_damping_factor	0.65	max_gaz_rise [m/s]	1.0
max_gaz_drop [m/s]	-0.15	max_rp	1.0

4.2.1.2 Moving platform

In this subsection an experiment with a moving landing platform is reported. As before, the time for completing the landing manoeuvre is not considered as key-factor but the attention is on the ability of the UAV to approach the unmanned ground vehicle (UGV) and land on it with high precision. Considering the size of the deck, identical of the UAV's one, it has been decided to send a stopping command to the UGV once the quadcopter is located exactly on it. This decision was forced by the trade-off posed by the size of the visual marker. Recalling what said in Section 3.3, a smaller one could allow to perceive it also when the displacement on the $z-$axis between the UAV and the base is relatively small because it would occupy a smaller portion of the camera's FOV. On the other hand, it would be more difficult to perceive it from long distances.

In Table 4.3 the controller parameters are reported. Differently from the experiments with a static landing platform, the K_rp parameter has been increased to allow the UAV to move faster along the x and y dimensions. To compensate for the fact that approaching the UGV at high speed would cause the UAV to lose the marker from its frontal camera, the value of the max_gax_drop parameter has been decreased. In this way, the quadcopter moves at higher speed on the $z-$axis and it can use its frontal camera for more time while leaning. The parameter controlling the yaw speed, max_yaw, is kept at its maximum value to minimise the alignment time with the landing base. Regarding the UGV, a velocity command on x of 0.1m/s was constantly sent to make it move in a straight line. Again, the time required for the approach was not a metric of the experiments and for this reason it was decided to use such a slow speed for the ground robot.

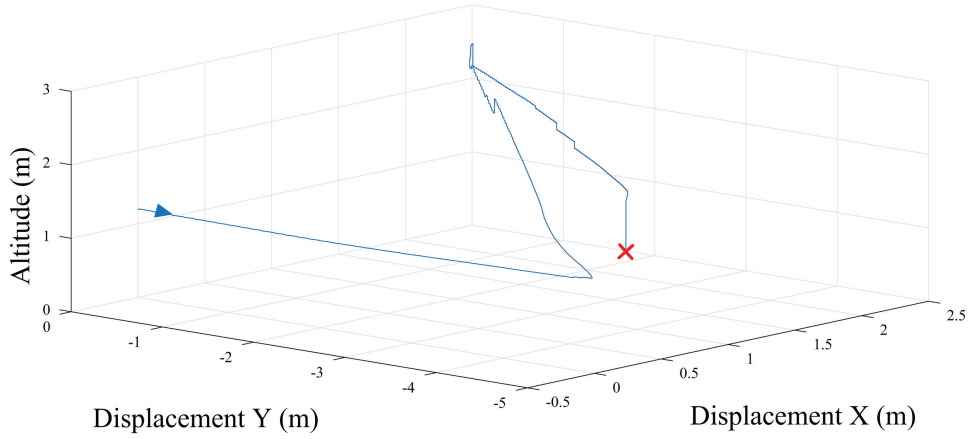


Figure 4.7: quadcopter trajectory in three-dimensional space and top view during the experiment with a moving platform proceeding in straight line.

As in the previous experiments, the 3D trajectory and the state profile are reported in Fig. 4.7 and Fig. 4.9. The quadcopter, with the controller set as described above, was able to follow and land on the visual marker in almost 82 s.

As in the case of a static marker, Fig. 4.10a shows the UAV starts moving at time $t = 18s$ and rotating in order to keep the visual marker at the centre of its frontal camera's field of view. This is what happens at time $t = 26s$ and shown in Fig. 4.10b. At $t = 41s$ the UAV reaches its minimum altitude and it's now impossible for it to see the visual marker, as illustrated in Fig. 4.10c. At this point, the video stream starts to be acquired from the bottom camera and the UGV's estimated position is sent to the controller. At the same time, it is told to the UAV to increase its altitude to augment the total area covered with its downward-looking camera. Doing this, at $t = 58s$ the UAV is located exactly above the UGV and it rotates accordingly until the proper orientation is reached (Fig. 4.10d). The landing base is at the centre of the camera's FOV, therefore a null velocity command is sent to stop the UGV. Fig. 4.10e and 4.10f show the UAV can then descend slowly to centre the marker properly and, in the end, land on it.

Further analysis can be done with the results reported in Fig. 4.8. The data are recorded since $t = 0s$, after which all the tools are started and a constant velocity com-

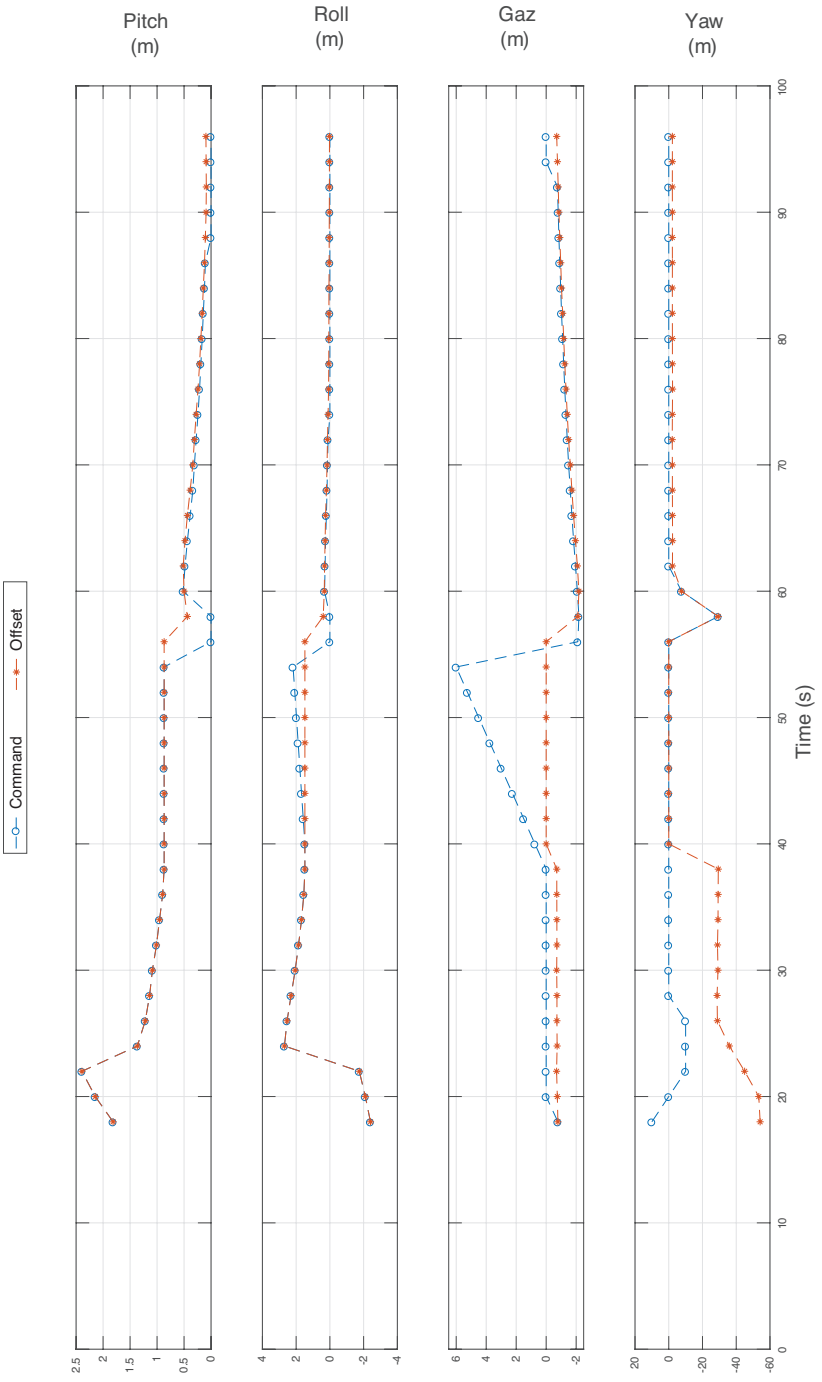


Figure 4.8: Controller commands and visual offsets in the experiment with a mobile ground robot moving in a straight line.

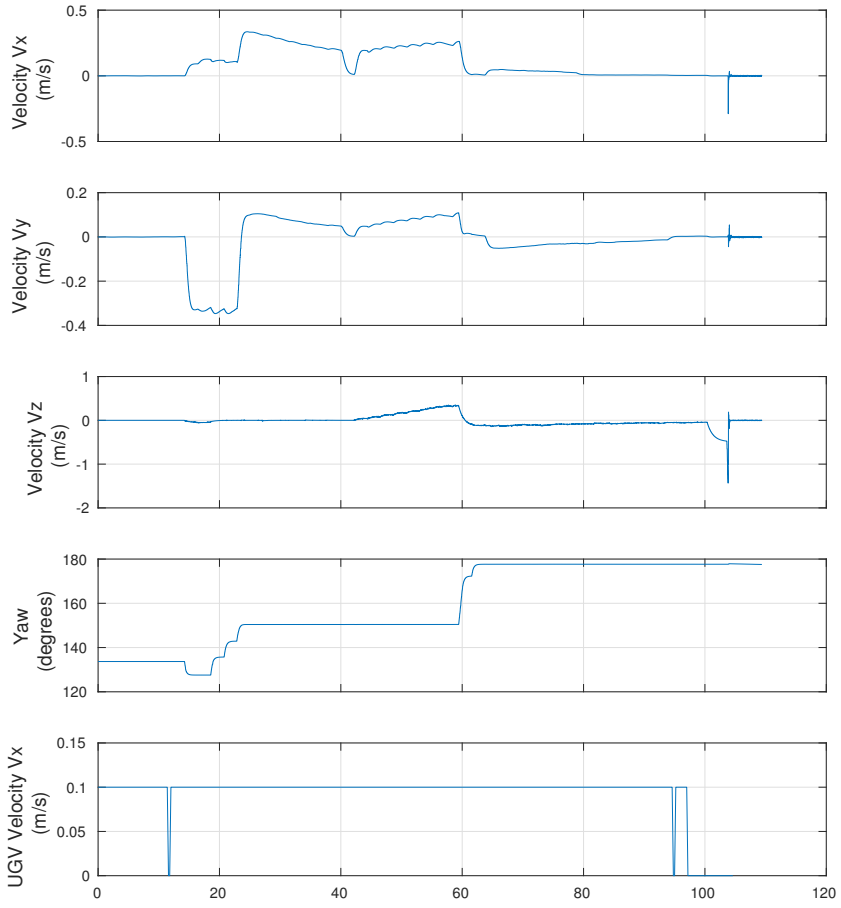
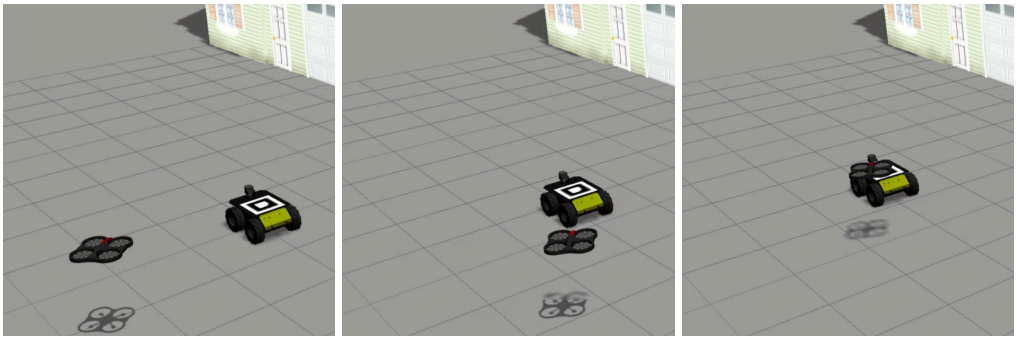


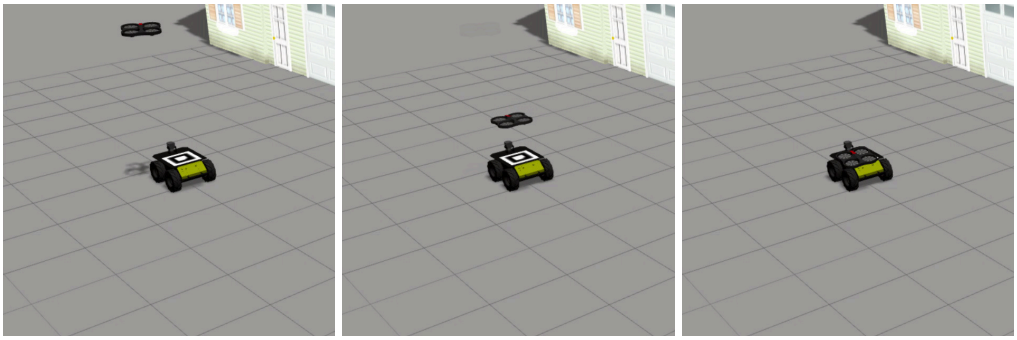
Figure 4.9: quadcopter's velocity and yaw profile during the experiment with a moving platform proceeding in straight line.

mand of 0.1 m/s is sent to the UGV at $t = 10s$. The UAV starts to acquire the pictures and to move at $t = 18s$.

In the same way of the experiment with a static deck, the curve of the controller's commands and the one related to the offsets overlap for most of the time. The difference with the previous experiments consists in that, while the marker is lost, the UAV is not directed towards the last observation. In fact, the EKF is able to estimate the landing platform's current pose with reference to the instant of time when the marker has been lost. This relative-pose is added to the last observation in order to produce a new com-



(a) t = 0s. (b) t = 26s. (c) t = 41s.



(d) t = 60s. (e) t = 96s. (f) t = 100s.

Figure 4.10: Landing manoeuvre of a VTOL UAV on a moving base.

mand.

This is what is possible to see in the plot between $t = 42s$ and $t = 58s$. Here, the two curves differ: while the offsets (on the roll, pitch, yaw and the altitude) remain constant because no new marker observations have been done by the UAV, the commands slightly change. The plot is now discussed in more details. While the yaw command remains identical to 0 because the UAV is already aligned with the landing base, the pitch command does not change because the UGV is moving in a straight line and it is not deviating on the lateral direction from his current track. Differently from the pitch but for the same reason, the roll command is changed including at every instant the new relative-pose (changing on the longitudinal direction) of the UGV.

To conclude, the offset between the two curves for the roll (between $t = 90s$ and $t = 96s$), the altitude (between $t = 22s$ and $t = 40s$) and the yaw (from $t = 60s$ until the end) is justified, as explained for the previous experiment, by the adoption of a threshold to speed up the completion of the landing manoeuvre.

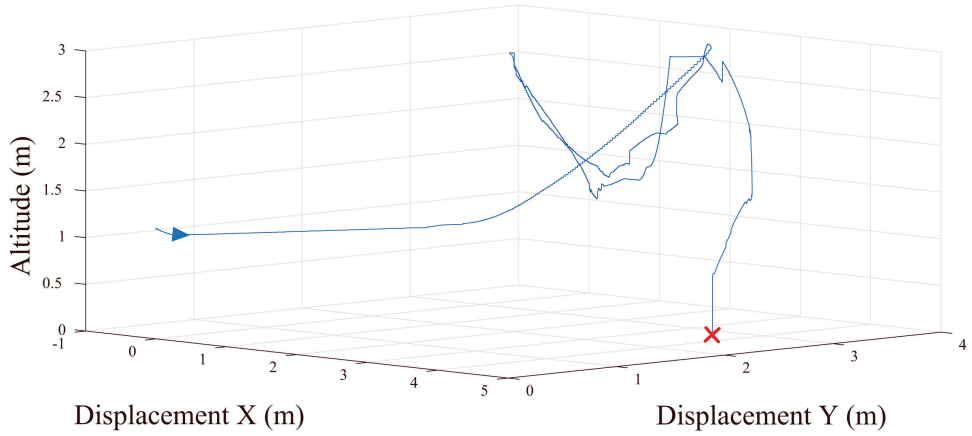


Figure 4.11: Quadcopter trajectory in three-dimensional space and top view during the experiment with a moving platform that also rotates.

For a different reason it is possible to explain why the yaw's command and its respective offset are diverse. As discussed in the previous sections, while flying with the frontal camera, the UAV keeps the fiducial marker at the centre of the FOV. In this case, if the UAV rotated to align with the marker it would most probably lose it. To avoid so, the UAV is instructed that, when using the frontal camera, it can rotate only ten degrees per time and only it the marker is located at the edge of the FOV. Doing so, the alignment is speed down but it is prevented the marker is lost.

The last experiment has been done with a moving platform that not only proceeds in straight line but also rotates changing its heading angle. The goal of this experiment is to show how the pose estimation filter can compensate for the lack of the visual technique in predicting where the UAV has to move to perceive the visual marker again. The controller parameters used are the same of the previous experiment and they are shown in Table 4.3. Results are reported in Fig. 4.11 and 4.12, showing the UAV's profile and its trajectory. In Fig. 4.14 are shown the comparison between the offsets obtained through the vision algorithm and the commands sent to the controller. Finally, in Fig. 4.13 the UGV's linear and rotational speed is reported. Note that the UGV is able only to move in a straight line or rotate on the spot at the same time.

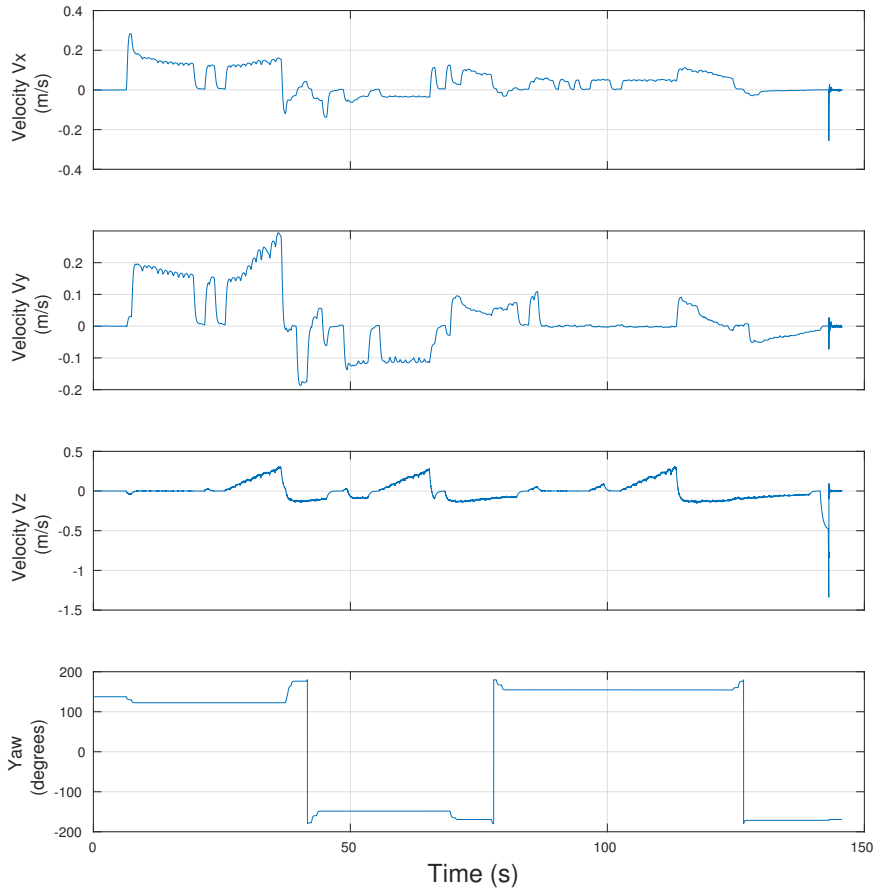


Figure 4.12: Quadcopter's velocity and yaw profile during the experiment with a moving platform that also rotate.

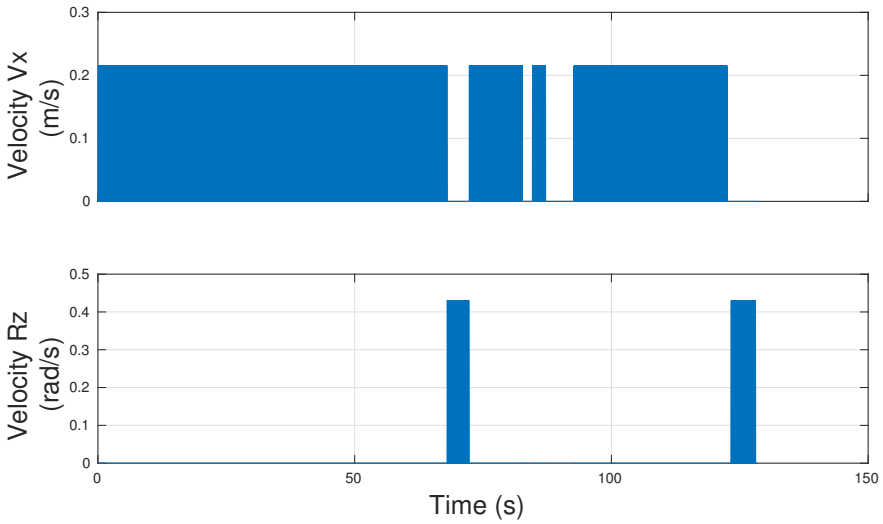


Figure 4.13: Linear and Rotational velocity of the UGV.

In Fig. 4.14 it is possible to see that, as in the previous experiments, the curve of the offsets and the one related to the commands overlap for most of the time. All the analysis made before still hold, but it is interesting to notice how the framework here presented is able to react properly when the visual marker is lost and to redirect the UAV above it despite changes in position.

4.2.2 Landing on an unmanned surface vehicle

In the second series of simulated experiments, the proposed algorithm has been tested against a floating base under multiple conditions, namely three. In the first scenario, the USV is subjected only to a rolling movement while floating in the same position for all the length of the experiment; in the second scenario, the USV is subjected only to a pitching movement; while in the last scenario the USV is subject to both rolling and pitching disturbances at the same time. Fig. 4.15 illustrates the rotation angles around their corresponding axis. In all the simulations, the disturbances are modelled as a signal having a maximum amplitude of 5 degrees and a frequency of 0.2 Hz. Rolling and pitching of a vessel generate upward and downward acceleration forces directed tangentially to the direction of rotation, which causes linear motion knowns as swaying and surging along the transverse or longitudinal axis respectively (Handbook 2003).

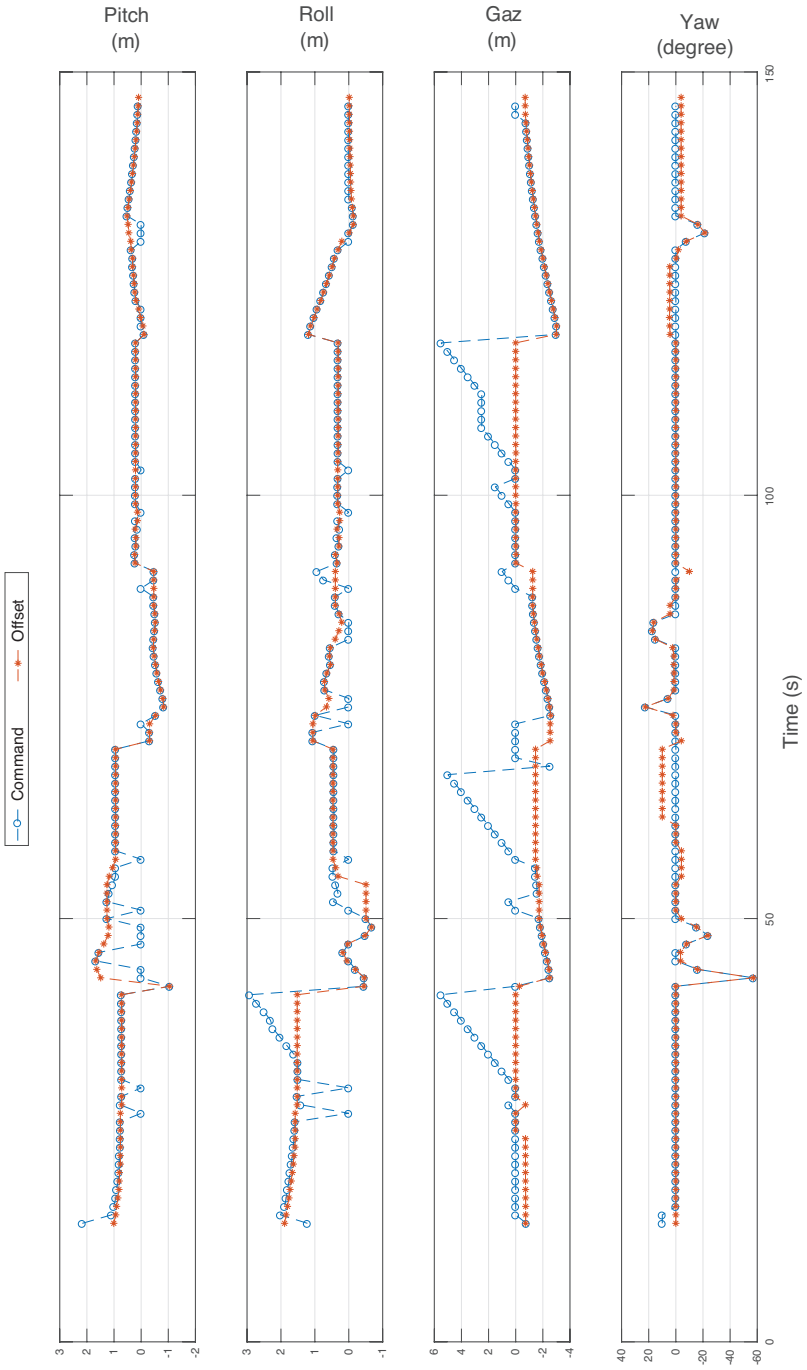


Figure 4.14: Controller commands and visual offsets in the experiment with a mobile ground robot that also rotates.

