Automatic
Control &
Systems
Engineering.

Vision-Based Localisation and Tracking of a UGV with a Quadcopter

Xiaotian DAI

September, 2014

Supervisor: Dr T J Dodd

A dissertation submitted in partial fulfilment of the requirements for the degree of M. Sc. (Eng.) in Control Systems

EXECUTIVE SUMMARY

This work represents a visual servoing framework of a low cost UAV (*unmanned aerial vehicle*) platform. This system is consisted of a Blob visual detector and a PID position controller. The designed UAV system has the ability to follow and track a moving ground vehicle in real-time. Plenty of experiments have been done to examine the performance of this system including the stability of position holding, the robustness against external forces and the success rate of visual detection during flight tests. The experimental results prove the usability and effectiveness of the proposed system.

INTRODUCTION / BACKGROUND

The UAVs have enormous applications these days in military, searching and rescuing, civilian surveillance and remote sensing. The *quadcopter*, as a special class of UAVs, has the ability to perform aggressive maneuvers and is of great interested by academic researchers. However, the ability of single UAV is limited and thus the cooperation with other vehicles is an essential for executing more complex tasks. In a UGV-UAV cooperative task, it is often required that the UAV should be able to track the team leader autonomously. The use of visual servoing is a possible solution to such a problem and the design of a visual servoing system is the main concern of this work.

AIMS AND OBJECTIVES

This project is aimed to develop a ground vehicle tracking system of an UAV based on computer vision in a GPS-denied environment. The UAV is designed to search and find the ground vehicle with the view from its bottom camera. Once found the vehicle, the UAV should be able to track it with vision feedback. A ground station running Linux is also designed to control the UAV and the UGV and monitor their current statuses.

ACHIEVEMENTS

A framework for UAV visual servoing was proposed in this work. The system is based on the ROS system and the AR.Drone quadcopter platform. A Blob visual detection algorithm was designed to locate the position of a coloured marker and a position controller was used to control the movement of the UAV. The designed visual servoing system has the ability to hover at a 3-D point, follow a pre-defined figure path and track and follow a moving vehicle.

CONCLUSIONS / RECOMMENDATIONS

The designed system was proved to be effective and robust in plenty of test flights. The system is able to accept considerable disturbances and can estimate its relative position to the target and re-find the UGV when it was lost. Further improvements could be done on system modelling, shape analysis of the detected object and to utilise on-board image processing.

ABSTRACT

This work represents a framework for visual detection and tracking on a quadrotor helicopter platform. The proposed system, which includes a UGV and an AR.Drone quadcopter, uses centralized control based on a ground station running ROS system. A coloured marker is attached on the top of the UGV, serving as the target of the Blob Detection algorithm. The detected position of the object is transformed from pixel frame into a 3-D relative position by a transformation projection. The solved coordinates are used as the references of the position controller to control the movement of the quadcopter. The position controller is a combination of four individual PID controllers and each of them controls one state of interest of the system. The designed visual system was tested under different experiments and has a satisfactory control performance against disturbances and accidental events.

Keywords: quadcopter, visual servoing, blob detection, PID

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Dr T J Dodd. He provided me with the hardware and resources I needed for this research and gave me properly guidance during the whole period of this project. This work would not be accomplished without his kindly help.

I would also like to show my gratitude to Dr J. Aitken who encouraged me by demonstrating a video of his research with the AR.Drone quadcopter. This gave me a glimpse of the research value of the AR.Drone platform. I also want to show my appreciation to my college Chuhao Liu. Thanks him for showing me the Gazebo simulator and some ideas about his project on multi UAVs.

I am also grateful that my college Yingyi Kuang could assisted me during the test flights and helped me with the video shooting for the final demonstration.

During my master's course, I learnt advanced knowledge on control systems and equipped myself with necessary research skills for further study. This would not be achieved without all the teachers and professors who ever helped me during my study in the Department of ACSE in the University of Sheffield. Thanks them for theirs' state-of-art teaching methodology and extensive knowledge on control systems.

Finally, special thanks to my parents whose love to me will never be forgot.

Xiaotian DAI

Aug 2014

TABLE OF CONTENTS

ABBREVIATIONS

Abbr.	**Description**
MAV	**M**icro **A**erial **V**ehicle
UAV	**U**nmanned **A**erial **V**ehicle
UGV	**U**nmanned **G**round **V**ehicle
VTOL	**V**ertical **T**ake-**o**ff and **L**anding
EKF	**E**xtended **K**alman **F**ilter
KF	**K**alman **F**ilter
SLAM	**S**imultaneous **L**ocalization **a**nd **M**apping
V-SLAM	**V**ision-based **S**imultaneous **L**ocalization **a**nd **M**apping
VS	**V**isual **S**ervoing
IBVS	**I**mage **B**ased **V**isual **S**ervoing
PnP	**P**erspective-**n**-**P**oint
FOV	**F**ield **o**f **V**iew
PEM	**P**rediction **E**rror **M**ethods
LTI	**L**inear-**T**ime-**I**nvariant
TF	**T**ransfer **F**unction
MIMO	**M**ulti-**I**nput-**M**ulti-**O**utput
SISO	**S**ingle-**I**nput-**S**ingle-**O**utput
DOF	**D**egrees **o**f **F**reedom
PID	**P**roportional-**I**ntegral-**D**erivative
MPC	**M**odel-based **P**redictive **C**ontrol
ROS	**R**obot **O**perating **S**ystem
API	**A**pplication **P**rogramming **I**nterface
SDK	**S**oftware **D**evelopment **K**it
FSM	**F**inite **S**tate **M**achine