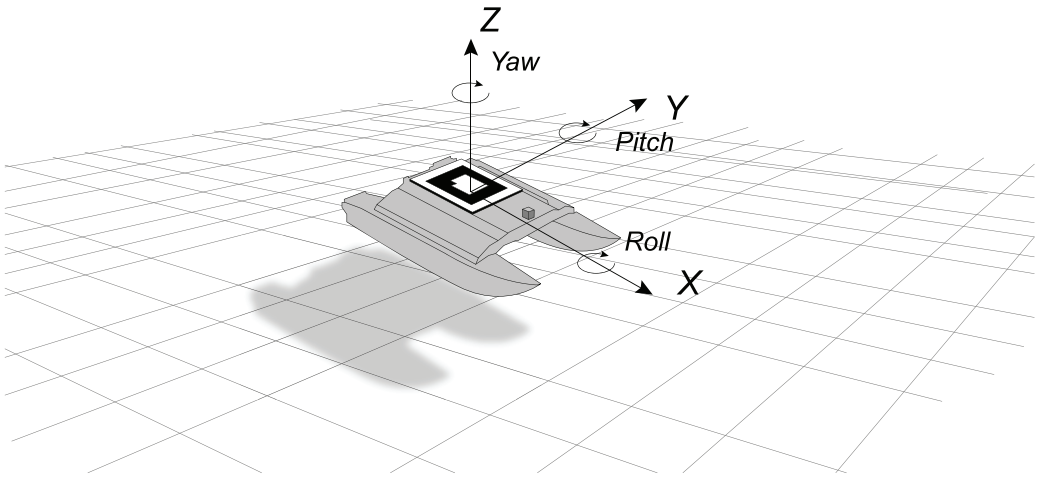


Figure 4.15: The movements around the vertical, longitudinal and lateral axis of the USV are called yaw, roll and pitch respectively.

Table 4.4: The controller parameters used in the simulations performed with a floating landing platform.

Parameter	**Value**	**Parameter**	**Value**
K_direct	5.0	K_rp	0.3
droneMass [kg]	0.525	max_yaw [rad/s]	1.0
xy_damping_factor	0.65	max_gaz_rise [m/s]	1.0
max_gaz_drop [m/s]	-0.1	max_rp	1.0

The controller's parameters are the same across all the experiments performed and they are shown in Table 4.4. The K_rp parameter, responsible to control the roll and pitch behaviour, is kept small in order to guarantee smooth movements along the leaning dimensions. In the same way, max_gaz_drop has been reduced to a value of 0.1 for decreasing the descending velocity. On the other hand, the max_yaw parameter, used to control the yaw speed, has been set to its maximum value because the drone must align with the base in the minimum amount of time possible. The others have been left to their default values.

4.2.2.1 Rolling platform

In this subsection, the results of a landing manoeuvre on a rolling floating base are reported. In particular, Fig. 4.16 illustrates the UAV and the USV's trajectory, respec-

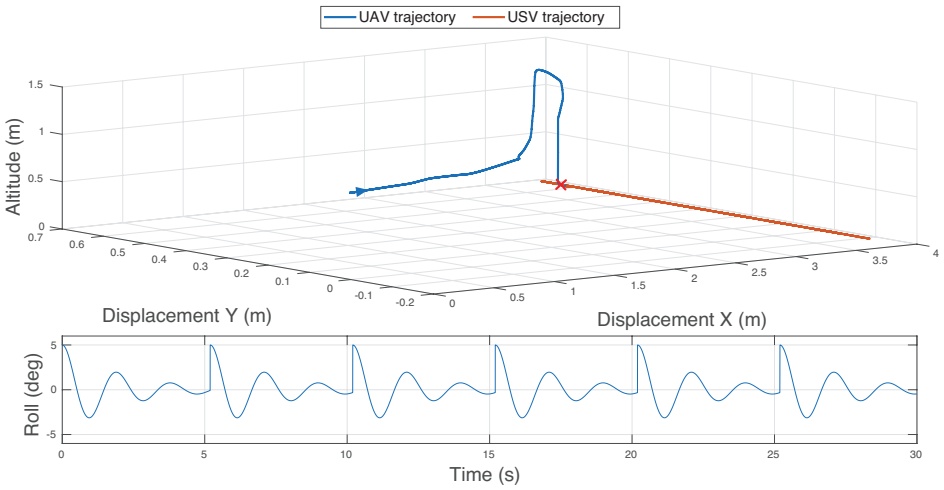


Figure 4.16: (Above) The UAV and USV 3D trajectories, in blue and red, respectively, in the UAV's reference frame. (Bottom) The roll disturbances the USV is subjected.

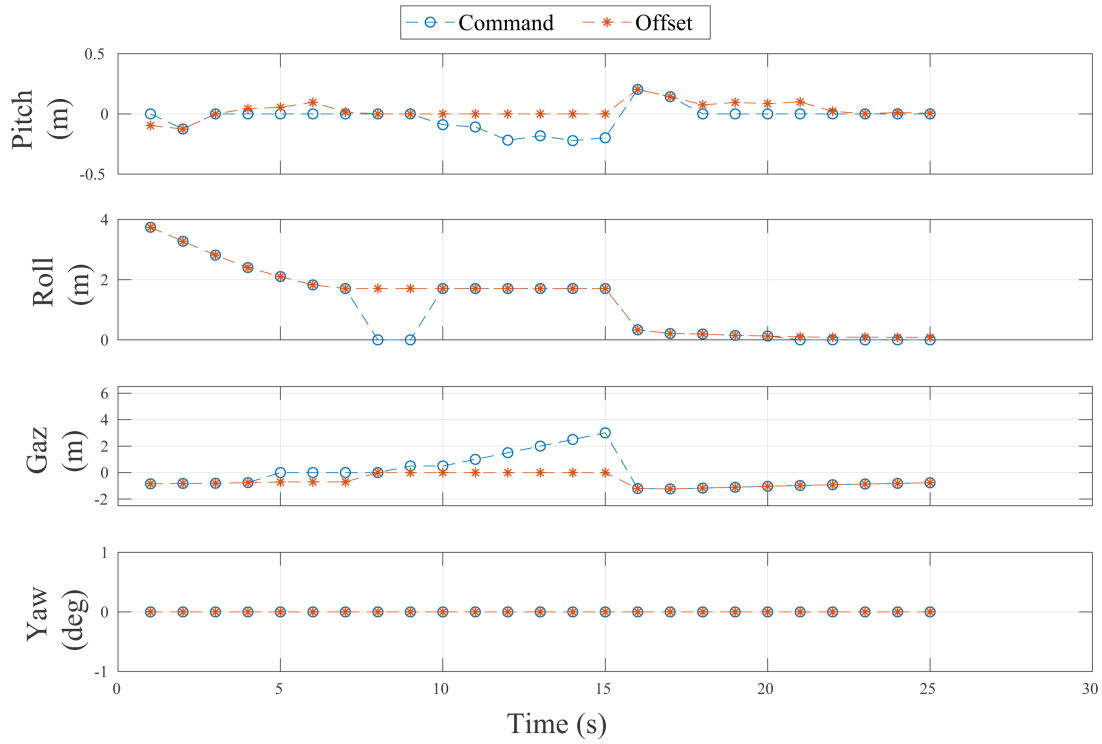


Figure 4.17: Controller commands and visual offsets in the experiment with a rolling landing platform.

tively in blue and red, in the UAV's reference frame; Fig. 4.18 show the UAV's profile along the trajectory, while Fig. 4.17 and Fig. 4.19 show the controller commands and the salient moments of the manoeuvre respectively.

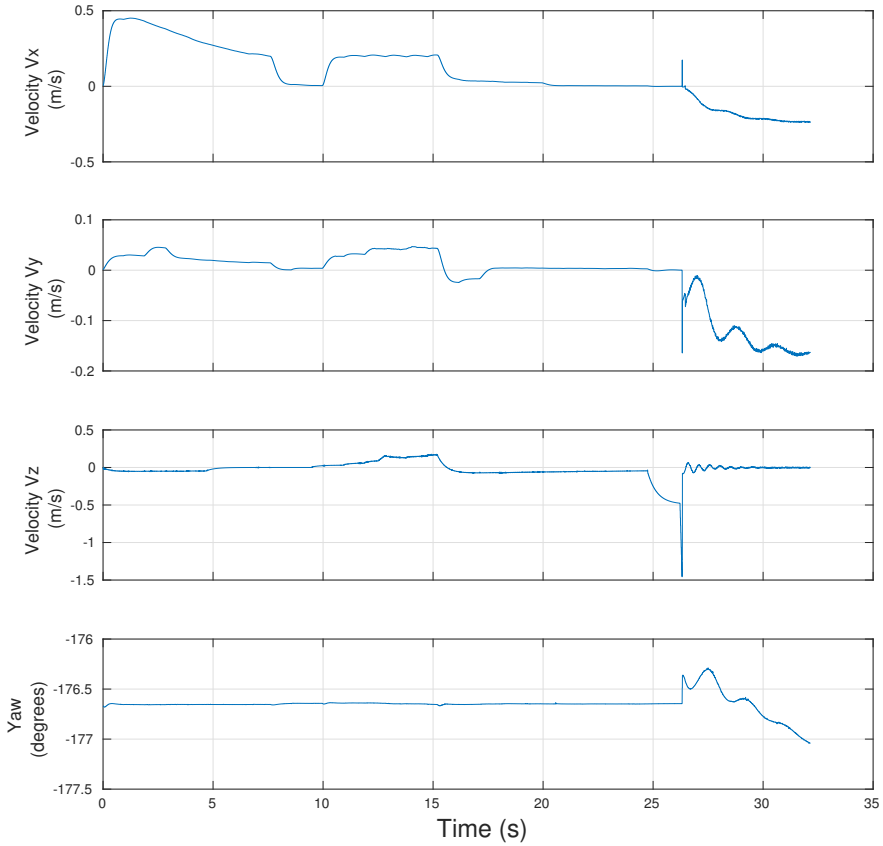


Figure 4.18: Quadcopter's velocity and yaw profile during the experiment with a rolling platform.

The marker has been successfully recognised at a distance of 3.74min front of the UAV, and at 0.09 meter on its left. The displacement on the $z-$axis, used as reference for the altitude, was of 0.84 meter instead. The UAV, with the parameters reported in the previous table 4.4, has been able to complete the landing in 25 s.

The quadcopter approaches the landing base trying to keep it at the centre (in a range of $\pm$ 10 degrees) of its camera's FOV. In the case the marker leaves this interval of tolerance, the UAV would rotate around its $z-$axis in order to centre it again. The approach continues until the UAV's low altitude prevents the marker to be seen from the frontal camera, as shown in Fig. 4.19-a ($t = 10s$). At this point, the video stream is switched from the frontal camera to the one located at the bottom of the quadcopter and

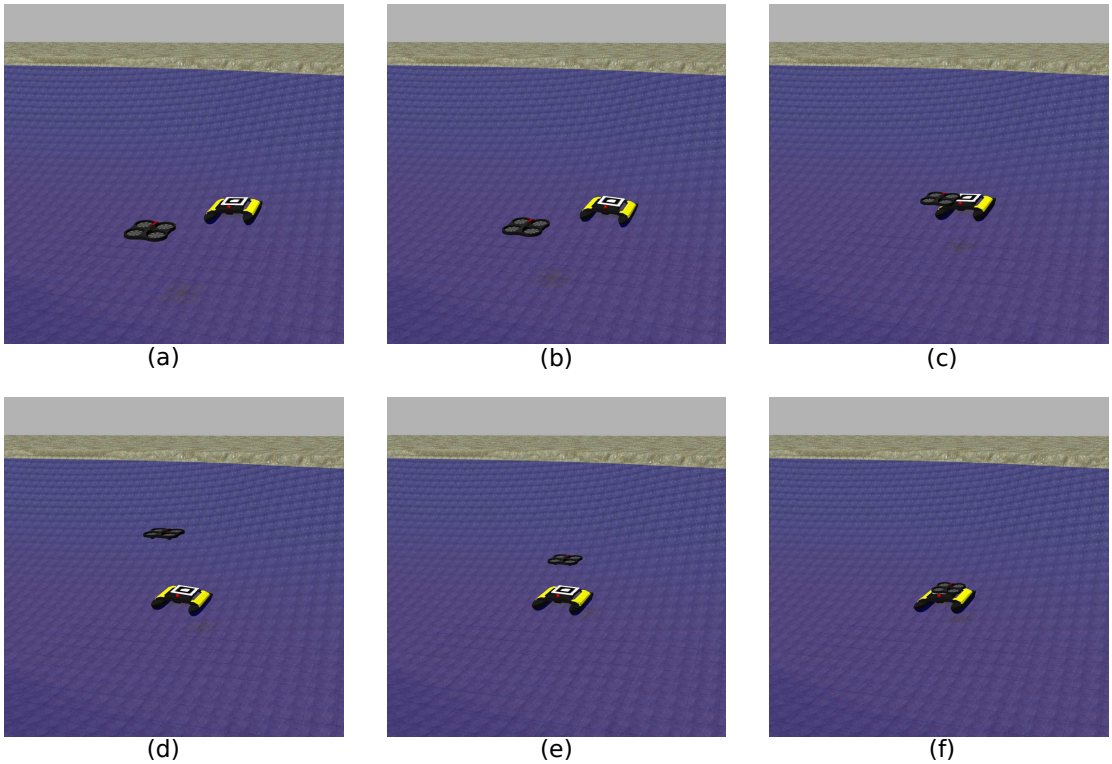


Figure 4.19: Landing manoeuvre of a vertical take-off and landing (VTOL) UAV on a USV subject only to rolling disturbances. The drone approach the deck first using its frontal camera (a - b) until the marker is not visible anymore (c). At this point, the altitude of the UAV is increased (d) while the bottom downward-looking camera is used for the tracking of the marker (e) and accomplish the landing manoeuvre (f).

looking down, and new commands are generated and sent. The UAV is instructed to move towards the last known position of the landing platform but increasing its altitude in order to enlarge the area covered by its bottom camera. At $t = 15s$, as represented in Fig. 4.19-b, the UAV is located exactly above the marker and it can now complete the landing phase: it descends while trying to keep the marker at the centre of its FOV, as shown in Fig. 4.19-c. Small velocity commands are sent on the leaning direction (x and y, respectively) in order to approach the final position with high accuracy. Finally, at $t = 25s$ the UAV reaches the minimum altitude required to shut-down its motors and land on the platform (Fig. 4.19-f).

The commands generated from the relative-pose between the UAV and the landing platform's frame are illustrated in Fig. 4.17. Here, the controller's commands are plot-

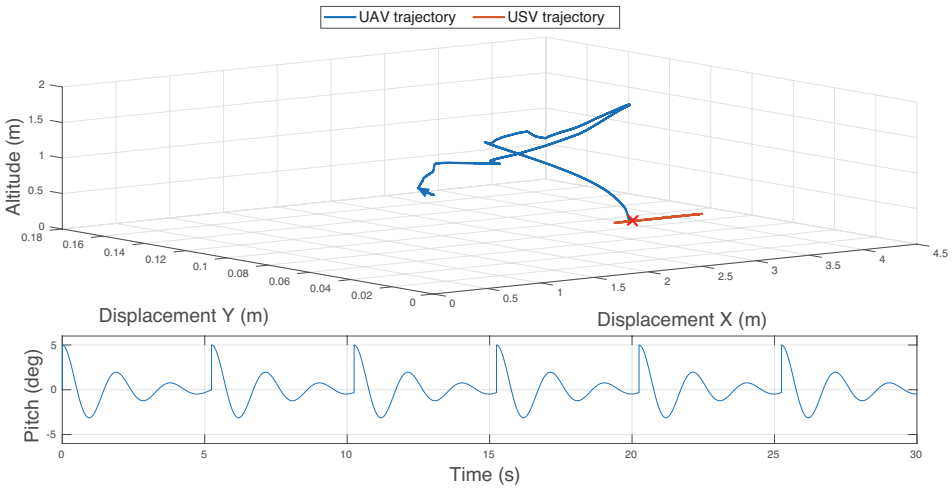


Figure 4.20: (Top) The UAV and USV 3D trajectories, in blue and red, respectively, in the UAV's reference frame. (Bottom) The pitch disturbances the USV is subjected.

ted against the perception from the camera. As it is possible to see in the figure, for most of the travel the two curves of the commands and of the observations overlap perfectly. When they do not, the marker is lost and the UAV actuates the compensatory behaviour: the estimation filter's output, namely the USV's predicted pose, is combined with the latest vision observation in order to generate new commands for the UAV. In this way it is possible to explain changes in roll, pitch and altitude in the graph. Since the UAV has the same yaw of the floating base, namely they have the same orientation along the z-axes, no rotation commands are issued for this degree of freedom.

Few words are reserved for the pitch's data between $t = 18s$ and $t = 22s$, and the gaz's ones between $t = 5s$ and $t = 8s$. In this case, the offsets are below a user-defined threshold and a null command is sent instead. The use of a threshold has been introduced for speeding up the landing phase: while testing the controller, it was noticed the UAV spent a lot of time while trying to align perfectly on the three axis with the centre of the landing plane, sometimes without any success. This has been identified as a limitation of controllers with fixed values parameters and a new more versatile solution is already planned as future work.

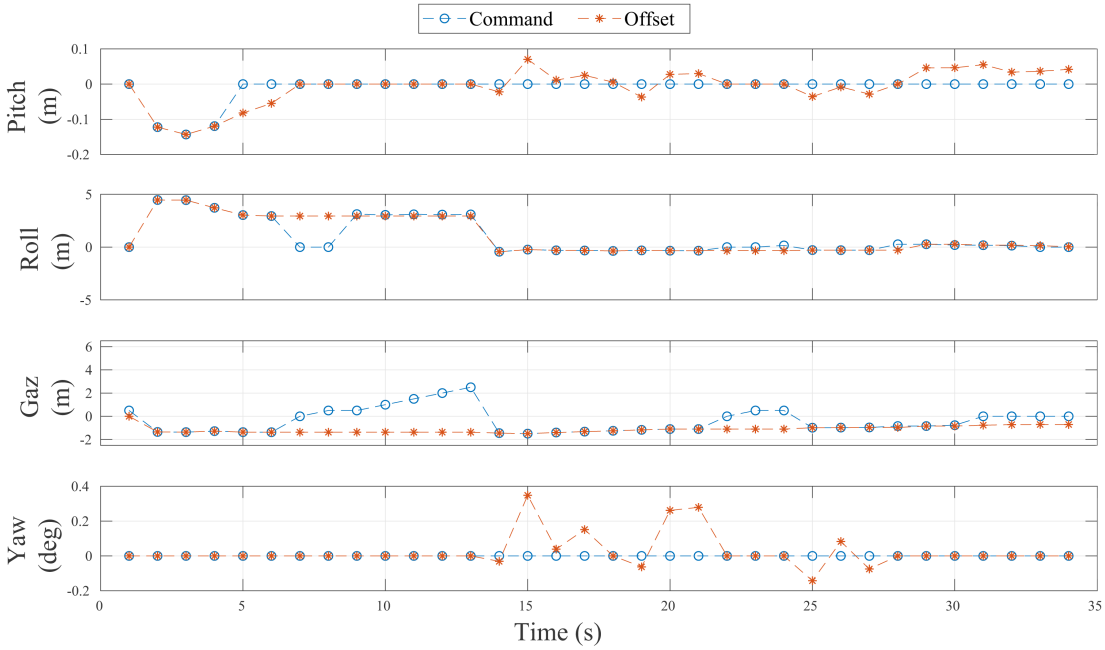


Figure 4.21: Controller commands and visual offsets in the experiment with a pitching landing platform.

4.2.2.2 Pitching platform

Here an experiment with a pitching floating platform is reported. As before, the time for completing the landing manoeuvre is not considered as key-factor but the attention is on the ability of the UAV to approach and land on the USV with high precision.

As in the previous experiments, the two vehicles 3D trajectory are reported in Fig. 4.20 in the UAV's reference frame, the controller commands in Fig. 4.21, the UAV's profile in Fig. 4.22 and example frames in Fig. 4.23. The quadcopter, with the same controller parameters of before, was able to follow and land on the visual marker in almost 34 s after identifying it 4.46 m ahead and 0.12 meter on its left.

As in the case of a rolling base, Fig. 4.23-a shows the UAV starts moving in order to keep the visual marker at the centre of its frontal camera's field of view. This is what happens at time $t = 26s$ and shown in Fig. 4.23-b. At $t = 6s$ the UAV reaches its minimum altitude and it is now impossible for it to see the visual marker, as illustrated in Fig. 4.23-c. At this point, the video stream starts to be acquired from the bottom

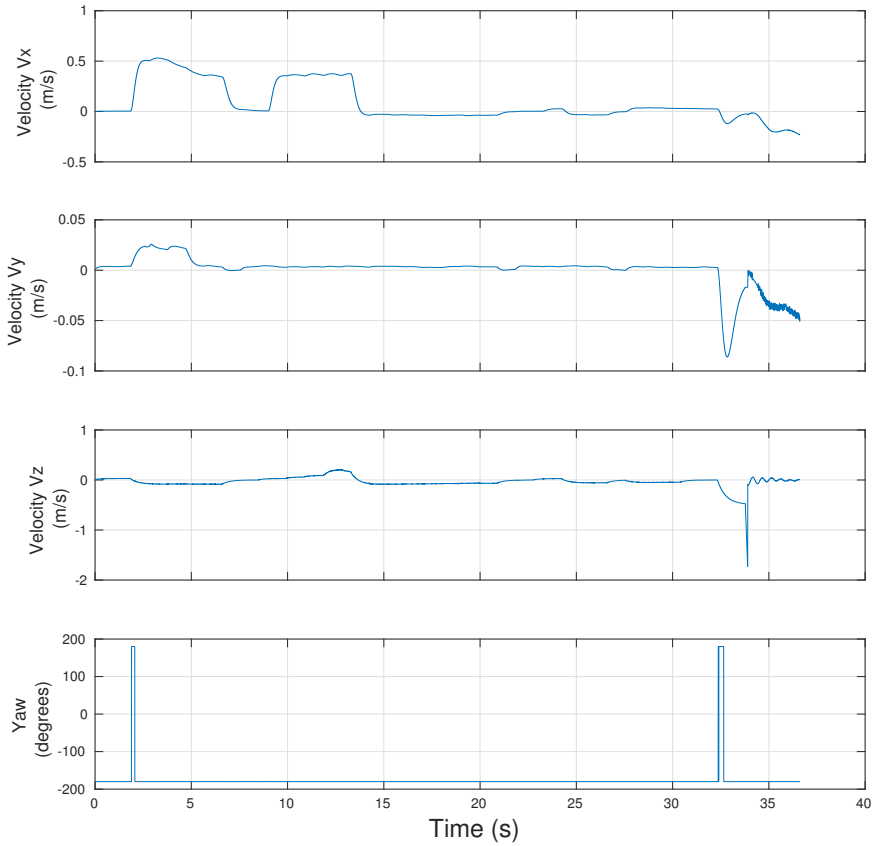


Figure 4.22: Quadcopter's velocity and yaw profile during the experiment with a pitching platform.

camera and the USV's estimated position is sent to the controller. At the same time, instructing the UAV to increase its altitude to augment the total area covered with its downward-looking camera. Doing this, at $t = 13s$ the UAV is located exactly above the USV. The landing base is at the centre of the camera's FOV, therefore a null velocity command is sent to stop the USV. Fig. 4.23-e and 4.23-f show the UAV can then descend slowly to centre the marker properly and, in the end, land on it.

Further analysis has been done with the results reported in Fig. 4.21. In the same way of the experiment with a rolling deck, the curve of the controller's commands and the one related to the offsets overlap for most of the time. All the considerations made before still hold: while the marker is lost, the EKF is able to estimate the landing platform's

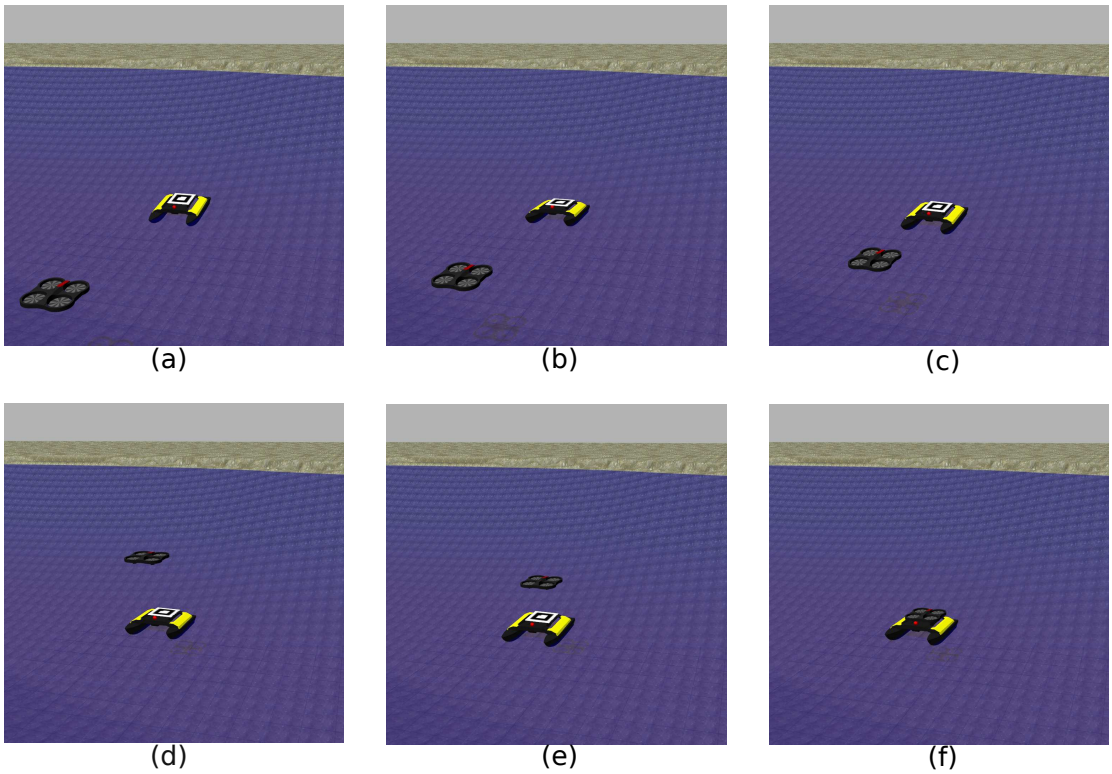


Figure 4.23: Landing manoeuvre of a VTOL UAV on a USV subject only to pitching disturbances. The drone approaches the deck first using its frontal camera (a - b) until the marker is not visible anymore (c). At this point, the altitude of the UAV is increased (d) while the bottom downward-looking camera is used for the tracking of the marker (e) and accomplish the landing manoeuvre (f).

current pose with reference to the instant of time when the marker has been lost. This relative-pose is added to the last observation in order to produce a new command. This is what is possible to see in the plot between $t = 21s$ and $t = 25s$. Here, the two curves differ: while all the offsets remain constant because no new marker observations have been done by the UAV, the commands (gaz and roll) slightly change. The plot is now discussed in more details. While the yaw and the pitch commands remain identical to 0 because the UAV is already aligned with the landing base (within the predefined bounds), the UAV's roll command is changed including at every instant the new relative-pose (changing on the longitudinal direction) of the USV.

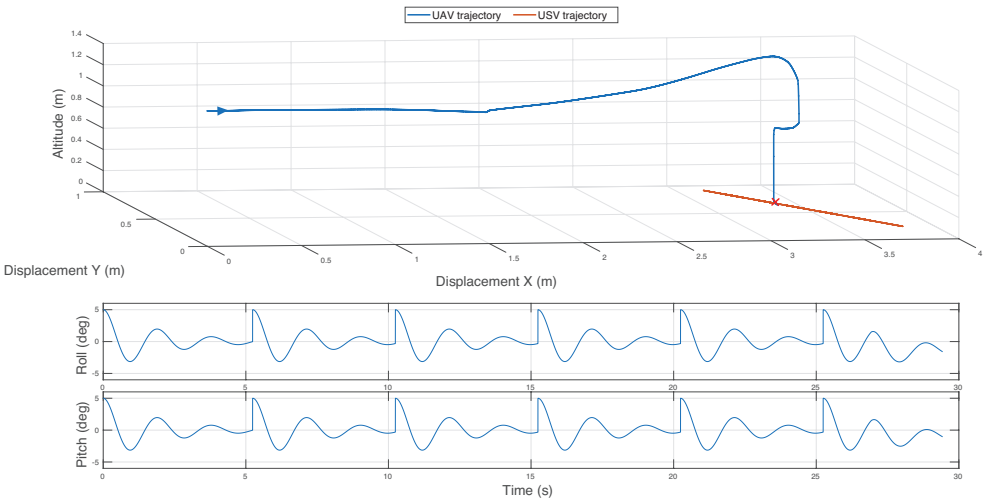


Figure 4.24: (Top) The UAV and USV 3D trajectories, in blue and red, respectively, in the UAV's reference frame. (Bottom) Both the roll and pitch disturbances the USV is subjected.

4.2.2.3 Rolling and pitching platform

Finally a simulation study has been conducted with a floating platform that is subject to both rolling and pitching stresses. The goal of this experiment is to test the developed landing algorithm against simulated harsh marine conditions.

The results are reported in Fig. 4.24, showing that both vehicles trajectories along a 23 s operation, while the vehicle's profile is shown in Fig. 4.26. The UAV successfully accomplish the landing manoeuvre starting from an initial marker's identification 3.71 m in front of it and 0.30 m on its left. Fig. 4.25 shows the comparison between the offsets obtained through the vision algorithm and the commands sent to the controller. It is possible to see that, as in the previous experiments, the curve of the offsets and the one related to the commands mainly overlap. All the analysis made before are still valid, but it is interesting to notice how the framework proposed is able to react properly also when the landing platform is subject to complex disturbances. The salient moments of the flight are illustrated in Fig. 4.27

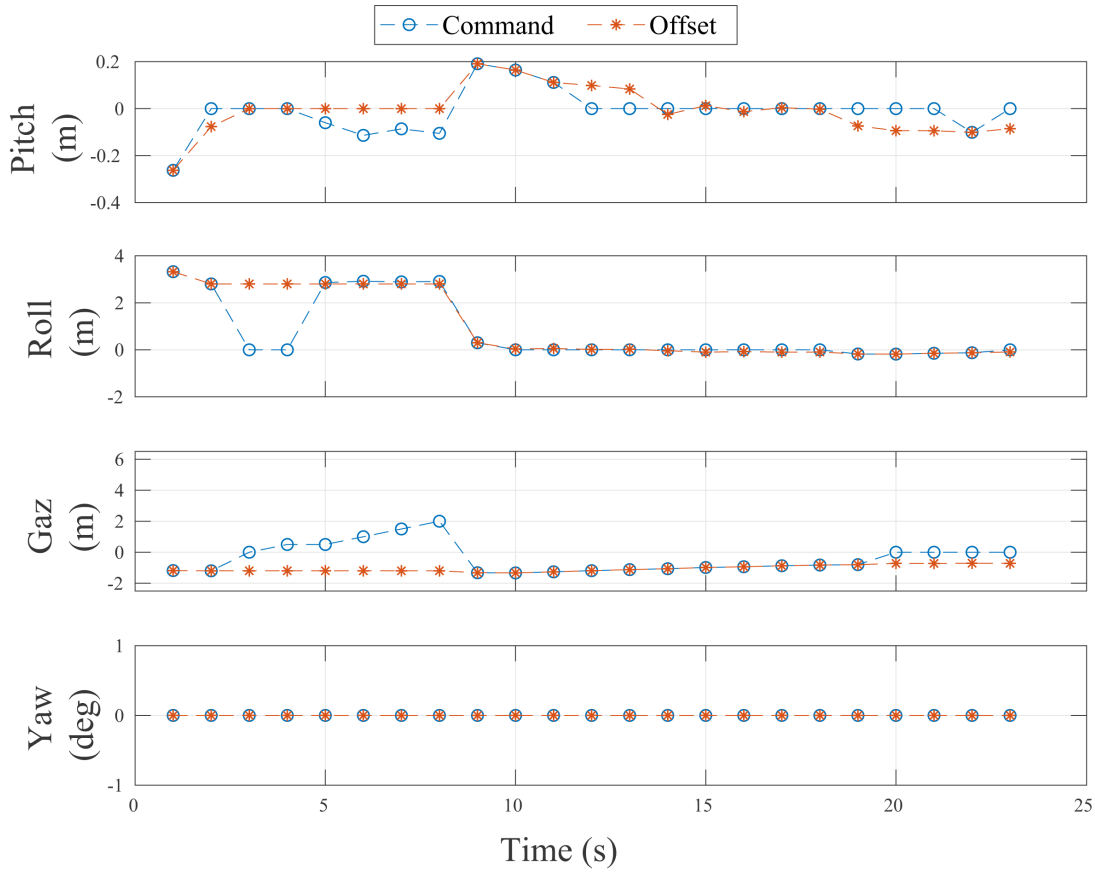


Figure 4.25: Controller commands and visual offsets in the experiment with a pitching and rolling landing platform, in order to simulate complex marine scenarios.

4.3 Conclusion

In this chapter, a solution to make an unmanned aerial vehicle to autonomously land on a flat surface of an autonomous vehicle is presented. It relies only on the UAV's on-board sensors and on the adoption of a visual marker on the landing platform. In this way, the UAV can estimate the 6-DOF landing area position through an image processing algorithm. The adoption of a pose estimation filter - in this case an extended Kalman filter - allows overcoming issues with fixed non-tilting cameras and the image processing algorithm. Not involving GPS signals in the pose estimation and in the generation of flight commands, allows the UAV to land also in situations where this signal is not available (indoor scenario or adverse weather conditions).

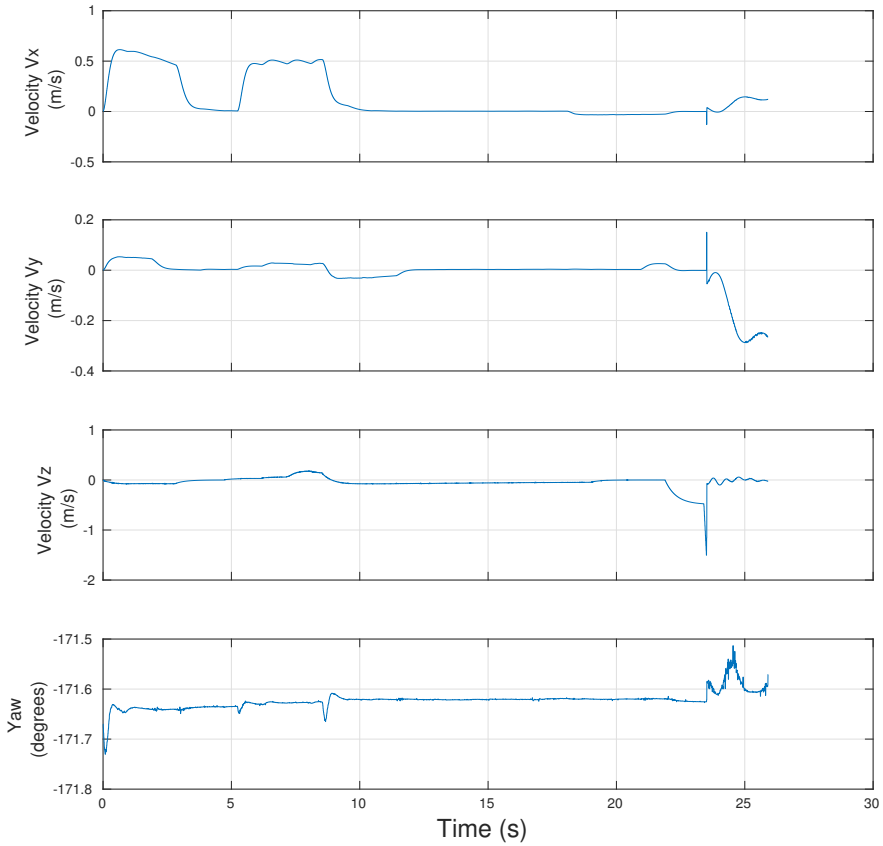


Figure 4.26: Quadcopter's velocity and yaw profile during the experiment with a floating platform subject to both roll and pitch.

The validation of the approach has been done in the simulation with a quad-rotor and two autonomous vehicles, a mobile ground robot and an unmanned surface vehicle, used as a platform on which to land. Various simulated experiments were performed, each of them with a different type of disturbance or motion acting on the landing base. In all scenarios, successful results were obtained.

The algorithm described in this chapter heavily relies on the ar_pose vision library for identifying the landing pad at distance. But the changes in attitude of the marker itself can pose various difficulties to the detection. In the next chapter, a new solution based on deep reinforcement learning (DRL) is described. Empowering deep neural network, it is possible to robustly detect the marker at long distance, without suffering due to a

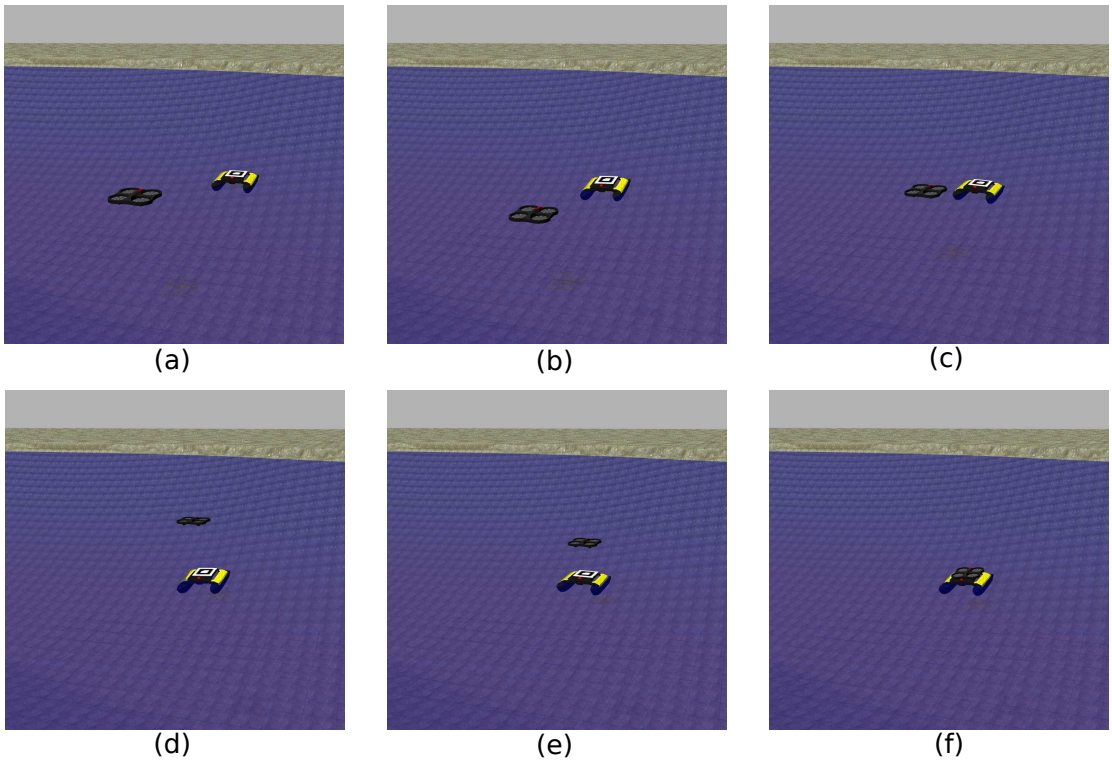


Figure 4.27: Landing manoeuvre of a VTOL UAV on a USV subject to both rolling and pitching disturbances, in order to simulate complex marine scenarios. The drone approaches the deck first using its frontal camera (a - b) until the marker is not visible anymore (c). At this point, the altitude of the UAV is increased (d) while the bottom downward-looking camera is used for the tracking of the marker (e) and accomplish the landing manoeuvre (f).

geometric projective transformation. Moreover, the real advantage of DRL is to have a unique model that performs object detection and control at the same time. Therefore, the network not only is trained to identify the marker within the camera's field of view, but is also able to redirect the UAV in the proximity of the marker and land on it.

Chapter 5

Autonomous Landing using Deep Reinforcement Learning

FOLLOWING the limitations of the method described in the previous chapter, mainly related to the hand-tuning of the controller, the current chapter illustrates a new solution to the autonomous landing problem employing deep learning techniques. In particular, combining artificial neural networks with reinforcement learning a UAV is taught to search for the landing platform and then land on it without any prior knowledge but only using exploratory behaviours. More importantly, there is no human intervention during the learning process. [1]

5.1 Introduction

In this work, a divide-and-conquer approach is adopted to split the landing task into two sub-tasks: marker detection and descend manoeuvre. Two specialised Deep Q-Networks (DQNs) (Mnih et al. 2015) are responsible for addressing each phase of the flight and are connected through an internal trigger callable by the network itself. To solve the problem of sparse reward, the concept of partitioned buffer replay is introduced, dividing the experiences based on the reward values and guaranteeing the presence of rare transitions at training time. An overview of the system is shown in Figure 5.1. The results obtained in the testing part showed that this method achieves performance comparable or even superior to an average human agent or a state-of-the-art solution such as the one already presented in Chapter 4.

[1] A video showing the working principle of this approach is at the following link: https://youtu.be/WZNV2h6vXxc

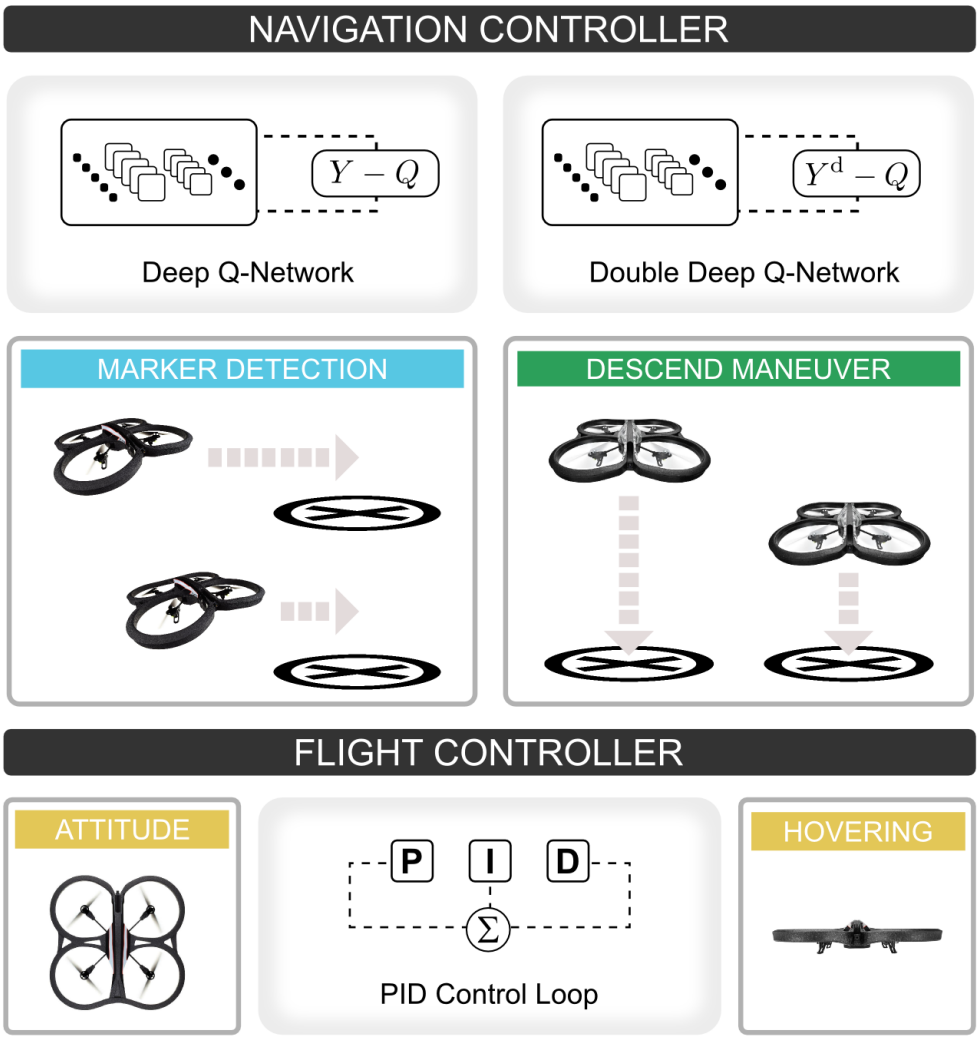


Figure 5.1: System overview. The navigation controller is built on top of the flight controller. The marker detection and the descent manoeuvre are achieved through two distinct DQNs.

5.2 Proposed method

In this section, the UAV landing issue is described in terms of reinforcement learning. Also, the technical solutions adopted are introduced and discussed.

5.2.1 Problem definition

There is a limited amount of work trying to address the landing problem using reinforcement learning, in particular deep reinforcement learning. This is due to the fact the use

of UAVs introduces many complications. In fact, most of the existing literature on DRL is focused on solving the well known Atari 2600 games suite, including popular games such as Breakout, Ms. Pacman and Space Invaders. Atari 2600 has been a challenging test-bed due to its high-dimensional video input (size 210 x 160, frequency 60 Hz) and the discrepancy of tasks between games. The Atari games offer a low-resolution 2D graphics with a pretty simple physical model. Differently, UAVs are real vehicles moving in our world and this poses many complications when developing a reinforcement learning solution. First of all, the real world is characterised by textures which are more detailed by those adopted in any video games. Even when using high definition cameras (equal or superior to the full high-definition resolution of 1920 x 1020) for taking pictures or recording videos, the camera is actually sub-sampling the resolution of the world. Dealing with high definition images means requiring more computational power and longer training time. The second problem associated with the real world is its three-dimensional nature. Having an additional dimension along which an agent can move, makes exploring the environment extremely time-consuming. Moreover, this also affects the distribution of the reward which can be more sparse than what it would be in a simple 2D environment, resulting in a more difficult and complex learning task. As of last, the complex structure of our world, involving people, objects, building and plants leads to occlusion and its partial observability. The drone can see only a small portion of the environment and it has to deal with projective transformation. Another complication is related to the platform itself. Differently from a virtual agent which moves in a discrete fashion along "railways", a UAV has six degrees of freedom, meaning that it can translate and rotate along three axes. Moreover, its motion is continuous in an interval defined between 0 and its maximum velocity. This makes the control part much more complex, especially when adopting a neural network as function approximator for learning the dynamical model. It should be now clear how complicated and tedious is developing a robot learning task compared to playing a video game. For this reason, applying DQN to the landing problem is challenging, even though it has already obtained state-of-the-art performances in two-dimensional games.

In this work, the landing problem is considered and divided into two sub-problems: marker detection and vertical descent. The first of the two requires an exploration on the xy-plane, where the UAV has to align its body frame with the one of the marker located on the deck of the USV. In contrast, during the vertical descent phase, the UAV has to reduce its altitude while keeping the marker centred. Since the introduction of a third spatial dimension leads to larger state space and sparse reward, the vertical descent is the more challenging phase. Two independent Deep Q-Networks have been trained separately for performing successfully the two tasks and integrated with a single state-machine. The first network has been trained to identify the marker at a fixed altitude of 20 m and guide the UAV above it. The DQN can stop the vehicle or move it in four directions (forward, backward, left, right). Moreover, it can call on an internal trigger which activates the second network. The new policy can stop and move the vehicle in five directions (forward, backward, left, right and down). It aims to control the UAV while vertically descending up to 1.5 m above the marker. The last module of the system is represented by a closed loop controller which slowly reduces the power of rotors until the touchdown. The closed loop mechanism allows fine control of the vehicle in the very last meter and can rely on distance sensors to safely land and turn off the UAV.

5.2.2 Notation

Formally, both the marker detection and vertical descent problems can be modelled as Markov Decision Processes (MDPs). In this study, the transition model and the reward functions are not given (model free problem). In this work, two deep neural networks are used as function approximators following the same approach presented in Mnih et al. (2015). The deep networks used take as input four 84×84 grey-scale images acquired by the downward looking camera mounted on the UAV, which are processed by three convolutional layers and two fully connected layers. The rectified linear unit (Glorot et al. 2011) is used as activation function. The first convolution has 32 kernels of 4×4 with a stride of 2, the second layer has 64 kernels of 4×4 with a stride of 2,

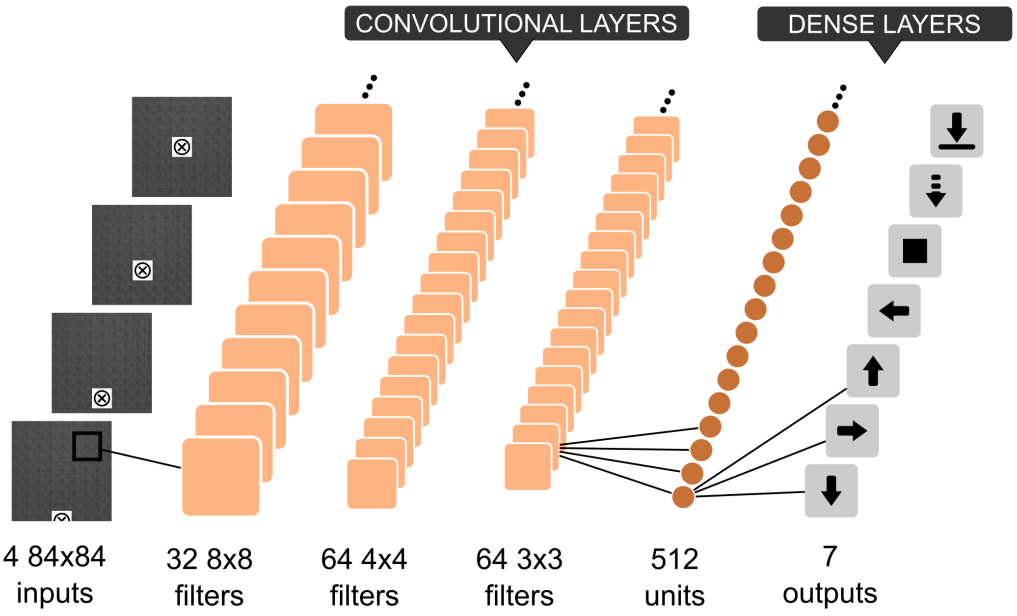


Figure 5.2: Graphical representation of the DQN. The network takes in input four 84×84 images, and generates as output the 7 actions: forward, right, backward, left, stop, descent, trigger.

the third layer convolves 64 kernels of 3×3 with a stride 1. The fourth layer is a fully connected layer of 512 units followed by the output layer which has a unit for each valid action (backward, right, forward, left, stop, descend and the trigger). The unit with the highest associated value is selected as next action for the UAV. Depending on the simulation, only a sub-set of the total actions is available (please refer to Section 5.3 for additional details). A graphical representation of the network is presented in Figure 5.2, while an example of the input images and the output of the first layer of kernel is provided in Figure 5.3.

It is particularly important to focus on the two phases (marker detection and vertical descent) that characterise the landing problem in order to isolate obstacles that may be encountered. The landmark detection phase has a large state space. If the UAV is generated within a bounding box of size $15m \times 15m \times 20m$ and it has 20 time steps of approximately 1 meter per episode, it means that the agent has a box of $35m \times 35m \times 40m$ to explore. This huge volume can be reduced quite considerably, if it is assumed to fly at fixed altitude. In this case, the vertical alignment with the landmark can be

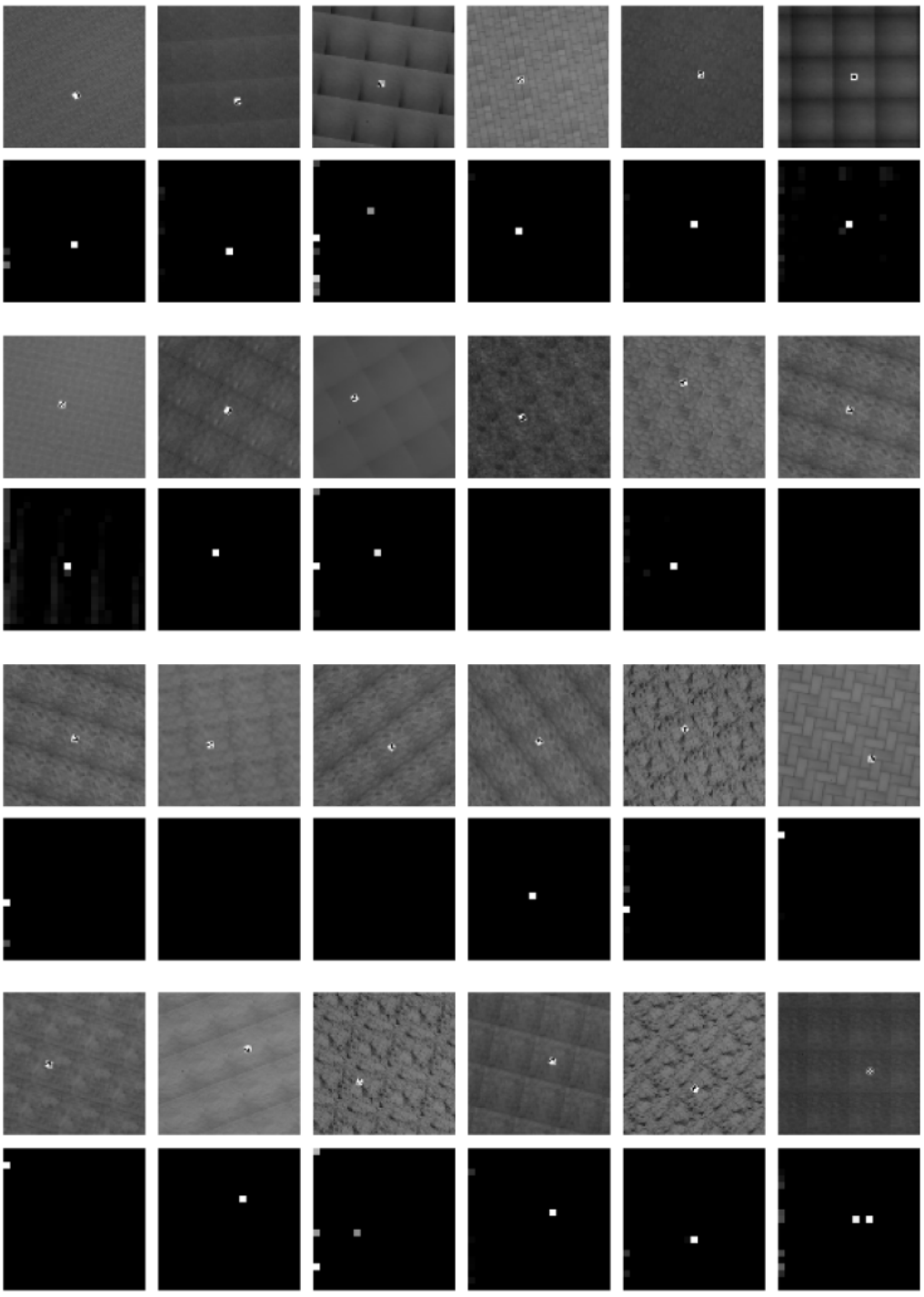


Figure 5.3: The grey-scale images given as input to the DQN and the feature map generated by the kernels in the first convolutional layer.

done using only shifts on the xy-plane. This expedient does not have any impact on the operation level but it dramatically simplifies the task, reducing the volume to explore to a flat slice of size $35m \times 35m \times 1m$. Additional details on the state space are given in Section 5.3.

The vertical descend phase can be considered a form of Blind Cliffwalk Schaul et al. (2015) where the agent has to take a right action (descend) in order to progress through a sequence of n states. At the end of the walk, the agent can obtain either a positive or a negative reward. In the landing scenario, the structure of the problem makes extremely difficult to obtain a positive reward because the target-zone is only a small portion of the state space. The consequence is that the buffer replay does not contain enough positive experiences, making the policy unstable. To resolve this issue it is possible to use a different form of a buffer which encourage to replay important transitions more frequently, as proposed in (Schaul et al. 2015). A new type of buffer replay is therefore introduced and called *partitioned buffer replay*, that discriminates between rewards and guarantees a fair sampling between positive, negative and neutral experiences. Further details about the model implementation and the hyper-parameters used are discussed in Section 5.3. The overestimation, already discussed in Section 3.7.2, is another problem associated with the reward sparsity. During preliminary research, it was observed that this phenomenon arose in the vertical descent phase. To address this problem, the double DQN has been used instead of the vanilla version. During the preliminary research it was experienced that the Q-max value (the highest utility returned by the Q-network) largely overshot the maximum utility of 1.0, which was correctly associated with the trigger (Figure 5.4). As a result, the UAV moved on top of the marker and then randomly shifted on the xy-plane without engaging the trigger. This is because the trigger leads to a terminal state of the MDP and it does not use the max operator for updating its utility.

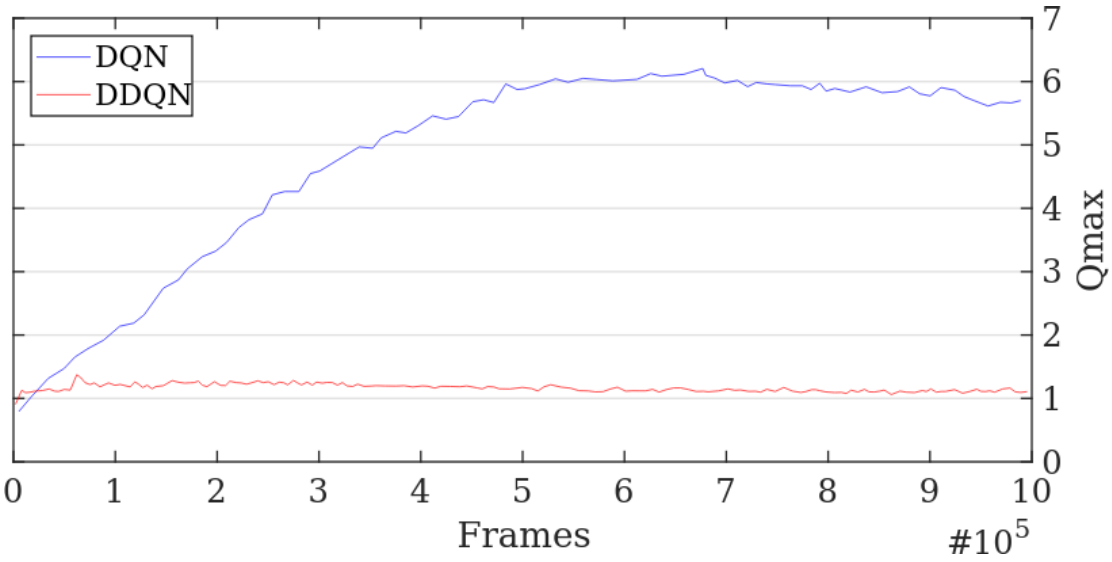


Figure 5.4: Comparison between the Q-max value (the highest utility returned by the Q-network) obtained with the classic DQN and the revised Double DQN. The overestimation of the Q-function prevents the DQN to converge to the real maximum utility of 1.0 and learn how to accomplish the task.

5.2.3 Partitioned buffer replay

In MDPs with a sparse and delayed reward it is difficult to obtain positive feedback. In these cases the experiences accumulated in the buffer replay may be extremely unbalanced. Neutral transitions are frequent and for this reason they are sampled with a high probability, whereas positive and negative experiences are uncommon and more difficult to sample. A preliminary study confirmed the vertical descent is affected by the sparsity of positive and negative rewards, leading to an underestimation of the utilities associated with the triggers. A solution for sparse reward was proposed in Narasimhan et al. (2015), dividing the experiences into two buckets depending on their priority (high or low). This work has been extended to K buckets. Another form of prioritized buffer replay has been proposed in Schaul et al. (2015). The authors suggest sampling important transitions more frequently. The prioritized replay estimates a weight for each experience based on the temporal difference error. Experiences are sampled with a probability proportional to the weight. The limitation of this form of prioritization is the introduction of another layer of complexity that may not be justified for applications

where there is a clear distinction between positive and negative rewards. Moreover this method requires $O(\log N)$ to update the priorities. This issue does not significantly affect performances on the standard benchmark but it has a relevant effect on robotics application, where there is a high cost in obtaining experiences.

In Section 5.2.2 it has been defined $D = (e_1, ..., e_t)$ being a dataset of experiences $e = (s, a, r, s')$ used to uniformly sample a batch at each iteration i. To create a partitioned buffer replay, the reward space needs to be divided into K partitions:

$$R = R(s,a) \rightarrow \text{Im } R \ = R_1 \cup ... \cup R_K \tag{5.1}$$

For any experience e_i and its associated reward $r_i = r(e_i)$, it is possible to define the K-th buffer replay:

$$D_K = \{(e_1, ..., e_N) : r_1, .., r_N \in R_K\} \tag{5.2}$$

The batch used for training the policy is assembled picking experiences from each one of the K datasets with a certain fraction $\rho \in \{\rho_1, ..., \rho_K\}$. In this work $K = 3$, meaning that there are three datasets with D^+ containing experiences having positive rewards, D^- containing experiences having negative rewards, and $D^\sim$ for experiences having neutral rewards. The fraction of experiences associated with each one of the dataset is defined as ρ^+, ρ^-, and $\rho^\sim$.

When using a partitioned buffer replay there is a substantial increase in the available number of positive and negative experiences. For instance, using a single buffer of size 2×10^4 and accumulating 8.4×10^4 transitions, the total number of positive experiences is 343 and the number of negative experiences is 2191. Using a partitioned buffer with size 2×10^4 for the neutral partition, and size 10^4 for positive and negative partitions, the total number of positive experiences is 1352 and the number of negative experiences 9270.

5.2.4 Hierarchy of DQNs

The method proposed is based on the use of a hierarchy of DQNs representing sub-policies used to deal with different phases of the navigation. Similarly to a finite-state machine, the global policy is divided into modules and each module is governed by a specific DQN or control loop. The DQNs are able to autonomously understand when it is time to call the next state. The advantages of such a method are twofold. On the one hand it is possible to reduce the complexity of the task using a divide-and-conquer approach. On the other hand, the use of a function approximator is confined in specific sandboxes making their use in robotic applications safer.

This approach is inspired by hierarchical reinforcement learning (Barto & Mahadevan 2003), whereby sub-policies control the agents within a subset of a core MDP. The options control the agent in sub-regions of a core MDP called semi-MDPs. In the present work, the overall landing task is mapped to the core MDP which is split into multiple simplified MDPs. In this way, standard Q-learning is still an effective algorithm for teaching the agent to learn how to solve the task. Moreover, in this work it has been assumed that it is possible to combine the auxiliary MDPs in an ordered sequence and connect the elements of the sequence through shared states. In shared states the use of particular actions, called triggers, enables the passage to the next MDP. The triggers are additional actions that do not belong to the action space of the core MDP. Engaging the triggers in shared states leads to a reward that is equal to the maximal reward associated with the core MDP. How to define the auxiliary MDPs is an operation left to the designer that requires some knowledge of the core MDP.

The finite MDP describing the landing problem can be divided into three main stages: landmark detection, descending manoeuvre, touchdown. The first two phases are described in Section 5.2.1 and 5.2.2. The touchdown consists in decreasing the power of the motors in the last few centimetres of the descent, then safely deactivate the UAV components (e.g. motors, cameras, boards, control unit, etc.) after landing. The focus is only on the first two stages, because they represent the most challenging part of

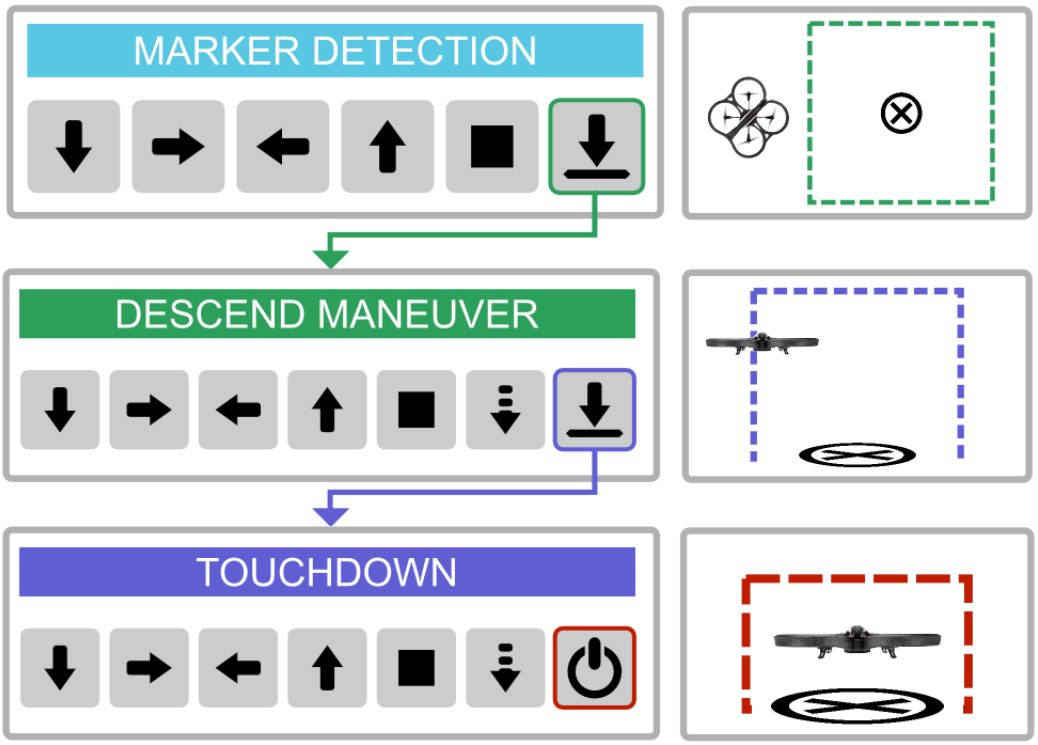


Figure 5.5: Finite-state machine for autonomous landing based on the DQN hierarchy method. Each state has a specific trigger that enables the DQN in the next stage.

the landing procedure. A graphical representation of a hierarchical state machine is represented in Figure 5.5. The first DQN (marker detection) was trained to receive a positive reward when the trigger was enabled inside a target area. Negative reward was given if the trigger was enabled outside the target area. The second network (descent manoeuvre) was trained using the same idea. In a preliminary phase a single network was trained to achieve both detection and descending. Given the size of the combined spaces the network was not able to converge to a stable policy. As a baseline it is also reported the accumulated reward curve of this network in Section 5.3.

5.2.5 Practical issues and safety constraints

The use of UAV in DRL arises practical issues that must be considered here. First of all, DRL requires the collection of a large number of experiences. The collection of those experiences can be extremely time consuming in real world applications, and it may require to fly for hundreds of hours with a UAV. The major problem here is that

commercial UAVs have limited autonomy which depends on different factors: payload, wind, propellers and motors quality. The maximum autonomy is on average 10 mins and in particular 15 mins for the Parrot AR Drone 2, meaning that in a practical scenario would be necessary to change the battery very often. Changing the battery means to securely land the UAV, turn it off, switch the battery pack and take off. It is clear that this is a major limitation which must be handled in future research.

Another problem with UAVs is due to safety constraints. UAVs must always be flown in a safe manner with respect to other aerial vehicles, people and property. In this sense, many countries recently adopted specific laws for ensuring safety measures. Q-learning requires a large amount of exploration to obtain a stable value-action function. In some conditions, exploration can lead to undefined behaviour and this is particularly dangerous when UAVs are used. Training the algorithm in real world would mean to find a safe area for the tests, to obtain the authorisation necessary for the flights, and to have a certified pilot who can take control in case of need. Given all these limitations, it is reasonable to start investigating the problem using a simulator instead of moving directly to the real world. In a simulated environment the UAV can operate freely without any safety constraint. The simulator gives the possibility to speed-up, interrupt and restart the training. Moreover there are no autonomy constraints. In the next section the simulated environment used in this work will be described.

5.2.6 The simulated environment

Previous work in DRL was followed by the release of an open-source library for increasing the research in the field, such as the Arcade Learning Environment (Bellemare et al. 2015) including the Atari 2600 games and the OpenAI Gym (Brockman et al. 2016) consisting in high level interfaces for classic problems (e.g. gridworld, pole balancing, mountain car, etc) and more complex ones (e.g. board games, humanoid walker, Doom the videogame, etc). Unfortunately, at the current stage there is no standard environment for developing and testing UAV landing algorithms. To support the simulated experiments conducted, it has been worked on a simulated environment

based on the Gazebo simulator (Koenig & Howard 2004) and ROS (Robot Operating System) (Quigley et al. 2009). Gazebo is well known within the robotics community, thanks to its ability to simulate complex robots, its flexibility given by the possibility of changing run-time parameters and four different physics engines. The simulator is called Quadrotor Landing Benchmark (QLaB) and it is released under an open source license[2]. QLaB includes configuration files, textures and examples which allow the user to easily train and test new algorithms. The texture dataset consists of more than a hundred patches, 70% used for training and 30% for testing. The patches are high-definition pictures representing common ground floors belonging to eight categories: asphalt, brick, grass, pavement, sand, snow, soil and water. The *world* is a 100×100 m floor which can be covered by a uniform texture. Using a variety of floor it is possible to verify the generalization capabilities of the algorithm in different conditions.

To guarantee challenging conditions, it was added support for standard flight and disturbed flight, where drift is injected in the velocity of the vehicle on the x-y-z components, and on the z rotational axis. The drift consists of a scalar sampled from a uniform distribution in the range $[-0.03., 0.03]$ (meter per seconds) and accumulated for a time interval of 5 seconds. The drift represents challenging conditions where inaccuracies in the navigation measurements and external environmental factors (e.g. wind) can significantly deteriorate the precision of the movements. The trajectory of the UAV can be heavily affected by the drift. For example, the altitude can oscillate of ± 0.25 m when the UAV is moving on a straight line at a constant speed on the xy-plane. A representation of the drift is presented in Figure 5.6.

The UAV used in the environment is a widely diffused commercial quad-rotor, more precisely the Parrot AR Drone 2, previously described in Chapter 4.

The simulator allows training and testing in two cases:

1. Landmark detection. The vehicle is generated at a fixed altitude inside a bounding box of size $15m \times 15m \times 20m$. To obtain a positive reward the UAV has to find

[2]https://github.com/pulver22/QLAB

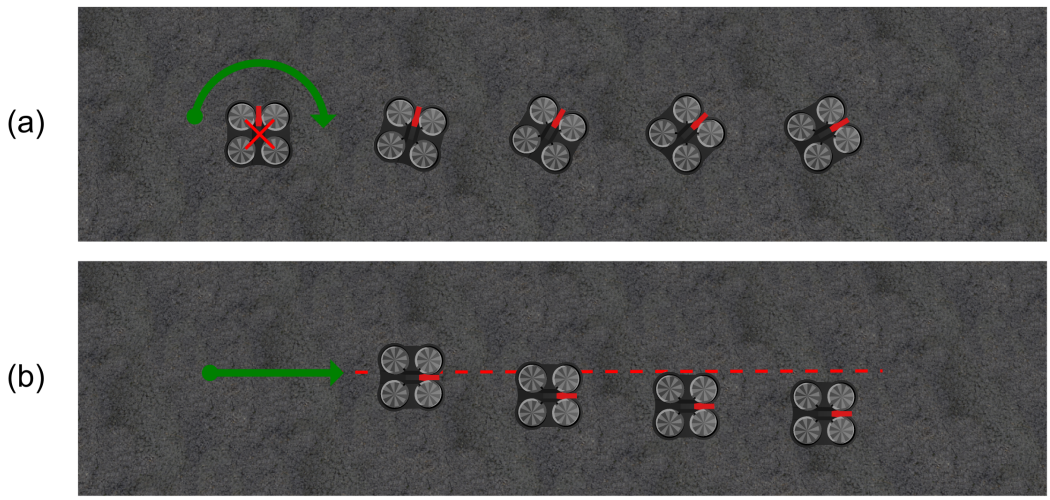


Figure 5.6: Illustration of the vehicle drift. In (a) is represented the drift on the z axis, which is particularly visible when the UAV does not move. In (b) is represented the drift on the xy-plane of the UAV moving on a straight line (dashed red).

the landmark and moves over it in a target-zone of size $3m \times 3m \times 20m$ centred on the landmark.

2. Vertical descent. The UAV is generated in a small bounding box of size $3m \times 3m \times 20m$ and it obtains the positive reward only when entering in a small target-zone of size $0.75m \times 0.75m \times 1.5m$ centred on the landmark.

To standardise the tests, different parameters have been defined: number of attempts per texture, time limit, sensors available and marker type. For each one of the tests a total of 100 landing manoeuvres must be attempted. A time limit of 40 s (20 steps) is applied in the landmark detection test, whereas a time limit of 80 s (40 steps) is applied in the vertical descent test. When the episode finishes, a new one is started and the UAV is randomly generated inside the large bounding box. The only sensor used is the downward looking camera, which can be read at a user-defined frame rate and resolution. Since there is not a standard marker, it has been decided to take the one used in the first challenge of the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) (Dias et al. 2016) competition. In this challenge, 23 teams coming from all around the world competed to make a UAV land on the top of a vehicle moving along an

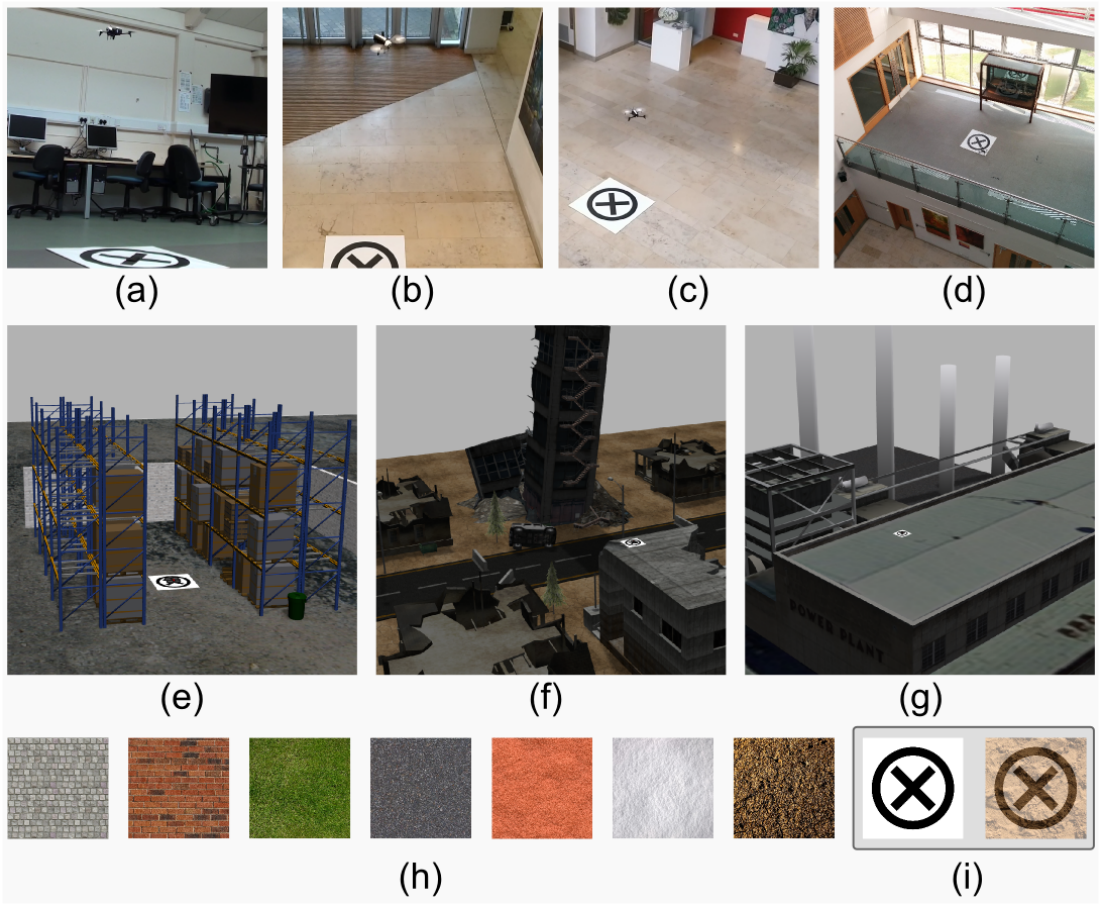


Figure 5.7: Real environments: laboratory (a), small hall (b), large hall (c), mezzanine (d). Photo-realistic environments: warehouse (e), disaster site (f), powerplant (g). Textures (h): pavement, brick, grass, asphalt, sand, snow, soil. Marker and corrupted marker (i).

8-shape path in the shortest amount of time and with the highest accuracy as possible. On the top of the moving vehicle, it was located a 1-meter white square containing a black circle and a cross. Finding this marker could be challenging because it does not contain any distinctive colour. Moreover, its shape could complicate future extraction due to the absence of well defined corners.

5.2.7 Training through domain randomization

The reality gap is the obstacle that makes it difficult to implement many robotic solutions, developed in simulation or in a constrained lab environment, in the real world. This is especially true for DRL where a large number of episodes (namely interaction

with the environment) is necessary in order to obtain stable policies.

Recent research worked on bridging this gap using domain transfer techniques. An example is *domain randomization* (Tobin et al. 2017), a method for training models on simulated images that transfer to real images by randomising rendering in the simulator. Here, domain randomization was adopted in order to train the UAV in simple simulated environments and test it in complex environments, both simulated and real. Few examples of the environments used for training and testing are given in Figure 5.7. The remarkable property of this approach is that it does not require any pre-training on real images. If the variability is significant enough, models trained in simulation generalise to the real world with no additional training or fine-tuning. In the next session it will be show how domain randomization has been included in the training phase and how the experiments have been organised.

5.3 Experiments

This section presents the methodology and the results obtained in training and testing the DQNs used in the proposed system. Preliminary research has been performed in order to highlight the core problems in both tasks (Section 5.3.1). In Section 5.3.2 is presented the methodology and the results obtained in a series of simulations performed to train the DQN to specialise in the landmark detection phase. In Section 5.3.3 is presented the second series of simulation which have been performed to train the policy for the vertical descent phase.

5.3.1 Preliminary research

In the preliminary phase, it was trained the DQNs in both landmark detection and vertical descent using a naive approach, in order to see the weights that different variables have on the results achieved. Moreover, it was tested the influence of different hyperparameters on the training through grid-search. The results obtained in this phase represent an important baseline which gives a reference point for all the other simulations.

5.3.1.1 Description of the task

The *world* was represented by a uniform asphalt texture of size $60m \times 60m$ with the landmark positioned in the centre. A bounding box of size $20m \times 20m \times 20m$ centred on the marker was used to generate the UAV at the beginning of each episode. A smaller bounding box of size $6.96m \times 6.96m \times 6.96m$ (approximately 4% of the total volume) represented the target-zone. A negative reward of -1.0 was given when the UAV touched the ground outside the valid area and when it overstepped the large bounding box. In all these cases, the episode finished and a new one started. A positive reward of 1.0 was given when the UAV entered in the target-zone represented by the small bounding box. A negative cost of living of -0.01 was used as constant punishment in all the other cases.

The action-space was discretised in nine valid actions: backward, right, forward, left, rotate counter-clockwise, rotate clockwise, stop, descend and ascend. The action was repeated for 0.25 seconds and represented a velocity command of 0.5 meters per second. All the movements were affected by inertia and noise.

The buffer replay was filled before the training with 4×10^5 frames using a random policy. The training phase ran for 1.4×10^6 frames, taking approximately 8 days to complete. It was used an ε-greedy policy with ε decayed linearly from 1.0 to 0.1 over the first 500k frames and fixed at 0.1 thereafter. The discount factor γ was set to 0.99. As an optimizer, it was used the RMSProp algorithm (Tieleman & Hinton 2012) with a batch size of 32. The weights were initialised using the Xavier initialisation method (Glorot & Bengio 2010). The DQN algorithm was implemented in Python using the Tensorflow library (Abadi et al. 2016). Simulations were performed on a workstation with an Intel i7 (8 core) processor, 32 GB of RAM, and the NVIDIA Quadro K2200 as the graphical processing unit. The training phase took around 6 days with the physic engine running at real time speed.

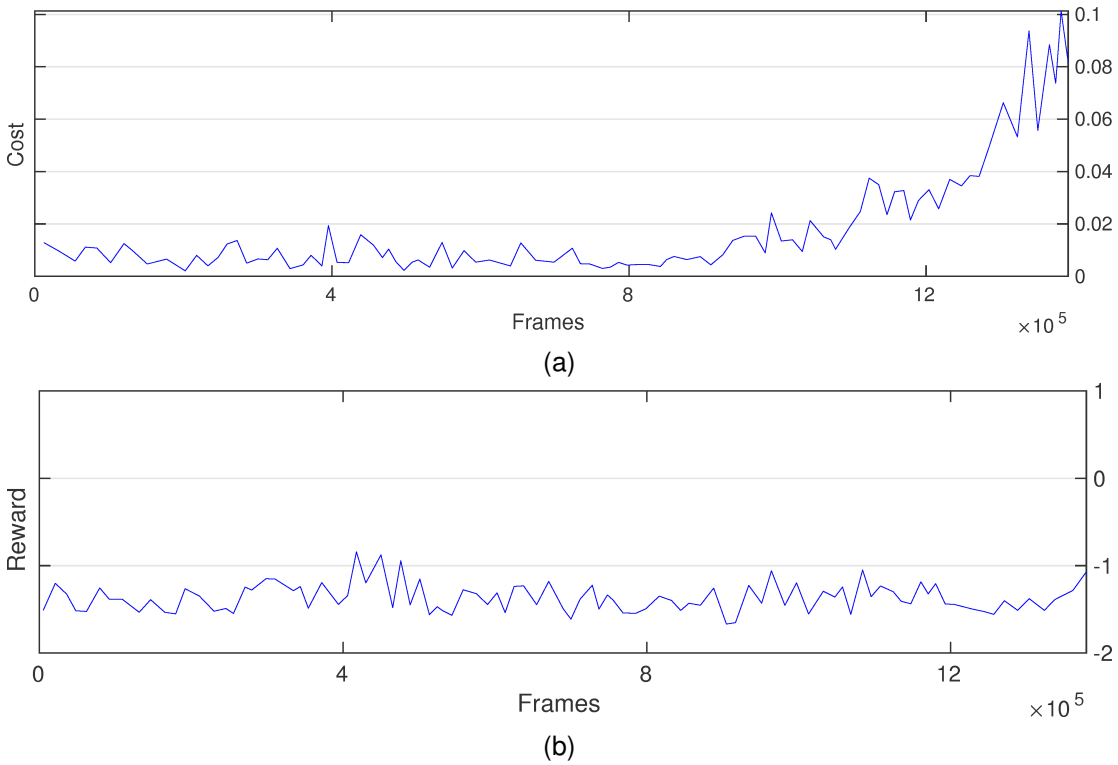


Figure 5.8: Results of the preliminary research in which the marker detection and descending manoeuvre were not splitted in two sub-problems. The cost (a) did not significantly decrease, whereas the accumulated reward per episode (b) remained constant along the training.

5.3.1.2 Results

The results of this simulation showed that the UAV could not find a valid policy for reaching the marker (Figure 5.8). The cost decreased in the first episodes, remaining constant until frame 1×10^6, after this point it started oscillating rapidly and eventually diverging. Analyzing the results for accumulated reward (Figure 5.8b) there is clearly no improvement.

The physics of the system made extremely challenging to apply DRL to the landing problem. An oscillatory effect during acceleration and deceleration made the UAV swinging on the roll and pitch axes (Figure 5.9a). This effect generated artefacts in the image acquisition of the down-looking camera. Moreover a summation of forces effect introduced a substantial shift in the trajectory (Figure 5.9b). The shift increased the

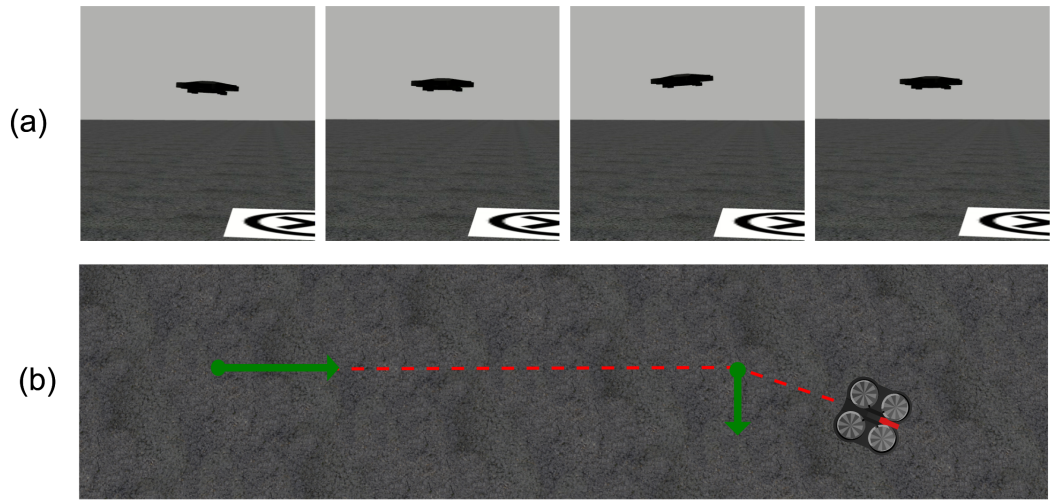
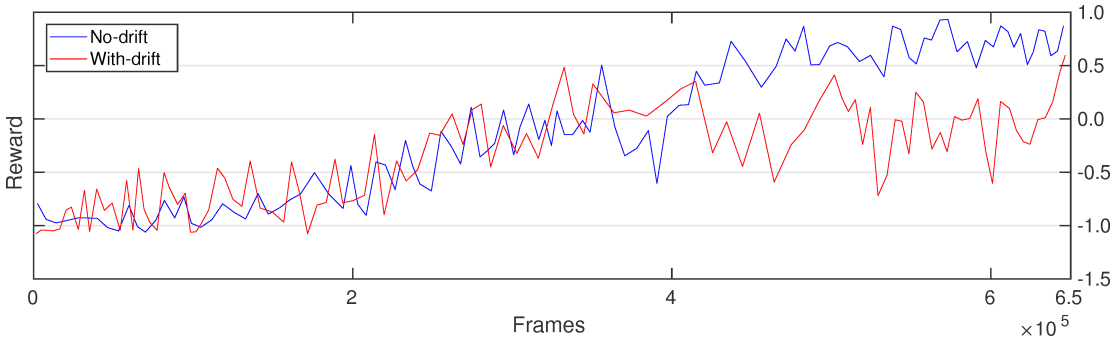


Figure 5.9: Illustration of the vehicle inertia. In (a) is represented the oscillatory inertia for roll and pitch, which is visible when the UAV starts and stops. In (b) is represented the direction of the UAV moving in a straight line (dashed red) and the diagonal trajectory acquired due to the addition of two forces (green arrows). The two forms of inertia are part of the physical engine and were present in standard and drift conditions.

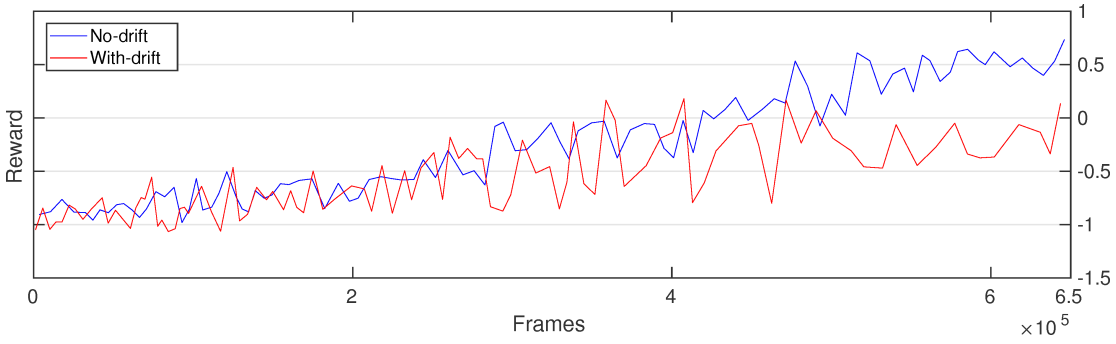
complexity of the action space introducing an high variance. hłIt is possible to see in Figure 5.10 that the effects introduced by the drift have a huge impact on the training time, taking much more time than in a situation without these effects. Even if it does not represent a realistic scenario, it was decided to deactivate the drift within QLaB in order to speed up the convergence time. As it will be shown later, the network demonstrated to be robust enough once tested in the real world.

It is possible to summarize the main problems found during the preliminary research in two points:

1. Inertia and drift in the UAV movements. The roll and pitch oscillations of the UAV (Figure 5.9a) caused the acquisition of corrupted frames. This was due to the projection of the marker on the low-resolution image plane, which generated some artefacts. Moreover, the summation of forces (Figure 5.9b) and the drift (Figure 5.6) injected considerable noise in the action space adding another level of complexity.



(a)



(b)

Figure 5.10: Convergence time of a DQN trained with or without the drift in the simulator, on a single texture (a) or on a variety of textured randomly sampled every 50 episodes (b).

2. Large state space and sparse reward. In the landing phase, the reward is obtained only at the end of the descent. The vehicle requires a temporally extended exploration in order to obtain the positive feedback.

In the next sections, it will be explained how those problems were handled, obtaining robust policies for both landmark detection and vertical descent.

5.3.2 Marker detection

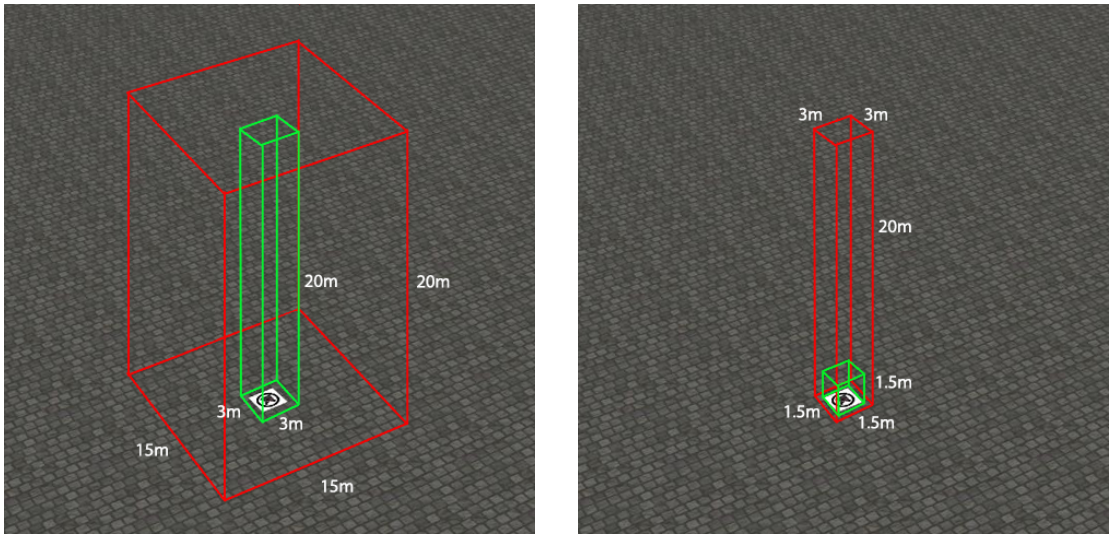
In the first series of simulations, it was trained and tested the DQNs for the marker detection phase. Three networks having the same structure (Figure 5.2) were considered and trained in three different conditions. The first network was trained with a uniform asphalt texture (DQN-single), whereas the second network was trained with multiple textures (DQN-multi) and the third network on marine textures (DQN-marine). The ability to generalise to new unseen situations is very important for a neural net-

work and it should be seriously taken into account in the landing problem. Training the first network on a single texture is a way to quantify the effect of a limited dataset on the performance of the agent. In the DQN-multi condition, the networks were trained using 70 textures divided into seven different groups: asphalt, brick, grass, pavement, sand, snow, soil (Figure 5.7-h). These networks should outperform the ones trained in the condition with a single texture. For DQN-marine condition, the training has been performed using ten textures belonging to the marine class and characterised by light reflection and sea foam.

At each episode, the UAV started at a fixed altitude of 20 m that was maintained for the entire flight. This expedient was useful for two reasons: it significantly reduced the state space to explore and it allowed visualising the marker in most of the cases giving a reference point for the navigation. In a practical scenario this solution does not have any impact on the flight, the UAV is kept at a stable altitude and the frames are acquired regularly. To stabilise the flight discrete movements were introduced, meaning that each action was repeated for 2 s and then stopped leading to an approximate shift of 1 m, similarly to the no-operation parameter used in Mnih et al. (2015). The frames from the camera were acquired between the actions (at a frequency of 0.5 Hz) when the vehicle was stationary. This expedient stabilised convergence reducing perspective errors.

5.3.2.1 Description of the task

The training of the network is performed inside the QLaB simulator. The *world* is represented by a uniform texture of size $100m \times 100m$ with a fixed non-moving marker located at the centre (Figure 5.7). At the beginning of each episode, the UAV was generated at an altitude of 20 m inside the perimeter of the large bounding box ($15m \times 15m \times 20m$) with a random position and orientation (Figure 5.11). A positive reward of 1.0 was given when the UAV activated the trigger in the target-zone, and a negative reward of -1.0 was given if the UAV activated the trigger outside the target-zone. A negative cost of living of -0.01 was applied to all the other conditions. A time limit of 40



(a) (b)

Figure 5.11: Flying-zone (red) and target-zone (green) for landmark detection (a) and vertical descent (b).

Table 5.1: Statistics for the agents trained on marker detection.

Agent type	**Synchronization step**	**Buffer Replay size**	**Training steps**
DQN-single	10000	0.4×10^5	0.71×10^6
DQN-multi	10000	0.4×10^5	6.5×10^5
DQN-marine	30000	0.4×10^5	6.5×10^5

seconds (20 steps) was used to stop the episode and start a new one. The texture was modified every 50 episodes and randomly sampled between all those available within the training set.

The agent had five possible actions available: forward, backward, left, right and the trigger. The action was repeated for 2 seconds then the UAV was stopped and a new action was sampled. Details about the synchronisation step, buffer replay size and training time (expressed as the number of steps required before convergence) for the three networks trained are given in Table 5.1.

In order to control the exploration-exploitation dilemma, it was used an ε-greedy policy with ε decayed linearly from 1.0 to 0.1 over the first 500k frames and fixed at 0.1 thereafter. The discount factor γ was set to 0.99. As an optimizer, the RMSProp algorithm with a batch size of 32 was used. The weights were initialised using the

Xavier initialization method.

The DQN algorithm was implemented in Python using the Tensorflow library. Simulations were performed on a workstation with an Intel i7 (8 cores) processor, 32 GB of RAM, and the NVIDIA Quadro K2200 as the graphical processing unit. The training phase took around 3 days for all the agents. In all the cases, the physic engine running five times the real time speed.

5.3.2.2 Results

To test the performance of the policies, two groups of tests were identified. In the first group it was measured the detection success rate of both DQN-single and DQN-multi against a static landing pad, while in the second group the performance of DQN-marine was evaluated in landing on an unmanned surface vehicle subject to simulated sea currents.

The first group comprises six individual testing scenarios. (i) The first test was performed on 21 unknown uniform textures belonging to the same seven categories of the training set. (ii) The second test was done on the same environments but at different altitudes (20 m, 15 m, and 10 m). (iii) The third test was performed on the same 21 unknown textures but using a marker corrupted through a semi-transparent dust-like layer. (iv) The fourth test was done randomly sampling 25 textures from the test set and mixing them in a mosaic-like composition. (v) The fifth test has been done on three photo-realistic environments namely a warehouse, a disaster site, and a power-plant (Figure 5.7-e/g). (vi) The sixth and last test consisted of a real-world implementation in the mezzanine environment (Figure 5.7-d). The mezzanine is the only environment that allowed flying at a high altitude approximately of 15 m. It was also measured the performances of a random agent, the AR-tracker algorithm described in Chapter 4, and human pilots in all the simulated environments. All the human candidates were selected among other students, mainly PhD in computer science or psychology. They were men and women of age comprises between 23 and 35 years old. No one of them had any previous experience in flying drones. The human participants were supplied

with only low-resolution grey-scale images as those used in input for the networks. The frame rate between two consecutive images was kept fixed at 0.5 Hz and a time limit was applied to each episode to better compare the results with the DQNs. The human data has been collected using two methodologies. In the first approach, 7 volunteers used a space-navigator mouse that gave the possibility to move the UAV in the three dimensions at a maximum speed of 0.5 m/s. In the second methodology, 5 volunteers used a keyboard to move the UAV in four directions on the xy-plane through discrete steps of 1 m. The first methodology has been adopted in order to give to the subjects a natural control interface, whereas the second methodology gave the same control conditions of the UAV. In both conditions preliminary training (10 episodes) allowed the subject to familiarize itself with the task. After the familiarization phase the real test started. In the landmark detection, the subjects had to align the UAV with the ground marker and then trigger the landing procedure when inside the target-zone. The subjects performed five trials for each one of the environments contained in the test set (randomly sampled). A time limit of 40 seconds (20 steps) was applied to each episode. A landing attempt was declared as failed when the time limit expired or when the subject engaged the trigger outside the target-zone.

The second group of tests comprises four individual test cases, positioning the marker on the deck of an unmanned surface vehicle whose motion is affected by different simulated sea conditions. (i) The first test was performed in a static condition, in which the USV remains still in the same position, without its deck being subject to any change in attitude. (ii) In the second test, the USV motion is affected by a rolling condition, in which the USV oscillates around its x-axes in the range $[-5,5]$ deg. (iii) The third test was performed in the pitching condition, with the USV subject to perturbations that make it rotating along the y-axis in the range $[-5,5]$ deg. Finally, in the fourth test (iv), the USV's motion is affected by combined roll and pitch perturbations acting on both axis. A graphical representation of the training and testing conditions for the marine experiments is provided in Figure 5.12.

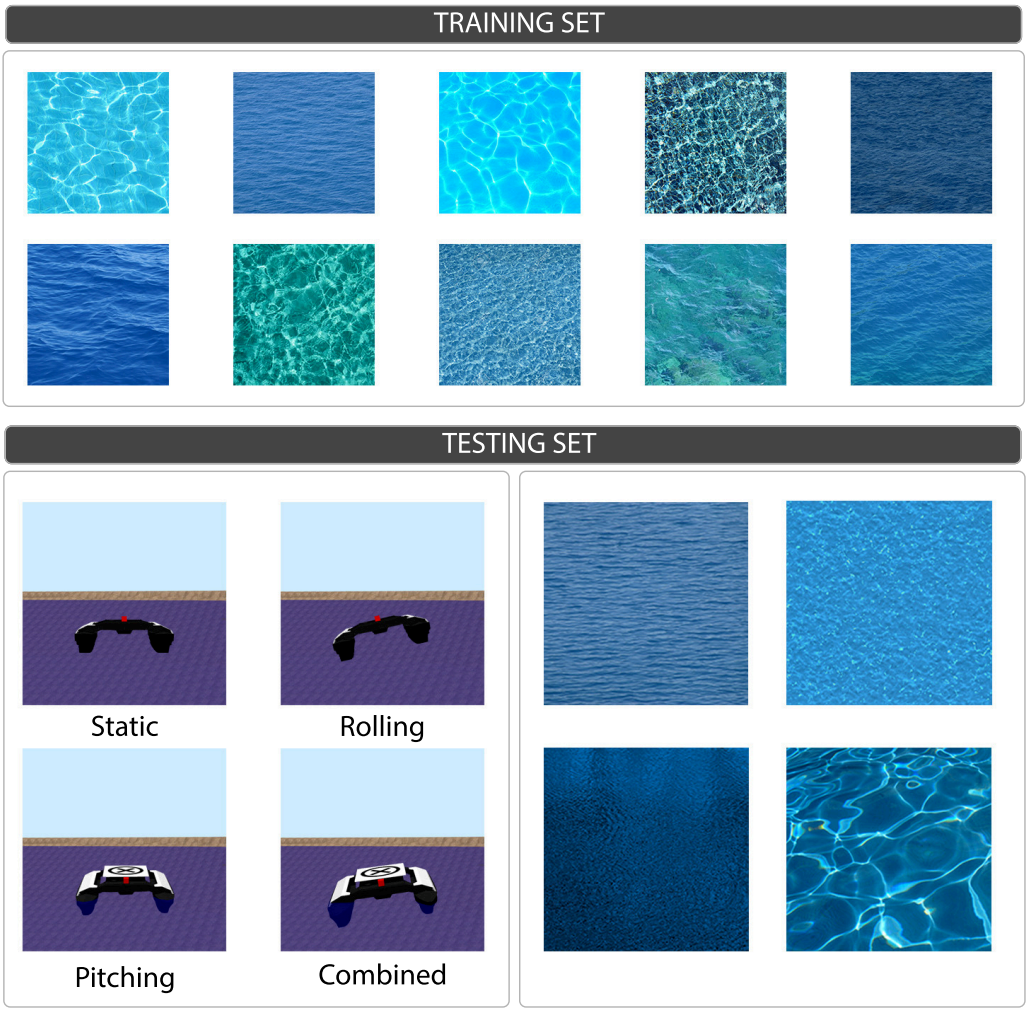


Figure 5.12: Textures used during the training (top) and testing (bottom) phases in the simulations with a floating marker. Four are the conditions in which every agent has been tested: with a static marker, with a rolling deck, a pitching one and when the deck is subject to both rolling and pitching combined.

Static marker The results for both DQN-single and DQN-multi show that the agents were able to learn an efficient policy for maximizing the reward. In both conditions, the reward increased stably without any anomaly (Figure 5.13 bottom). In the same figure it is also reported the reward curve for a baseline condition, where a single network has been trained to perform both detection and descending. The reward of the baseline did not increase significantly and the resulting policy was unable to engage the trigger inside the target-zone.

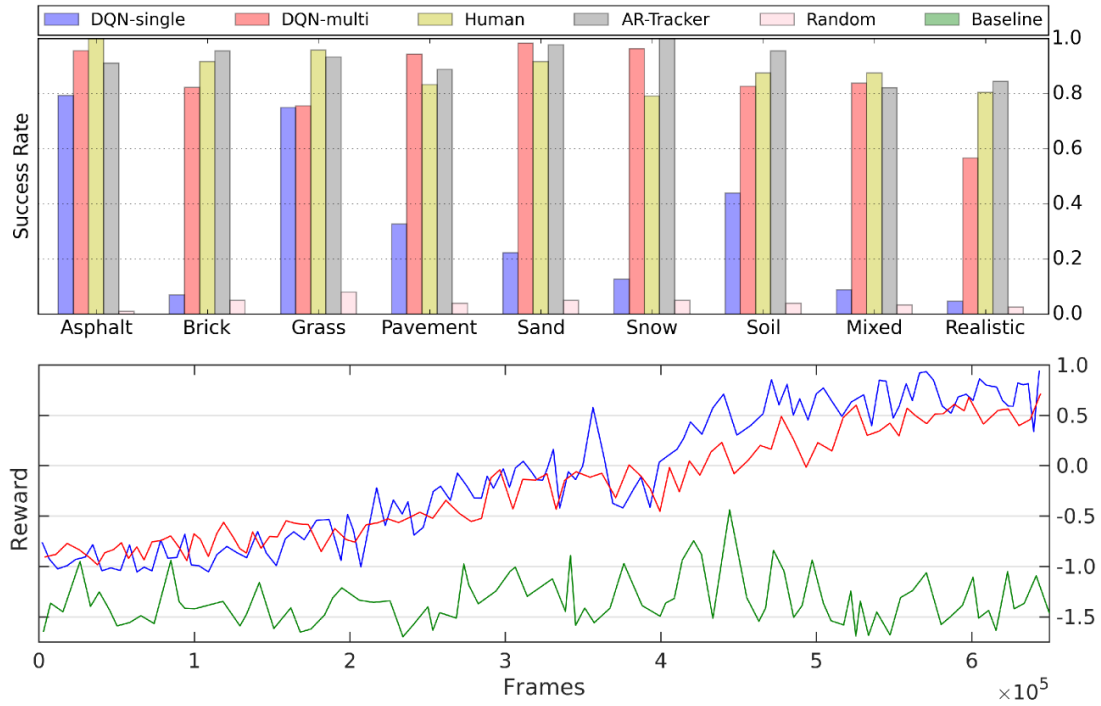


Figure 5.13: Results of the marker detection simulations with a static marker. (Top) Detection success rate. (Bottom) Accumulated reward per episode for DQN-single (blue line), DQN-multi (red-line), and baseline (green-line).

The results of the test phase are summarized in Figure 5.13 (top). The bar chart compares the performances of DQN-single, DQN-multi, human pilots, the AR-tracker and a random agent. For human pilots it is only reported the results for the discrete control condition, since the score was higher than the space-navigator condition ($+6\%$). The average score on the first test (uniform textures) for the DQN-multi is 91%. The score obtained by the agent trained on a single texture (DQN-single) is significantly lower (39%). The human performance is 90%, whereas the AR-tracker has an average score of 95%. The random agent has an average reward of 4% in this environment. Since both human pilots and DQNs used discrete steps to move in the environments, it is possible to estimate the average number of discrete steps required to accomplish the detection. For human pilots the average number of steps is 12, whereas for the DQN-multi is 6, meaning that humans were significantly slower.

Testing the DQN-multi at different altitudes it was noticed that the accuracy increased at 15 m (95%) and 10 m (93%), with respect to the accuracy at the training altitude of

Table 5.2: Landmark detection overall performances on uniform textures, mixed textures and realistic environment using a static marker. The performance is the average percentage across agents of correct landings. Standard deviation is in brackets. The best scores are in bold.

Agent type	Performance
DQN-single	$0.32(0.27)$
DQN-multi	$0.85(0.12)$
AR-Tracker	**0.92 (0.06)**
Random	$0.4(0.18)$
Humans	$0.88(0.06)$

20 m (89%). This result is explained by the fact that at lower altitudes the marker is more visible.

In the third test, the DQN-multi and AR-tracker were compared on uniform textures using the corrupted marker. It was observed a significant drop in the AR-tracker performances from 94% to 0% explained by the fact that the underlying template matching algorithm failed in identifying the corrupted marker. In the same condition, the DQN-multi performed well, with a limited drop in performance from 89% to 81%, confirming the better robustness of artificial neural networks in object detection compared to classic computer vision algorithms.

The results in the fourth test (mixed-textures) showed a lower performance for all the agents. DQN-multi has a success rate of 84% and the DQN-single of 9%. The human pilots have a performance of 88% and the AR-tracker of 82%.

The results of the fifth test (photo-realistic environments) showed a generic drop (DQN-multi=57%, DQN-single=5%, Human=81%, Random=3%, AR-tracker=84%).

The overall performance on uniform textures, mixed textures and realistic worlds is reported in Table 5.2 and it is 85% for DQN-multi, 32% for DQN-single, 88% for human pilots, and 92% for the AR-tracker.

Finally, DQN-multi was tested on the sixth test, controlling the UAV in a real-world environment, more precisely the mezzanine in Portland Square building at the University of Plymouth. The results showed an overall accuracy of 50% on a total of 10 flights. It

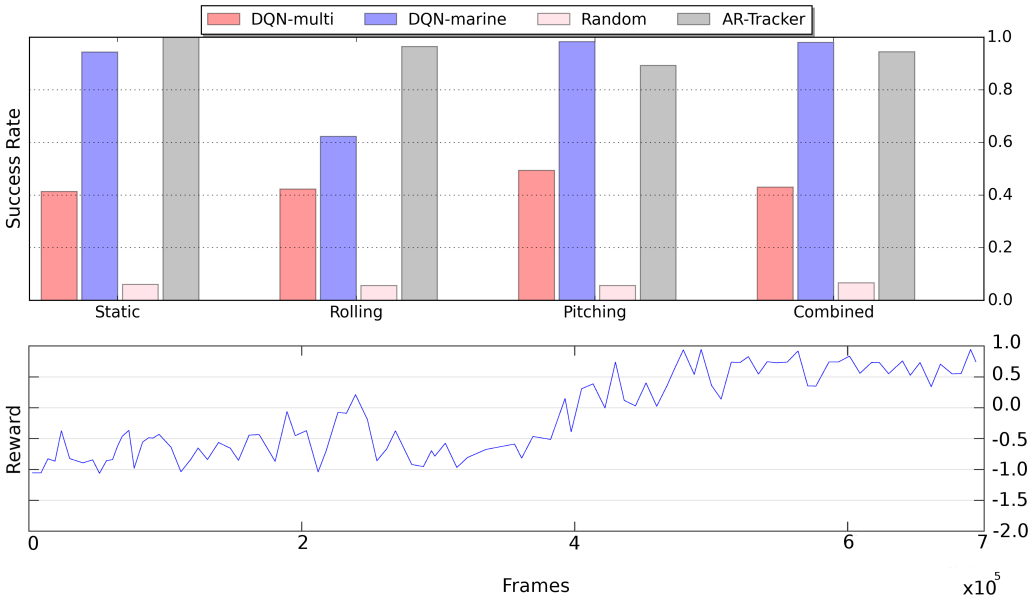


Figure 5.14: Results of the marker detection simulations with a floating marker. (Top) Comparison of the test phase for the vertical descent between the DQN trained on marine texture (DQN-marine), the DQN trained on multiple textures (DQN-multi), the AR-Tracker and a random agent. (Bottom) On the x-axis is represented the number of frames, whereas on the y-axis is represented the reward per episode.

must pointed out that this condition was very challenging because of the high variability in lighting and attitude instability due to altimeter's readings.

Floating marker The results of the tests performed with a floating marker, reported in Figure 5.14, showed that the agent was able to learn an efficient policy for maximizing the reward also in this scenario. The reward increased stably without any anomaly.

In contrast to the training phase, during which the marker was kept fixed in the same position directly on the surface of marine textures, the testing phase was performed positioning the marker on the deck of an unmanned surface vehicle whose motion is affected by different simulated sea conditions. Based on that, four test cases have been defined: (i) static, (ii) rolling, (iii) pitching and (iv) combined.

The results of the four tests are summarised in Figure 5.14 (top). The bar chart compares the performance of the trained agent, called DQN-marine, with a random agent, the AR-Tracker algorithm described in Chapter 4 and the DQN-multi described in the

Table 5.3: Landmark detection overall performances on the test with a floating marker. The performance is the average percentage across agents of correct landings. Standard deviation is in brackets. The best scores are in bold.

Agent type	Performance
DQN-marine	$0.88(0.15)$
DQN-multi	$0.44(0.03)$
AR-Tracker	**0.95 (0.03)**
Random	$0.06(0.01)$

previous section and not trained on marine textures.

The average score for the first test (static deck) was 94% for the DQN-marine, 41% for DQN-multi, 100% for the AR-Tracker and 6% for the random agent.

The second test (rolling deck) saw a general degradation in performance, with the DQN-marine be successful only in 62% of the cases, probably due to the fact the network has been trained on a fixed marker and it never experienced changes of attitude during the training. DQN-multi confirmed the performance of 42%, similarly to the previous test. Also the random agent still confirms the same performance (6%). The AR-Tracker proved to still be the agent with the best performance, obtaining a success rate of 96%.

In the third test with a pitching deck, the DQN-marine had success rate to 98%, as the DQN-multi with 49%. On the other hand, the AR-Tracker saw a drop in performance up to 89%. The random agent has the same success rate of the previous test cases, as expected.

The fourth test, combining rolling and pitching, better represents sea conditions. The results showed that the DQN-marine is the best agent with a performance of 98%, followed by the AR-Tracker with a performance of 94%, the DQN-multi with 43% and the random agent with 6%.

The overall performances are reported in Table 5.3. In general, AR-Tracker is the best agent in performing the detection phase of a floating marker with a success rate of 95%, followed by DQN-marine with 88%. This result is drastically influenced by the low performance encountered in the test with a rolling deck. The DQN-multi has the third

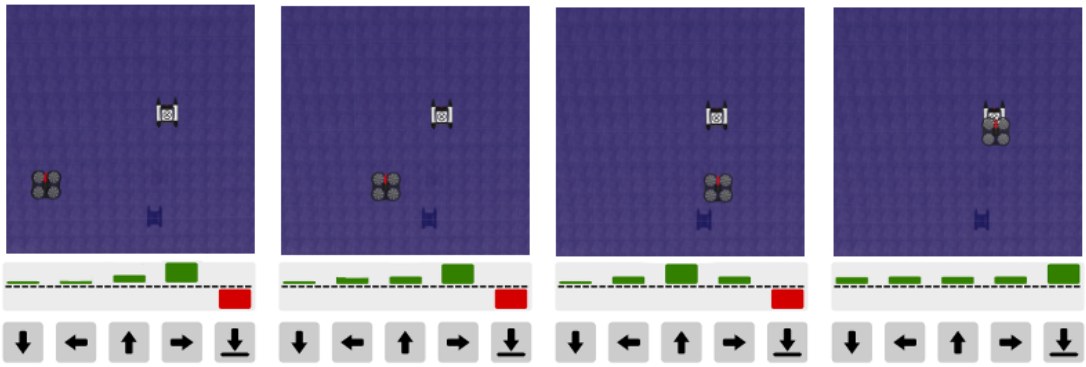


Figure 5.15: Snapshots representing the action distribution during the landmark detection phase. The bottom bar is the utility distribution of the actions. The trigger command has a negative utility (red bar) when the UAV is far from the marker.

place with an overall success rate of 44%. This result can be explained saying that the training set used in this work, representing sea surfaces with light reflection and sea foam, presents features that are different from those in the training set used for DQN-multi. As a result, the DQN-multi is not able to generalise well on the new dataset. Lastly, the random agent with the poorest performance of 6%.

It is possible to further analyse the DQN-marine policy observing the action-values distribution in different states (Figure 5.15). When the UAV is far from the marker, the DQN for landmark detection penalizes the landing action. However, when the UAV is over the marker this utility significantly increases triggering the network responsible for the vertical descent.

5.3.3 Vertical descent

In the second series of simulations, the DQN specialized in the vertical descent was trained and tested . To encourage the UAV to decrease its altitude, during the ε-greedy action selection, the action was sampled from a non-uniform distribution were the descend action had a probability of ρ and the other N actions a probability $\frac{1-\rho}{N}$. It was used the *exploring-start* to generate the UAV at different altitudes and to ensure a wider exploration of the state space.

Table 5.4: Statistics for the agents trained for the vertical descent.

Agent type	Synch. step	Positive Buffer Replay size	Negative Buffer Replay size	Neutral Buffer Replay size	Training steps
DQN-single	30000	5×10^5	5×10^5	10^6	10^6
DQN-multi	30000	5×10^5	5×10^5	10^6	10^6
DQN-marine	30000	5×10^5	5×10^5	1.2×10^6	10^6

Instead of the standard buffer replay, it was used a form of prioritized sampling with three separate buckets for storing neutral, negative and positive experiences, as described in Section 5.2.3. In this way, a fixed amount of positive and negative experiences was always inserted in the batch at training time.

I trained three networks, the first in a single texture condition (DQN-single), the second in multi-texture condition (DQN-multi) and the latter in a marine scenario (DQN-marine).

5.3.3.1 Description of the task

During the training phase, the *world* was represented by a uniform floor of size $100m \times 100m$ with a static marker positioned in the centre. The texture was randomly sampled every 50 episodes among the possible alternatives contained in the training set. The state-space in the vertical descent phase is significantly larger than in the marker detection and exploration is expensive. For this reason, the number of textures used for the training of DQN-multi was reduced, randomly sampling 20 textures from the 71. It can be hypothesised that using the entire training set can lead to better performance.

The UAV could use five actions: forward, backward, left, right and down. The action was repeated for 2 s leading to an approximate shift of 1 m because of the constant speed of 0.5 m/s. The descent action was performed at a lower speed of 0.25 m/s to reduce undesired vertical shifts and repeated for 4 s.

For the partitioned buffer replay it was chosen $\rho^+ = 0.25$, $\rho^- = 0.25$, and $\rho^\sim = 0.5$. In this way, a fixed value of 8 positive and negative experiences was always guaranteed in the batch of 32 experiences. A time limit of 80 s (40 steps) was used to stop the

episode and start a new one.

The UAV was generated with a random orientation inside a bounding box of size $3m \times 3m \times 20m$ m at the beginning of the episode (Figure 5.11). A positive reward of 1.0 was given only when the UAV entered in a target-zone of size $1.5m \times 1.5m \times 1.5m$, centred on the marker. A negative reward of -1.0 was given if the UAV descended above 1.5 m outside the target-zone. A cost of living of -0.01 was applied at each time step.

The same hyper-parameters described in Section 5.3.2.1 were used to train the agent. Details about the synchronisation step, buffer replay size and training time (expressed as the number of steps required before convergence) are reported in Table 5.4. In addition to the hardware mentioned in Section 5.3.2.1, it was also used a separate machine to collect preliminary experiences. This machine is a multi-core workstation with 32 GB of RAM and a GPU NVIDIA Tesla K-40. The training phase took around 3 days for the DQN-single and 5 days for the DQN-multi and DQN-marine agents. In all the cases, the physic engine running five times the real time speed.

5.3.3.2 Results

To test the performance of the agents, two groups of tests were designed, as for the marker detection simulations.

In the first group, the landing success rate of DQN-single, DQN-multi, human pilots, the AR-tracker of Chapter 4, and a random agent were measured in five tests. (i) In the first test, the agents performed landing on 21 unseen uniform textures. (ii) The second test consisted in landing on uniform textures with a corrupted marker (Figure 5.7-i). (iii) In the third test, 25 textures have been randomly sampled from the test set and mixed in a mosaic-like composition. (iv) In the fourth test, landing has been accomplished in three photo-realistic environments: warehouse, disaster site, powerplant (Figure 5.7-e/g). (v) In the fifth and latest test the UAV had to land in four real-world indoor environments: laboratory, small hall, large hall, mezzanine (Figure 5.7-a/d). The performance of human pilots has been measured in all the simulated environments through discrete and a continuous control using the same procedure described in Section 5.3.2.1. The hu-

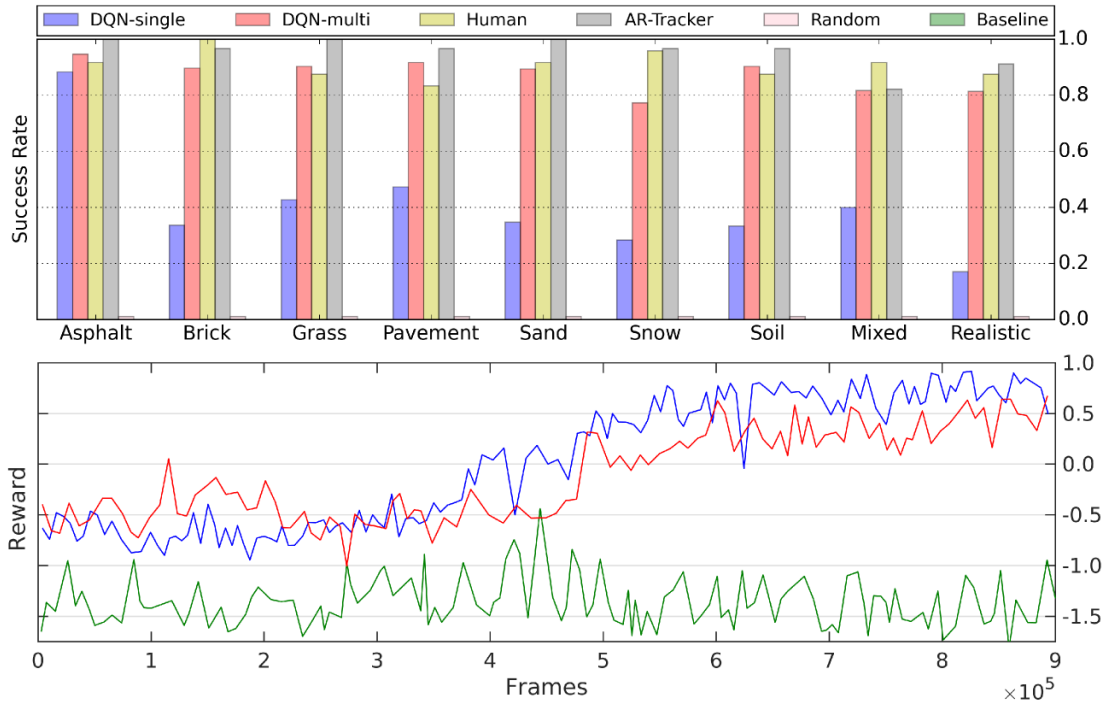


Figure 5.16: Results of the vertical descent simulations with a static marker. (Top) Descending success rate. (Bottom) Accumulated reward per episode for DQN-single (blue line), DQN-multi (red-line), and baseline (green-line).

man data has been obtained using a sample of 7 subjects. The subjects had to adjust the vertical position of the UAV in order to move toward the marker and obtain a positive reward.

In the second group of test, the landing success rate of DQN-marine, DQN-multi, the AR-Tracker, and a random agent are measured in four tests. (i) In the first test, the agents must land on the static deck of a USV. (ii) The second test consisted in landing on a rolling deck, while (iii) in the third test, the deck was subject to pitch. (iv) In the fourth and last test, the landing is accomplished on a deck subject to combined roll and pitch in order to simulate a more realistic condition.

Static marker The results achieved show that both DQN-single and DQN-multi were able to learn the task. The accumulated reward per episode showed in Figure 5.16 (bottom), increased stably in both DQN-single and DQN-multi. It is also reported the baseline curve of a network trained on both detection and descent which did not learn

to accomplish the task. The results of the test phase are summarized in Figure 5.16 (top). The bar chart compares the performances of the DQN-single, DQN-multi, human pilots, AR-tracker, and random agent. For human pilots it is only reported the score in the discrete control condition that is higher with respect to the space-navigator condition ($+4\%$).

The average score on the first test (uniform textures) is 89% for DQN-multi, 44% for DQN-single, 91% for humans, and 98% for the AR-tracker. Since both human pilots and DQNs used discrete steps to control the UAV, it is possible to estimate the average number of steps required to accomplish landing. For human pilots the average number of steps is 23, whereas for the DQN-multi is 19, meaning that human pilots were slower.

In the second test, the performances of the DQN-multi and AR-tracker were compared on uniform textures with a corrupted marker. The AR-tracker had a significant drop from 98% to 0% due to the failure of the underlying template-matching algorithm. The DQN-multi had a limited drop from 89% to 51% showing to be more robust to marker corruption.

The third test, in which 25 textures randomly sampled from the testing set were mixed, showed a general drop (DQN-multi= 82%, DQN-single=40%, Human=92%, Random=1%, AR-tracker=82%).

The fourth test was performed in simulated photo-realistic environments. It has been observed a drop similar to the previous test (DQN-multi= 81%, DQN-single=17%, Human=88%, Random=1%, AR-tracker=91%).

The overall performances on uniform textures, mixed textures and realistic environments are reported in Table 5.5 and are 87% for DQN-multi, 41% for DQN-single, 91% for human pilots, and 96% for the AR-tracker.

In the fifth and last test (real-world) the DQN-multi has been used to control the descending phase in 40 flights equally distributed in four environments (laboratory, small hall, large hall, mezzanine). The system obtained an overall success rate of 62%. Most

Table 5.5: Descending manoeuvre overall performances on uniform textures, mixed textures and realistic environment using a static marker. The performance is the average percentage across agents of correct landings. Standard deviation is in brackets.. The best scores are in bold.

Agent type	Performance
DQN-single	$0.41(0.18)$
DQN-multi	$0.87(0.05)$
AR-Tracker	**0.96 (0.05)**
Random	$0.01(0.0)$
Humans	$0.91(0.04)$

Figure 5.17: Snapshots representing vertical descent in the large hall environment. The bottom bar is the utility distribution of the actions. Descent has a negative utility (red bar) when the UAV is not centred on the marker.

of the missed landings have been caused by extreme light conditions (e.g. mutable natural light), and by flight instability (e.g. strong drift) caused by the altimeter's readings.

It is possible to further analyse the DQN-multi policy observing the action-values distribution in different states (Figure 5.17). When the UAV is far from the marker the DQN penalizes the descent. However, when the UAV is over the marker this utility significantly increases overcoming the others.

Floating marker As for the marker detection phase, the results show that the DQN-marine agent was able to learn an efficient policy for maximizing the reward. In Figure 5.18 the chart reward per episode is illustrated. The results of the test phase are summarized in Figure 5.18 (Top). The bar chart compares the performances of the DQN-marine, DQN-multi, AR-Tracker of Chapter 4 and a random agent.

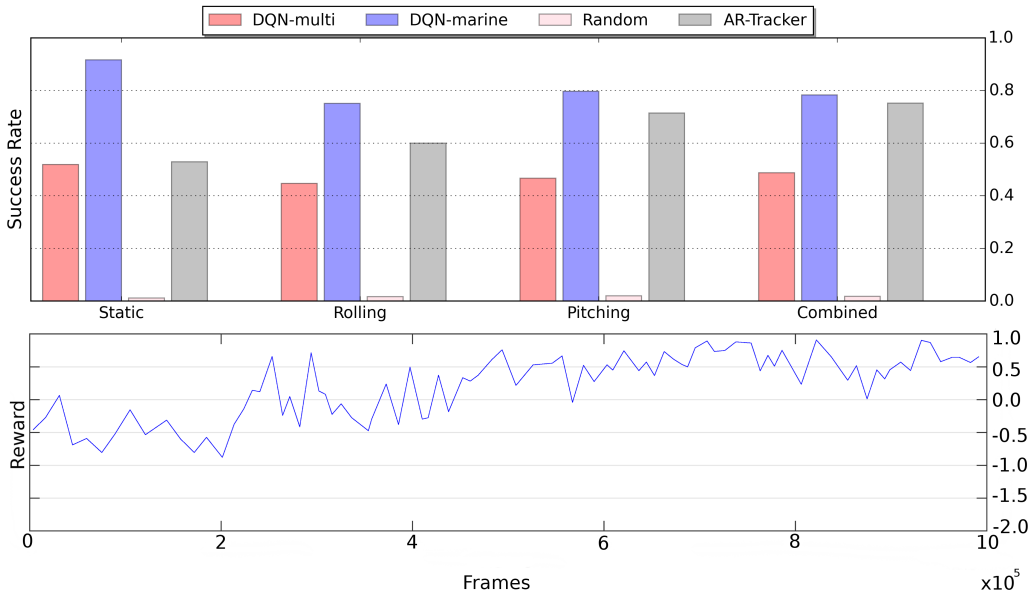


Figure 5.18: Results of the vertical descent simulations with a floating marker. (Top) Comparison of the test phase for the vertical descent between the DQN trained on marine texture (DQN-marine), the DQN trained on multiple textures (DQN-multi), the AR-Tracker and a random agent. (Bottom) On the x-axis is represented the number of frames, whereas on the y-axis is represented the reward per episode (c)

The average score for the first test (static deck) is 81% for the DQN-marine, 51% for DQN-multi, 53% for the AR-Tracker and 0% for the random agent.

The second test (rolling deck) saw a drop in performance of the DQN-marine to 75%, probably due to the fact the network has been trained on a fixed marker and it never experienced changes of attitude during the training. For the same reason, DQN-multi obtained a lower score of 45% compared to the previous test. While the random agent still confirms the same null performance, the AR-Tracker proved to be more successful than in the test with a static marker, obtaining now a performance score of 60%.

Similar performance have been registered also in the third test with a pitching deck (DQN-marine = 80%, DQN-multi = 47%, AR-Tracker = 71% and random agent = 0%).

The fourth test is the one probably most interesting, because better represents situations faced at sea. The results showed that the DQN-marine is the best agent with a performance of 79%, followed by the AR-Tracker with 75%, the DQN-multi with 49% and

Table 5.6: Descending manoeuvre overall performances on the tests with a floating marker. The performance is the average percentage across agents of correct landings. Standard deviation is in brackets. The best scores are in bold.

Agent type	Performance
DQN-marine	**0.81 (0.06)**
DQN-multi	$0.48(0.03)$
AR-Tracker	$0.64(0.09)$
Random	$0.02(0.01)$

the random agent with null performance.

The overall performance for all the four test cases is reported in Table 5.6. In general, the DQN-marine is the best agent in the descending manoeuvre with a success rate of 81%, while the AR-Tracker of Chapter 4 has only 64% of success across all the tests performed.

5.4 Summary

This chapter presents a deep reinforcement learning based solution for achieving autonomous landing capability for UAVs. Using the images acquired from the UAV bottom camera, a hierarchy of Deep Q-Networks was able to learn the high level navigation commands required for directing the UAV above the marker and then land on it. Given the complexity of the task (mainly related to a huge state-space to explore), a divide-and-conquer strategy was adopted to split the problem in marker detection and descending manoeuvre, both addressed by an independent network that is able to call the other one through a trigger action. In the first task, the UAV is kept at a fixed altitude of 20m and it can only translate on a plane in order to centre the marker. Once centred, the first DQN triggers the second one and the descending phase begins. During such phase, the quad-rotor decreases its altitude while keeping the marker centred through lateral shifts. In the end, once the UAV reaches an altitude of 1.5 m from the ground, the touchdown can be accomplished through an additional network or a closed loop controller.

In order to stabilise the learning, the Double-DQN architecture was adopted for the descending manoeuvre because not suffering from the overestimation of the Q-function

like the classic DQN. In order to compensate the reward sparsity, it was introduced a new form of buffer replay, called prioritized buffer replay. In this way, a certain amount of important experience is always guaranteed in the training mini-batch.

All the training has been performed inside the QLaB simulator while adopting the domain randomization technique. While continuously changing the environmental setting, the network became more robust to identify salient features in the images acquired by the camera. Using this approach, the network not only was able to learn how to generalise well on a wide set of simulated environments of incremental complexity, but it was also possible to fill the reality gap.

Simulated experiments proved that the algorithm is faster than human agents in achieving the task, while more robust to marker corruption than the algorithm presented in Chapter 4. The most remarkable outcome is that the networks were able to generalize to real environments despite the training was performed on a limited subset of textures. In all the missed landing the flight was interrupted because of the expiration time. Not even once the UAV landed outside of the pad. Most of the missed landing have been caused by extreme conditions (mutable lighting and strong drift), not modelled in the simulator.

Chapter 6

Conclusions and Future Work

THIS chapter provides an overview of the findings and topics covered in this thesis. In particular, Section 6.1 provides an overview of the problem addressed in the thesis and the solution proposed. Here, the main contribution of this work, namely the adoption of deep neural networks for learning a control policy for UAV autonomous landing, is discussed again together with all its advantages compared to human agents and existing solutions. Section 6.2 is a summary of the main goals of the thesis. Finally, Section 6.3 contains new directions which aim to expand the current research.

6.1 Overview

Despite the effort of the research community, autonomous navigation is still an open problem for Unmanned Surface Vehicles (USVs). This is true specifically while navigating at sea, where water and wind currents can pose serious challenges to our vehicle. With this in mind, the main motivation for the work developed and presented in this thesis was the need for a path planner for a USV, which is robust enough to changes in trajectories of obstacles moving in proximity to the own vessel. In order to do so, the planner can integrate information coming from an unmanned aerial vehicle (UAV) flying above the USV in the surrounding area. However, it is important to highlight that this is not a single-man task and it was never intended to be solved in one study; therefore, the results obtained make a significant contribution to the bigger goal yet to come.

The reason behind the idea of building a multi-agent system combining a UAV and a USV is mainly due to the possibility to better map the environment surrounding the vessel in a representation that will be used for the planning of a safe collision-free path.

In this way, the USV's sensors limitations described in Chapter 2 can be overcome and a real-time updated map of the environment is almost always available. In fact, detecting and identifying static and moving obstacles in a reasonably short amount of time is a major limitation of the existing literature concerning path planning methods for a USV. This is mainly due to sensor limitation, such as a short detection range or low resolution. A UAV can, therefore, be used as an additional flying eye to sense and map a larger environmental area. The final coupled system can be deployed for multiple applications, from data collection to water and coastal monitoring, to name just a few.

While realising the heterogeneous multi-agent system proposed, landing a UAV on the deck of a USV is definitely the most important whilst challenging task. This is because waves and wind effects can put at risk the success of the landing manoeuvre. The existing literature does not present a valuable number of works trying to solve this problem, hence the focus of this thesis.

Most of the existing work is based on integrating data coming from a USV with those from a UAV's onboard sensor suite. This thesis takes distance from this approach while proposing two solutions requiring a minimal or null share of information between the two vehicles involved. This is to reduce any possible delay in the communication that would make a landing unfeasible in real-time. Moreover, the core contribution of this thesis is represented by a solution based on the novel concept of Deep Reinforcement Learning (DRL), namely the combination of Deep Learning and classic Reinforcement Learning. This work follows the big breakthrough after the success obtained by DeepMind's Deep Q-Network (DQN), combining Convolutional Neural Networks (CNNs) with Q-learning. Many efforts have been undertaken by the research community to improve the stability of DQN, to propose better algorithms and to solve many robotic control problems such as mobile navigation and grasping. However, owing to the complexity of UAV navigation, given the six degrees of freedom of the platform and the dimension of the state space, few studies have been published while applying DRL to any task involving a UAV. The only example already existing at the beginning of the study here

presented is given by Zhang et al. (2016), in which model-predictive-control (MPC) is used at training time to generate navigation data. This data are then used to train a deep neural network that takes in only raw observation input from the UAV's sensors. As a result of the training process, the deep policy found is able to control the vehicle without knowing the full state of the vehicle and a much lower computational cost of MPC. In contrast, the solution presented in this thesis is the first attempt to use model-free control (using the DQN architecture) to navigate a UAV towards a destination, in this case, a landing pad. Differing from Zhang et al. (2016), an end-to-end solution is proposed in which a CNN learns directly how to map raw sensory input, in particular, low-resolution grey scale images, to high-level control actions, without any human supervision or intervention. Various improvements have been introduced to the original DQN, such as the idea of hierarchical deep reinforcement learning, adopting a divide-and-conquer strategy and splitting a complex task in multiple simpler ones (the overall landing procedure is reduced to marker detection and landing manoeuvre), and a new form of experience buffer replay to stabilise and improve the convergence of the learning. Moreover, thanks to a technique noted as domain randomization, the algorithm learnt to generalise well not only on a large variety of simulated environments but also in the real world, with numerous random variables not modelled within the training environments (e.g., light wind currents, variable light conditions, cluttered environments). In order to better evaluate this work, the DRL solution has been extensively tested against other agents, starting from the algorithm presented in Chapter 4. This has been implemented using well-known techniques in the robotic community, such as the extended Kalman filter for pose estimation and linear control. This approach was developed with the intent to identify the main problems affecting state-of-the-art solutions for autonomous landings and, for this purpose, it has been used as baselines in all the test performed. The other agents used in the comparison are represented by a random policy and an average human. The random policy simply selects at each time step a new action, sampling it from a uniform distribution across all the possible action. The average human agent is instead modelled using a small group of people

selected among other PhD students from the School of Computer Science and School of Psychology. They are people of both sexes and of age comprised between 23 and 35 years old, without any prior direct experience in piloting drones but eventually with some confidence in playing console video games. An extensive testing phase of landing in a controlled indoor scenario proved that the developed solution based on DRL can achieve performances which are comparable with the average human pilot and also the developed classic solution. More importantly from the thesis purposes, the DRL solution achieve similar performance than the classic one in marker detection at sea, while largely being more effective at landing on the USV.

The results contained in this thesis allow drawing important conclusions for what regards autonomous landing, in particular on the deck of a USV. First of all, it has been proved that convolutional neural networks are a valid alternative to classic computer vision techniques (such as template matching) when trying to identify and track a visual feature (e.g., the landing pad) across consecutive frames. Moreover, they represent a much better solution when dealing with sensor noise, being able to generalise from a noise-free training scenario, such as the QLaB simulator, to the real world. In addition to this, it has been demonstrated that it is possible to train a model-free learning algorithm which is able to successfully control a complex system such as a UAV. This means that reinforcement learning research is now mature enough to go beyond the limit posed by simple toy examples and 2D video games, and be used to train agent that moves and interact with the real world.

6.2 Summary of the contributions to knowledge

This section revises the contribution already presented in the introductory chapter of the current thesis (Chapter 1). The key goal of the work presented was the advancement of robotic control solutions for autonomous landing. Here, it is described how this high-level goal has been achieved in multiple stages:

1. Developing a vision-based method that only relies on onboard sensors mounted on a UAV and minimum communication between vehicles or a control station.

2. Developing a learning approach based on a hierarchy of convolutional neural network for feature learning that is more flexible and robust than traditional control solution based on classic computer vision techniques.

3. Extending the work in simulation to a real-world implementation with a satisfactory level of success.

At first, the thesis explored a new approach for autonomously landing a UAV on a visual marker while combining visual information coming from the vehicle's camera and estimates on the current 6-DOF pose of the landing platform. The idea was to develop such algorithm while using state-of-the-art solutions in classic robotics, in order to better identify the weakness of the existing literature and use it as a baseline for what represents the core contribution of the thesis itself. The effort made resulted in a simulated environment (ROS/Gazebo based) written in C++ to perform simulated studies and presented in Chapter 4. This contribution was published as a conference paper at ECMR 2017 (Polvara et al. 2017) and in *Drones* (Polvara, Sharma, Wan, Manning & Sutton 2018).

The second step was to propose a solution based on neural networks that was able to overcome the limitations of classical control theory and computer vision. For this reason, a robust and self-improving solution based on the novelty of deep reinforcement learning has been discussed and presented in Chapter 5. This solution achieved performance superior to human agents and comparable with state-of-the-art approaches based on template matching. Moreover, it also proved to be much more robust to marker corruption (meaning to image corruption) than the existing methods. As a result of this work, an open source simulator called *QLaB* has been developed and released to the community (`https://github.com/pulver22/QLAB`) for improving research on autonomous landing using reinforcement learning. This work has been presented at ICUAS 2018 (Polvara, Patacchiola, Sharma, Wan, Manning, Sutton & Cangelosi 2018) and recently published on *Robotica* (Polvara et al. 2019).

Finally, the previous work was integrated on a real platform, the Parrot Bebop 2, and

tested in various indoor environments after all the training phase has been performed in simulation. Thanks to a technique noted as domain randomization (Tobin et al. 2017), consisting of frequently randomising the environment in which a virtual agent is trained for accomplishing a task, the generalization capability of the DQN was largely improved in such a way the network not only was able to land on a wide set of simulated environments of incremental complexity, but it was also possible to fill the reality gap. Additional simulated and real experiments are part of a work still under revision.

6.3 Suggestions for future work

The long term goal of this thesis was to give a contribution to the field of marine robotics, in particular towards deploying a UAV in a marine scenario and make it land autonomously on the deck of a USV. The previous section provides an explanation of how this goal has been achieved. Now, possible future research directions are suggested and discussed.

6.3.1 Moving towards continuous control while combining model-based and model-free solutions

A particularly interesting direction of research concerns the improvement of the algorithm presented in Chapter 5, proposing a continuous control solution based on more recent model-free deep reinforcement solution (e.g., Deep Deterministic Policy Gradient (Mnih et al. 2016), Trust Policy Region Optimization (Schulman et al. 2015), Proximal Policy Optimization (Schulman et al. 2017)). Applying the DQN architecture to the landing problem required lengthly heuristic trials and experience in order to discretise both the state- and the action-space. Even though real experiments proved this solution is feasible, having a vehicle moving in a discrete manner in the real world is not acceptable (also for safety reason). Therefore, a continuous control approach that generates smooth trajectories is desirable.

A second point in which the actual work can be improved is linked with the model-free nature of the DQN and the algorithms listed just before. A limitation of all model-

free solutions is the sampling efficiency, meaning the agent needs to perform a large amount of interaction with the surrounding environment in order to achieve a stable control policy. Therefore, it would be interesting to study how model-based and model-free methods can be combined in order to speed up the learning process. A first attempt has been achieved by Bansal et al. (2017) in which the cost estimated of the model-based component is used as prior by a Bayesian Optimisation model-free policy search to guide the policy exploration. In this way, it is possible to exploit the dynamics model learned throughout the entire state-action space.

6.3.2 Distilling hierarchical knowledge into a single model

In Chapter 5 it has been explained that the DRL solution proposed consists of a hierarchy of DQN after having adopted a divide-and-conquer strategy for simplifying and addressing the problem in an easier manner. On the other hand, this would require two or more relatively large neural networks to be deployed at operation time, with limitations posed by restricted computation power and memory space. In this sense, it would be interesting to explore the possibility of using a technique called "distillation". This idea, initially proposed by Buciluǎ et al. (2006), suggests that is possible to transfer knowledge from a complex model to a much smaller one. Hinton et al. (2015) extended this work obtaining interesting results on the MNIST dataset for digit classification and on the Android Voice Assistant while distilling the knowledge in an ensemble of models into a single model. In the simplest classification scenario, the knowledge transfer is realised by training the distilled model on a transfer set and using a soft target distribution produced by the original large model. In order to produce softer probability distributions over classes, a high temperature value is used in the softmax function. Chapter 5 has already shown that it was not possible for a single network to learn how to perform marker detection and vertical descent in a combined way. Therefore, a possible direction for future research is to investigate if it possible to use distillation to compress all the hierarchy of DQNs into a single model that is able to successfully achieve autonomously landing. In order to do so, the logits of the last fully

connected layer of each of the networks belonging to the hierarchy would be used as a soft target for the distilled model. In this way, the distilled model should theoretically be able to acquire the knowledge of the first network in the hierarchy for the marker detection phase and, at the same time, the knowledge of the second network for the descending manoeuvre.

Glossary

2D Two Dimensions

3D Three Dimensions

ACO Ant Colony Optimisation

ALM Adaptive Learning Module

ANN Artificial Neural Network

APF Artificial Potential Field

AR Augmented Reality

BB Bounding Box

CNN Convolutional Neural Network

COLREGS International Regulations for Preventing Collisions at Sea

CPA Closest Point of Approach

DI Danger Index

DLT Direct Linear Transform

DNN Deep Neural Network

DOF Degrees of Freedom

DQN Deep Q-Network

DRL Deep Reinforcement Learning

EA Evolutionary Algorithm

EKF Extended Kalman Filter

FL Fuzzy Logic

FOV Field of View

GPP Global Path Planner

GPS Global Positioning System

GPU Graphical Processing Unit

GR Guarding Ring

HD High Definition

HOG Histogram of Orient Gradient

IMU Inertial Measurement Unit

IR Infra-red

LIDAR Laser Direction and Ranging

LOAM Local Obstacle Avoidace Module

LPP Local Path Planner

MAS Multi-Agent System

MBZIRC Mohamed Bin Zayed International Robotics Challenge

MDP Markov Decision Process

ML Machine Learning

MPC Model Predictive Control

OAABHW Obstacle Avoidance Algorithm Based on a Heading Window

PID Proportional-Integral-Derivative

PP Path Planner

QLab Quadrotor Landing Benchmark

RANSAC Random Sample Consensus

RL Reinforcement Learning

ROS Robot Operating System

SARSA State-Action-Reward-State-Action

SDK Software Development Kit

STD Standard Deviation

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

USV Unmanned Surface Vehicle

VG Visibility Graph

ViBe Visual Background extractor

VO Velocity Obstacle

VTOL Vertical Takeoff and Landing

List of references

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. et al. (2016), 'Tensorflow: Large-scale machine learning on heterogeneous distributed systems', *arXiv preprint arXiv:1603.04467* .

Abawi, D. F., Bienwald, J. & Dorner, R. (2004), Accuracy in optical tracking with fiducial markers: An accuracy function for artoolkit, *in* 'Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 260–261.

Achanta, R., Hemami, S., Estrada, F. & Susstrunk, S. (2009), 'Frequency-tuned Salient Region Detection', *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009* pp. 1597–1604.

Allen, C. H. (2012), 'The Seabots are Coming Here: Should they be Treated as 'Vessels' ?', *The Journal of Navigation* **65**, 749–752.

Almeida, C., Franco, T., Ferreira, H., Martins, A., Santos, R., Almeida, J. M. & Silva, E. (2009), 'Radar Based Collision Detection Developments on USV ROAZ II', *Oceans 09 - Bremen* pp. 1–6.

Alves, J., Oliveira, P., Oliveira, R., Pascoal, A., Rufino, M., Sebastiao, L. & Silvestre, C. (2006), 'Vehicle and mission control of the delfim autonomous surface craft', *14th Mediterranean Conference on Control and Automation, 2006. MED '06*. pp. 1–6.

Azuma, R. T. (1997), 'A survey of augmented reality', *Presence: Teleoperators and virtual environments* **6**(4), 355–385.

Azzabi, T., Amor, S. B. & Nejim, S. (2014), 'Obstacle Detection for Unmanned Surface Vehicle', *International Conference on Electrical Sciences and Technologies in Maghreb 2013* pp. 1–7.

Bagnell, J. A. & Schneider, J. G. (2001), Autonomous helicopter control using reinforcement learning policy search methods, *in* 'Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on', Vol. 2, IEEE, pp. 1615–1620.

Bandyophadyay, T., Sarcione, L. & Hover, F. S. (2010), A simple reactive obstacle avoidance algorithm and its application in singapore harbor, *in* 'Field and Service Robotics', Springer, pp. 455–465.

Bansal, S., Calandra, R., Levine, S. & Tomlin, C. (2017), 'Mbmf: Model-based priors for model-free reinforcement learning', *arXiv preprint arXiv:1709.03153* .

Barnich, O. & Van Droogenbroeck, M. (2011), 'ViBe: A universal background subtraction algorithm for video sequences', *IEEE Transactions on Image Processing* **20**(6), 1709–1724.
URL: *http://www.telecom.ulg.ac.be/research/vibe*

Barto, A. G. & Mahadevan, S. (2003), 'Recent advances in hierarchical reinforcement learning', *Discrete Event Dynamic Systems* **13**(4), 341–379.

Belcher, P. (2002), 'A sociological interpretation of the colregs', *The Journal of Navigation* **55**, 213–224.

Bellemare, M., Naddaf, Y., Veness, J. & Bowling, M. (2015), The arcade learning environment: An evaluation platform for general agents, *in* 'Twenty-Fourth International Joint Conference on Artificial Intelligence'.

Bertram, V. (2008), 'Unmanned Surface Vehicles : A Survey', *Skibsteknisk Selskab, Copenhagen, Denmark* pp. 1–14.

Bibuli, M., Bruzzone, G., Caccia, M. & Lapierre, L. (2009), 'Path-following algorithms and experiments for an unmanned surface vehicle', *Journal of Field Robotics* **26**(8), 669–688.
URL: *http://dx.doi.org/10.1002/rob.20303*

Blaich, M., Koehler, S., Schuster, M., Reuter, J. & Tietz, T. (2015), 'Mission Integrated Collision Avoidance for USVs using Laser Ranger', *Oceans 2015 - Genova* pp. 0–5.

Borghgraef, A., Barnich, O., Lapierre, F., Van Droogenbroeck, M., Philips, W. & Acheroy, M. (2010), 'An evaluation of pixel-based methods for the detection of floating objects on the sea surface', *EURASIP Journal on Advances in Signal Processing* **2010**(1), 1–11.
URL: *http://dx.doi.org/10.1155/2010/978451*

Bouguet, J. (2000), 'Pyramidal implementation of the lucas kanade feature tracker', *Intel Corporation, Microprocessor Research Labs* .

Bourgault, F., GÃűktogan, A., Furukawa, T. & Durrant-Whyte, H. F. (2004), 'Coordinated search for a lost target in a bayesian world', *Advanced Robotics* **18**(10), 979–1000.
URL: *http://dx.doi.org/10.1163/1568553042674707*

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. & Zaremba, W. (2016), 'Openai gym', *arXiv preprint arXiv:1606.01540* .

Buciluǎ, C., Caruana, R. & Niculescu-Mizil, A. (2006), Model compression, *in* 'Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 535–541.

Campbell, S., Naeem, W. & Irwin, G. W. (2012), 'A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres', *Annual Reviews in Control* **36**(2), 267–283.

Casalino, G., Turetta, A. & Simetti, E. (2009), 'A Three-layered Architecture for Real Time Path Planning and Obstacle Avoidance for Surveillance USVs Operating in Harbour Fields', *Oceans 2009 - Europe* pp. 1–8.

Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P. & Longhi, S. (2009), A vision-based guidance system for uav navigation and safe landing using natural landmarks, *in*

‘Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009’, Springer, pp. 233–257.

Chang, K.-Y., Jan, G. E. & Parberry, I. (2003), ‘A Method for Searching Optimal Routes with Collision Avoidance on Raster Charts’, *The Journal of Navigation* **56**(3), 371–384.
URL: *http://www.journals.cambridge.org/abstract_S0373463303002418*

Chen, C., Seff, A., Kornhauser, A. & Xiao, J. (2015), Deepdriving: Learning affordance for direct perception in autonomous driving, *in* ‘Proceedings of the IEEE International Conference on Computer Vision’, pp. 2722–2730.

Chen, J., Pan, W., Guo, T., Huang, C. & Wu, H. (2013), ‘An Obstacle Avoidance Algorithm Designed for USV Based on Single Beam Sonar and Fuzzy Control’, *IEEE International Conference on Robotics and Biomimetics (ROBIO), 2013* pp. 2446–2451.

Chenavier, F. & Crowley, J. L. (1992), Position estimation for a mobile robot using vision and odometry, *in* ‘Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on’, IEEE, pp. 2588–2593.

Colorado, J., Perez, M., Mondragon, I., Mendez, D., Parra, C., Devia, C., Martinez-Moritz, J. & Neira, L. (2017), ‘An integrated aerial system for landmine detection: Sdr-based ground penetrating radar onboard an autonomous drone’, *Advanced Robotics* **31**(15), 791–808.
URL: *http://dx.doi.org/10.1080/01691864.2017.1351393*

Commandant, U. (1999), ‘International regulations for prevention of collisions at sea, 1972 (72 colregs)’, *US Department of Transportation, US Coast Guard, COMMANDANT INSTRUCTION M* **16672**.

Cui, Y., Matsubara, T. & Sugimoto, K. (2017), ‘Kernel dynamic policy programming: Applicable reinforcement learning to robot systems with high dimensional states’, *Neural Networks* .

Darwin, C., Glick, T. F. & Kohn, D. (1996), *On evolution: The development of the theory of natural selection*, Hackett Publishing.

Dias, J., Althoefer, K. & Lima, P. U. (2016), 'Robot competitions: What did we learn?[competitions]', *IEEE Robotics & Automation Magazine* **23**(1), 16–18.

Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs.', *Numerische Mathematik* **1**, 269–271.

Dryanovsk, I., Morris, B. & Duonteil, G. (2010), 'ar_pose', http://wiki.ros.org/ar_pose.

Engel, J., Sturm, J. & Cremers, D. (2012*a*), 'Accurate figure flying with a quadrocopter using onboard visual and inertial sensing', *Imu* **320**, 240.

Engel, J., Sturm, J. & Cremers, D. (2012*b*), Camera-based navigation of a low-cost quadrocopter, *in* '2012 IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 2815–2821.

Engel, J., Sturm, J. & Cremers, D. (2014), 'Scale-aware navigation of a low-cost quadrocopter with a monocular camera', *Robotics and Autonomous Systems* **62**(11), 1646–1656.

Ettinger, S. M., Nechyba, M. C., Ifju, P. G. & Waszak, M. (2003), 'Vision-guided flight stability and control for micro air vehicles', *Advanced Robotics* **17**(7), 617–640.
URL: *http://dx.doi.org/10.1163/156855303769156983*

Ettinger, S., Nechyba, M., Ifju, P. & M (2003), 'Towards Flight Autonomy: Vision-based Horizon Detection for Micro Air Vehicles', *Proceedings of the Florida Conference in Recent Advances in Robotics* **7**(17), 617–640.
URL: *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.6951&rep=rep1&type*

Fahimi, F. (2008), *Autonomous robots: modeling, path planning, and control*, Vol. 107, Springer Science & Business Media.

Falanga, D., Zanchettin, A., Simovic, A., Delmerico, J. & Scaramuzza, D. (2017), Vision-based autonomous quadrotor landing on a moving platform, *in* 'IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)', IEEE.

Fayek, H. M., Lech, M. & Cavedon, L. (2017), 'Evaluating deep learning architectures for speech emotion recognition', *Neural Networks* .

Fischler, M. a. & Bolles, R. C. (1981), 'Random Sample Consensus: A Paradigm for Model Fitting with Applicatlons to Image Analysis and Automated Cartography', *Communications of the ACM* **24**(6), 381 – 395.

Forster, C., Faessler, M., Fontana, F., Werlberger, M. & Scaramuzza, D. (2015), Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles, *in* 'Robotics and Automation (ICRA), 2015 IEEE International Conference on', IEEE, pp. 111–118.

Gal, O. & Zeitouni, E. (2013), Tracking objects using phd filter for usv autonomous capabilities, *in* 'Robotic sailing 2012', Springer, pp. 3–12.

Glorot, X. & Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks., *in* 'Aistats', Vol. 9, pp. 249–256.

Glorot, X., Bordes, A. & Bengio, Y. (2011), Deep sparse rectifier neural networks., *in* 'Aistats', Vol. 15, p. 275.

Goldstein, H. (1980), *Classical mechanics*, World student series, Addison-Wesley, Reading (Mass.), Menlo Park (Calif.), Amsterdam.
URL: *http://opac.inria.fr/record=b1078625*

Gui, Y., Guo, P., Zhang, H., Lei, Z., Zhou, X., Du, J. & Yu, Q. (2013), 'Airborne vision-based navigation method for uav accuracy landing using infrared lamps', *Journal of Intelligent & Robotic Systems* **72**(2), 197.

Halterman, R. & Bruch, M. (2010), Velodyne hdl-64e lidar for unmanned surface vehicle

obstacle detection, *in* 'SPIE Defense, Security, and Sensing', International Society for Optics and Photonics, pp. 76920D–76920D.

Handbook, C. (2003), 'Cargo loss prevention information from german marine insurers', *GDV, Berlin* .

Harris, C. & Stephens, M. (1988), 'A Combined Corner and Edge Detector', *Procedings of the Alvey Vision Conference 1988* pp. 147–151.

Hart, P., Nilsson, N. & Raphael, B. (1968), 'A formal basis for the heuristic determination of minimum cost paths', *IEEE Transactions on Systems Science and Cybernetics* **4**(2), 100–107.

Hartley, R. I. & Zisserman, A. (2004), *Multiple View Geometry in Computer Vision*, second edn, Cambridge University Press, ISBN: 0521540518.

Hasegawa, K. (2009), 'Advanced marine traffic automation and management system for congested waterways and coastal areas.pdf'.

Hasegawa, K. & Kouzuki, A. (1987), 'Automatic collision avoidance system for ships using fuzzy control', *Journal of the Kansai Society of Naval Architects* .

Hebb, D. O. (1949), *The organization of behavior: A neuropsychological theory*, Psychology Press.

Herissé, B., Hamel, T., Mahony, R. & Russotto, F.-X. (2012), 'Landing a vtol unmanned aerial vehicle on a moving platform using optical flow', *IEEE Transactions on Robotics* **28**(1), 77–89.

Hinton, G., Vinyals, O. & Dean, J. (2015), 'Distilling the knowledge in a neural network', *arXiv preprint arXiv:1503.02531* .

Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press.

Hough, P. (1962), 'Method and Means for Recognizing Complex Patterns', U.S. Patent 3.069.654.

Hwang, C.-N. (2002), 'The integrated design of fuzzy collision-avoidance and h [infty infinity]-autopilots on ships', *Journal of Navigation* **55**(01), 117–136.

Ito, M., Zhnng, F. & Yoshida, N. (1999), Collision avoidance control of ship with genetic algorithm, *in* 'Control Applications, 1999. Proceedings of the 1999 IEEE International Conference on', Vol. 2, IEEE, pp. 1791–1796.

Jetto, L., Longhi, S. & Venturini, G. (1999), 'Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots', *IEEE Transactions on Robotics and Automation* **15**(2), 219–229.

Jodoin, P. M., Konrad, J. & Saligrama, V. (2008), Modeling background activity for behavior subtraction, *in* 'Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on', pp. 1–10.

Kalman, R. E. (1960), 'A new approach to linear filtering and prediction problems', *Journal of basic Engineering* **82**(1), 35–45.

Kao, S.-L., Lee, K.-T., Chang, K.-Y. & Ko, M.-D. (2007), 'A fuzzy logic method for collision avoidance in vessel traffic service', *Journal of Navigation* **60**(01), 17–31.

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

Kemp, J. (2002), 'Collision regulations - discussion', *The Journal of Navigation* **55**, 145–155.

Kemp, J., Bechley, M., Cockcroft, N., Jurdzinski, M. & Thirslund, S. (2012), 'Collision Avoidance at Sea in the Mid-20th Century', *The Journal of Navigation* **65**, 191–205.

Khatib, O. (1985), ‘Real-time Obstacle Avoidance for Manipulators and Mobile Robots’, *IEEE International Conference on Robotics and Automation, (1985)* **2**, 500–505.

Kim, J., Jung, Y., Lee, D. & Shim, D. H. (2014), Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera, *in* ‘Unmanned Aircraft Systems (ICUAS), 2014 International Conference on’, IEEE, pp. 1243–1252.

Kim, S. J., Jeong, Y., Park, S., Ryu, K. & Oh, G. (2018), A survey of drone use for entertainment and avr (augmented and virtual reality), *in* ‘Augmented Reality and Virtual Reality’, Springer, pp. 339–352.

Kober, J., Bagnell, J. A. & Peters, J. (2013), ‘Reinforcement learning in robotics: A survey’, *The International Journal of Robotics Research* **32**(11), 1238–1274.

Koenig, N. & Howard, A. (2004), Design and use paradigms for gazebo, an open-source multi-robot simulator, *in* ‘Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on’, Vol. 3, IEEE, pp. 2149–2154.

Kruijff, G., Kruijff-KorbayovÃą, I., Keshavdas, S., Larochelle, B., JanÃ■cek, M., Colas, F., Liu, M., Pomerleau, F., Siegwart, R., Neerincx, M., Looije, R., Smets, N., Mioch, T., van Diggelen, J., Pirri, F., Gianni, M., Ferri, F., Menna, M., Worst, R., Linder, T., Tretyakov, V., Surmann, H., Svoboda, T., Reintein, M., Zimmermann, K., PetrÃ■cek, T. & Hlavac, V. (2014), ‘Designing, developing, and deploying systems to support human-robot teams in disaster response’, *Advanced Robotics* **28**(23), 1547–1570. **URL:** *http://dx.doi.org/10.1080/01691864.2014.985335*

Kumar, V. & Michael, N. (2012), ‘Opportunities and challenges with autonomous micro aerial vehicles’, *The International Journal of Robotics Research* **31**(11), 1279–1291.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D. & Huntsberger, T. L. (2014), ‘Safe Maritime Autonomous Navigation With COLREGS, Using Velocity Obstacles’, *IEEE Journal of Oceanic Engineering* **39**(1), 110–119.

Lange, S., Sunderhauf, N. & Protzel, P. (2009), 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, pp. 1–6.

Larson, J., Bruch, M., Ebken, J., Warfare, N., Diego, S. & Diego, S. (2007a), 'Autonomous Navigation and Obstacle avoidance for unmanned surface vehicles', *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* pp. 17–20.

Larson, J., Bruch, M., Halterman, R., Rogers, J. & Webster, R. (2007b), 'Advances in Autonomous Obstacle Avoidance for Unmanned Surface Vehicles', *Techniques* pp. 1–15.

Lazarowska, A. (2015), 'Ship's trajectory planning for collision avoidance at sea based on ant colony optimisation', *Journal of Navigation* **68**(02), 291–307.

Lebbad, A. & Nataraj, S. (2015), A bayesian algorithm for vision based navigation of autonomous surface vehicles., *in* 'RAM/CIS', IEEE, pp. RAM:59–64.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. et al. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Lee, C. Y. (1961), 'An algorithm for path connections and its applications', *IRE Transactions on Electronic Computers* **EC-10**(3), 346–365.

Lee, D., Ryan, T. & Kim, H. J. (2012), Autonomous landing of a vtol uav on a moving platform using image-based visual servoing, *in* 'Robotics and Automation (ICRA), 2012 IEEE International Conference on', IEEE, pp. 971–976.

Lee, S.-M., Kwon, K.-Y. & Joh, J. (2004), 'A fuzzy logic for autonomous navigation of marine vehicles satisfying colreg guidelines', *International Journal of Control Automation and Systems* **2**, 171–181.

Lee, T., Kim, H., Chung, H., Bang, Y. & Myung, H. (2015), 'Energy efficient path plan-

ning for a marine surface vehicle considering heading angle', *Ocean Engineering* **107**, 118–131.

Lee, Y.-I. & Kim, Y.-G. (2004), *Intelligent Data Engineering and Automated Learning – IDEAL 2004: 5th International Conference, Exeter, UK. August 25-27, 2004. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, chapter A Collision Avoidance System for Autonomous Ship Using Fuzzy Relational Products and COLREGs, pp. 247–252.

Leng, J., Liu, J. & Xu, H. (2013), 'Online Path Planning based on MILP for Unmanned Surface Vehicles', *Oceans 2013- San Diego* pp. 1–7.

Levine, S., Finn, C., Darrell, T. & Abbeel, P. (2016), 'End-to-end training of deep visuomotor policies', *Journal of Machine Learning Research* **17**(39), 1–40.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J. & Quillen, D. (2016), 'Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection', *The International Journal of Robotics Research* p. 0278364917710318.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D. (2015), 'Continuous control with deep reinforcement learning', *arXiv preprint arXiv:1509.02971* .

Lin, S., Garratt, M. A. & Lambert, A. J. (2017), 'Monocular vision-based real-time target recognition and tracking for autonomously landing an uav in a cluttered shipboard environment', *Autonomous Robots* **41**(4), 881–901.

Linchant, J., Lisein, J., Semeki, J., Lejeune, P. & Vermeulen, C. (2015), 'Are unmanned aircraft systems (uass) the future of wildlife monitoring? a review of accomplishments and challenges', *Mammal Review* **45**(4), 239–252.

Liu, Y.-H., Du, X.-M. & Yang, S.-H. (2006), The design of a fuzzy-neural network for ship collision avoidance, *in* 'Advances in Machine Learning and Cybernetics', Springer, pp. 804–812.

Liu, Z., Zhang, Y., Yu, X. & Yuan, C. (2016), 'Unmanned surface vehicles: An overview of developments and challenges', *Annual Reviews in Control* **41**, 71–93.

Maini, R. & Aggarwal, H. (2009), 'Study and Comparison of Various Image Edge Detection Techniques', *International Journal of Image Processing* **147002**(3), 1–12. **URL:** *http://wwwmath.tau.ac.il/ turkel/notes/Maini.pdf*

Mamdani, E. H. & Assilian, S. (1975), 'An experiment in linguistic synthesis with a fuzzy logic controller', *International journal of man-machine studies* **7**(1), 1–13.

McCulloch, W. S. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics* **5**(4), 115–133.

Minaeian, S., Liu, J. & Son, Y. (2016), 'Vision-based target detection and localization via a team of cooperative uav and ugvs', *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.* **46**(7), 1005–1016.

Minsky, M. & Papert, S. A. (1969), *Perceptrons: An introduction to computational geometry*, MIT press.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. & Kavukcuoglu, K. (2016), Asynchronous methods for deep reinforcement learning, *in* 'International Conference on Machine Learning', pp. 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. et al. (2015), 'Human-level control through deep reinforcement learning', *Nature* **518**(7540), 529–533.

Moore, T. & Stouch, D. (2016), *A Generalized Extended Kalman Filter Implementation for the Robot Operating System*, Springer International Publishing, Cham, pp. 335–348.

Murphy, D. W. & Cycon, J. (1999), Applications for mini vtol uav for law enforcement, *in* 'Enabling technologies for law enforcement and security', International Society for Optics and Photonics, pp. 35–43.

Murphy, R. R., Steimle, E., Griffin, C., Cullins, C., Hall, M. & Pratt, K. (2008), 'Cooperative use of unmanned sea surface and micro aerial vehicles at hurricane wilma', *Journal of Field Robotics* **25**(3), 164–180.

Naeem, W. & Sutton, R. (2009), 'An Intelligent Integrated Navigation and Control Solution for an Unmanned Surface Craft', *IET Irish Signals and Systems Conference (ISSC 2009)* pp. 9–9.
URL: *http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5524711' escapeXml='false'/>*

Narasimhan, K., Kulkarni, T. & Barzilay, R. (2015), 'Language understanding for text-based games using deep reinforcement learning', *arXiv preprint arXiv:1506.08941* .

Neumann, P. P., Bennetts, V. H., Lilienthal, A. J., Bartholmai, M. & Schiller, J. H. (2013), 'Gas source localization with a micro-drone using bio-inspired and particle filter-based algorithms', *Advanced Robotics* **27**(9), 725–738.
URL: *http://dx.doi.org/10.1080/01691864.2013.779052*

Ng, A., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E. & Liang, E. (2006), 'Autonomous inverted helicopter flight via reinforcement learning', *Experimental Robotics IX* pp. 363–372.

Niu, H., Lu, Y., Savvaris, A. & Tsourdos, A. (2016), 'Efficient path planning algorithms for unmanned surface vehicle', *IFAC-PapersOnLine* **49**(23), 121–126.

Nonami, K., Kendoul, F., Suzuki, S., Wang, W. & Nakazawa, D. (2010), *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles*, 1st edn, Springer Publishing Company, Incorporated.

Park, K. T. & Jeong, J. (2012), Object detection in infrared image with sea clutter, *in* 'Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on', pp. 2143–2146.

Patacchiola, M. & Cangelosi, A. (2017), 'Head pose estimation in the wild using convolutional neural networks and adaptive gradient methods', *Pattern Recognition* .

Pavel, M. S., Schulz, H. & Behnke, S. (2017), 'Object class segmentation of rgb-d video using recurrent convolutional neural networks', *Neural Networks* **88**, 105–113.

Pereira, E., Bencatel, R., Correia, J., Félix, L., Gonçalves, G., Morgado, J. & Sousa, J. (2009), 'Unmanned air vehicles for coastal and environmental research', *Journal of Coastal Research* pp. 1557–1561.

Perera, L., Carvalho, J. & Soares, C. G. (2012), 'Intelligent ocean navigation and fuzzy-bayesian decision/action formulation', *IEEE Journal of Oceanic Engineering* **37**(2), 204–219.

Perera, L. P., Ferrari, V., Santos, F. P., Hinostroza, M. A. & Soares, C. G. (2015), 'Experimental evaluations on ship autonomous navigation and collision avoidance by intelligent guidance', *IEEE Journal of Oceanic Engineering* **40**(2), 374–387.

Peters, J. & Schaal, S. (2008), 'Reinforcement learning of motor skills with policy gradients', *Neural networks* **21**(4), 682–697.

Pinto, E., Santana, P. & Barata, J. (2013), On collaborative aerial and surface robots for environmental monitoring of water bodies, *in* 'Doctoral Conference on Computing, Electrical and Industrial Systems', Springer, pp. 183–191.

Polvara, R., Patacchiola, M., Sharma, S., Wan, J., Manning, A., Sutton, R. & Cangelosi, A. (2018), Toward end-to-end control for uav autonomous landing via deep reinforcement learning, *in* '2018 International Conference on Unmanned Aircraft Systems (ICUAS)', pp. 115–123.

Polvara, R., Sharma, S., Wan, J., Manning, A. & Sutton, R. (2017), Towards autonomous landing on a moving vessel through fiducial markers, *in* '2017 European Conference on Mobile Robots (ECMR)', pp. 1–6.

Polvara, R., Sharma, S., Wan, J., Manning, A. & Sutton, R. (2018), 'Vision-based autonomous landing of a quadrotor on the perturbed deck of an unmanned surface vehicle', *Drones* **2**(2), 15.

Polvara, R., Sharma, S., Wan, J., Manning, A. & Sutton, R. (2019), 'Autonomous vehicular landings on the deck of an unmanned surface vehicle using deep reinforcement learning', *Robotica* pp. 1–16.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. & Ng, A. Y. (2009), Ros: an open-source robot operating system, *in* 'ICRA workshop on open source software', Vol. 3, Kobe, Japan, p. 5.

Ren, S., He, K., Girshick, R. & Sun, J. (2015), Faster r-cnn: Towards real-time object detection with region proposal networks, *in* 'Advances in neural information processing systems', pp. 91–99.

Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychological review* **65**(6), 386.

Ruffier, F. & Franceschini, N. (2015), 'Optic flow regulation in unsteady environments: a tethered mav achieves terrain following and targeted landing over a moving platform', *Journal of Intelligent & Robotic Systems* **79**(2), 275–293.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. et al. (1988), 'Learning representations by back-propagating errors', *Cognitive modeling* **5**(3), 1.

Saripalli, S., Montgomery, J. F. & Sukhatme, G. S. (2003), 'Visually guided landing of an unmanned aerial vehicle', *IEEE transactions on robotics and automation* **19**(3), 371–380.

Saripalli, S. & Sukhatme, G. (2006), Landing on a moving target using an autonomous helicopter, *in* 'Field and service robotics', Springer, pp. 277–286.

Schaul, T., Quan, J., Antonoglou, I. & Silver, D. (2015), 'Prioritized experience replay', *arXiv preprint arXiv:1511.05952* .

Scherer, S., Chamberlain, L. & Singh, S. (2012), 'Autonomous landing at unprepared sites by a full-scale helicopter', *Robotics and Autonomous Systems* **60**(12), 1545–1562.

Schmidhuber, J. (2015), 'Deep learning in neural networks: An overview', *Neural networks* **61**, 85–117.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. & Moritz, P. (2015), Trust region policy optimization, *in* 'International Conference on Machine Learning', pp. 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. (2017), 'Proximal policy optimization algorithms', *arXiv preprint arXiv:1707.06347* .

Schuster, M., Blaich, M. & Reuter, J. (2014), 'Collision Avoidance for Vessels using a Low-Cost Radar Sensor', *19th The International Federation of Automatic Control, Cape Town, South Africa* pp. 9673–9678.

Sereewattana, M., Ruchanurucks, M. & Siddhichai, S. (2014), Depth estimation of markers for uav automatic landing control using stereo vision with a single camera, *in* 'Int. Conf. Information and Communication Technology for Embedded System.'.

Shakernia, O., Ma, Y., Koo, T. J. & Sastry, S. (1999), 'Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control', *Asian journal of control* **1**(3), 128–145.

Shi, H. & Wang, H. (2009), A vision system for landing an unmanned helicopter in a complex environment, *in* 'Sixth International Symposium on Multispectral Image Processing and Pattern Recognition', International Society for Optics and Photonics, pp. 74962G–74962G.

Shim, D., Chung, H., Kim, H. J. & Sastry, S. (2005), Autonomous exploration in unknown urban environments for unmanned aerial vehicles, *in* 'Proc. AIAA GN&C Conference'.

Simetti, E., Torelli, S., Casalino, G. & Turetta, A. (2014), 'Experimental results on obstacle avoidance for high speed unmanned surface vehicles', *Oceans 2014 - St. John's* pp. 1–6.

Simmons, R. & Henriksen, L. (1996), 'Obstacle Avoidance and Safeguarding for a Lunar Rover', *AIAA Forum on Advanced Developments in Space Robotics* .
URL: *http://www-2.cs.cmu.edu/afs/cs/usr/reids/www/home/papers/AIAAobsAvoid.pdf*

Smierzchalski, R. (1999), 'Evolutionary trajectory planning of ships in navigation traffic areas', *Journal of marine science and technology* **4**(1), 1–6.

Smierzchalski, R. & Michalewicz, Z. (2000), 'Modeling of ship trajectory in collision situations by an evolutionary algorithm', *IEEE Transactions on Evolutionary Computation* **4**(3), 227–241.

Stacy, N., Craig, D., Staromlynska, J. & Smith, R. (2002), The global hawk uav australian deployment: imaging radar sensor modifications and employment for maritime surveillance, *in* 'Geoscience and Remote Sensing Symposium, 2002. IGARSS'02. 2002 IEEE International', Vol. 2, IEEE, pp. 699–701.

Statheros, T., Howells, G. & Mcdonald-maier, K. (2008), 'Autonomous Ship Collision Avoidance Navigation Concepts , Technologies and Techniques', *The Journal of Navigation* **61**, 129–142.

Sutton, R. S. & Barto, A. G. (1998), *Introduction to Reinforcement Learning*, 1st edn, MIT Press, Cambridge, MA, USA.

Szlapczynski, R. (2006), 'A New Method of Ship Routing on Raster Grids, with Turn Penalties and Collision Avoidance', *The Journal of Navigation* **59**, 27–42.
URL: *http://www.journals.cambridge.org/abstract_S0373463305003528*

Szlapczynski, R. (2013), 'Evolutionary sets of safe ship trajectories within traffic separation schemes', *The Journal of Navigation* **66**(1), 65.

Szlapczynski, R. (2015), 'Evolutionary planning of safe ship tracks in restricted visibility', *Journal of Navigation* **68**(01), 39–51.

Tam, C., Bucknall, R. & Greig, A. (2009), 'Review of Collision Avoidance and Path Planning Methods for Ships in Close Range Encounters', *The Journal of Navigation* **62**, 455–476.
URL: *http://discovery.ucl.ac.uk/108866/*

Tan, A., Wee, W. C. & Tan, T. J. (2010), 'Criteria and Rule Based Obstacle Avoidance for USVs', *2010 International WaterSide Security Conference* pp. 1–6.

Tang, D., Hu, T., Shen, L., Zhang, D., Kong, W. & Low, K. H. (2016), 'Ground stereo vision-based navigation for autonomous take-off and landing of uavs: a chan-vese model approach', *International Journal of Advanced Robotic Systems* **13**(2), 67.

Tang, P., Zhang, R., Liu, D., Zou, Q. & Shi, C. (2012), 'Research on Near-field Obstacle Avoidance for Unmanned Surface Vehicle Based on Heading Window', *Proceedings of the 24th Chinese Control and Decision Conference, CCDC 2012* pp. 1262–1267.

Theodore, C., Rowley, D., Ansar, A., Matthies, L., Goldberg, S., Hubbard, D. & Whalley, M. (2006), Flight trials of a rotorcraft unmanned aerial vehicle landing autonomously at unprepared sites, *in* 'Annual Forum Proceedings-American Helicopter Society', Vol. 62(2), AMERICAN HELICOPTER SOCIETY, INC, p. 1250.

Thrun, S. & Schwartz, A. (1993), Issues in using function approximation for reinforcement learning, *in* 'Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum'.

Tieleman, T. & Hinton, G. (2012), 'Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude', *COURSERA: Neural networks for machine learning* .

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W. & Abbeel, P. (2017), 'Do-

main randomization for transferring deep neural networks from simulation to the real world', *arXiv preprint arXiv:1703.06907* .

Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I. L., Ruess, F., Suppa, M. & Burschka, D. (2012), 'Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue', *IEEE robotics & automation magazine* **19**(3), 46–56.

Tsou, M.-C. (2010), 'Integration of a geographic information system and evolutionary computation for automatic routing in coastal navigation', *Journal of Navigation* **63**(02), 323–341.

Van Hasselt, H., Guez, A. & Silver, D. (2016), Deep reinforcement learning with double q-learning., *in* 'AAAI', pp. 2094–2100.

Vetrella, A. R., Sa, I., Popović, M., Khanna, R., Nieto, J., Fasano, G., Accardo, D. & Siegwart, R. (2018), Improved tau-guidance and vision-aided navigation for robust autonomous landing of uavs, *in* 'Field and Service Robotics', Springer, pp. 115–128.

Wang, H., Wei, Z., Ow, C. S., Ho, K. T., Feng, B. & Huang, J. (2012), 'Improvement in real-time obstacle detection system for usv', *12th International Conference on Control Automation Robotics Vision (ICARCV), 2012* pp. 1317–1322.

Wang, H., Wei, Z., Wang, S., Ow, C. S., Ho, K. T. & Feng, B. (2011), 'A Vision-based Obstacle Detection System for Unmanned Surface Vehicle', *IEEE Conference on Robotics, Automation and Mechatronics, RAM* pp. 364–369.

Watts, A. C., Ambrosia, V. G. & Hinkley, E. A. (2012), 'Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use', *Remote Sensing* **4**(6), 1671–1692.

WawrzyńSki, P. & Tanwani, A. K. (2013), 'Autonomous reinforcement learning with experience replay', *Neural Networks* **41**, 156–167.

Wenzel, K. E., Rosset, P. & Zell, A. (2010), 'Low-cost visual tracking of a landing place and hovering flight control with a microcontroller', *Journal of Intelligent & Robotic Systems* **57**(1), 297–311.

Widrow, B. et al. (1960), *Adaptive" adaline" Neuron Using Chemical" memistors.".*

Wilson, P. A., Harris, C. J. & Hong, X. (2003), 'A line of sight counteraction navigation algorithm for ship encounter collision avoidance', *The Journal of Navigation* **56**, 111–121.

Xie, S., Wu, P., Peng, Y., Luo, J., Qu, D., Li, Q. & Gu, J. (2014), 'The obstacle avoidance planning of usv based on improved artificial potential field', *IEEE International Conference on Information and Automation (ICIA), 2014* pp. 746–751.

Yakimenko, O. A., Kaminer, I. I., Lentz, W. J. & Ghyzel, P. (2002), 'Unmanned aircraft navigation for shipboard landing using infrared vision', *IEEE Transactions on Aerospace and Electronic Systems* **38**(4), 1181–1200.

Yu, Z. & Zhang, C. (2015), Image based static facial expression recognition with multiple deep network learning, *in* 'Proceedings of the 2015 ACM on International Conference on Multimodal Interaction', ACM, pp. 435–442.

Zadeh, L. A. (1965), 'Fuzzy sets', *Information and control* **8**(3), 338–353.

Zeng, X. (2003*a*), 'Evolution of the safe path for ship navigation', *Applied Artificial Intelligence* **17**(2), 87–104.
URL: *http://dx.doi.org/10.1080/713827101*

Zeng, X. (2003*b*), 'Evolution of the safe path for ship navigation', *Applied artificial intelligence* **17**(2), 87–104.

Zhang, R., Tang, P., Su, Y., Li, X., Yang, G. & Shi, C. (2014), 'An Adaptive Obstacle Avoidance Algorithm for Unmanned Surface Vehicle in Complicated Marine Environments', *IEEE/CAA Journal of Automatica Sinica* **1**(4), 385–396.

Zhang, T., Kahn, G., Levine, S. & Abbeel, P. (2016), Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search, *in* 'Robotics and Automation (ICRA), 2016 IEEE International Conference on', IEEE, pp. 528–535.

Zhou, D., Zhong, Z., Zhang, D., Shen, L. & Yan, C. (2015), Autonomous landing of a helicopter uav with a ground-based multisensory fusion system, *in* 'Seventh International Conference on Machine Vision (ICMV 2014)', International Society for Optics and Photonics, pp. 94451R–94451R.