LIST OF SYMBOLS

Symbol	Description
x, y, z	x, y, z Position
$\dot{x}, \dot{y}, \dot{z}$	x, y, z Speed
ϕ, θ, ψ	Roll, Pitch, Yaw (Euler Angles)
$\dot{\phi}, \dot{\theta}, \dot{\psi}$	Roll, Pitch, Yaw Angular Rate
X_d	Desired States of Interest
X_c	Current States of Interest
q_d	Desired System States
q_c	Current System States
${}^{G}X$	X in the Ground Frame
${}^{C}X$	X in the Camera Frame
${}^{B}_{A}T$	Coordinate Transform Matrix ($A \rightarrow B$)
$L_{(P_A,P_B)}$	Pixel Distance between Point A and Point B
κ_p	Proportional Gain of the PID Controller
κ_i	Integral Gain of the PID Controller
κ_d	Derivative Gain of the PID Controller
δ_k	Sampling Interval at Discrete Time k
e_x	Position Error in x axis
e_y	Position Error in y axis
e_d	Relative Distance Error

List of Figures

List of Tables

Chapter 1

Introduction

This chapter describes the background and objectives of this research. The first section gives a brief view of the applications using UAVs. Different types of UAVs are discussed and quadcopter, as a special kind of UAV, is highlighted for its agility and flexibility. The next section declares the problem involved in this study as well as the aims and objectives of this project. The project management is then followed, describing the time schedule and resource inputs and outputs of this work. The organisation of this thesis is given as the last part of this chapter.

1.1 Background and Motivation

1.1.1 UAV Applications

Unmanned aerial vehicle, also known as UAV, is defined as an aerial vehicle which can perform complex tasks without or with a limit of human interference. Nowadays, UAVs have shown their immense utility and have been widely in numerous domains ranging from military operations, civilian surveillance and monitoring, searching and rescuing to remote sensing of agriculture land [23].

It has no doubt that the first application of UAV is in military use. The most famous military UAV platform so far is probably the RQ-4 Global Hawk developed under United States Air Force (Fig. 1.1). Global Hawk is a high-altitude UAV platform for surveillance and security purpose. It is equipped with a multi-mode radar and electro-optical visible/infra-red sensor and is able to execute tasks continuously for more than 28 hours. Its prototype the Global Hawk ATCD have

been used in the wars in Afghanistan and Iraq. In April 2001 Global Hawk carried out eleven sorties in Australia with maritime surveillance as the main operational focus [34].

Fig. 1.1: Global Hawk 1 by U.S. Air Force photo by Bobbi Zapka. Global Hawk is one of the most famous military UAV platform. It has a cruising altitute of 60,000 ft and a maximum speed of approximately 575 km/h. Picture reproduced from [40].

Fig. 1.2: ResQu Unmanned Aircraft designed to help fighting natural disasters in Australia. Reprinted from `http://www.unmannedsystemstechnology.com/` .

Another important utilization of UAVs is in disaster response (Figure 1.2). The main role of UAVs in such applications is to acquire RGB or infra-red images with remote sensing instruments. The high-resolution images taken from the UAV

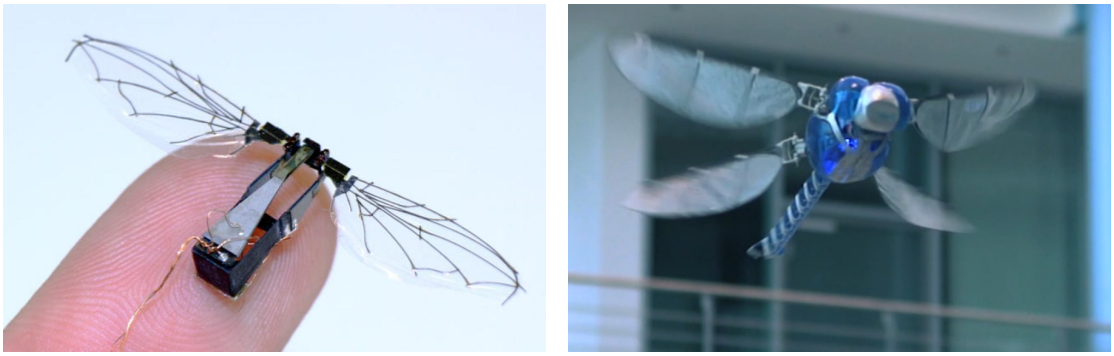
will then be transmitted back to the ground station and be analysed by experts to produce surface models, building structures and elevation maps. These data is of magnificent importance to survival searching, access route investigation and rescue coordination. A brief review of recent utilizations of UAVs for monitoring after a disaster is given in [1]. Another example is discussed by [8], which describes the use of UAVs in obtaining advanced warning and damage assessments for tornados and thunderstorms.

1.1.2 Micro Aerial Vehicle

Micro Aerial Vehicle, or MAV, is defined as a class of UAV with a restricted size and limited payload. The size of MAVs ranges from a dimension of about $0.5m \times 0.5m$ to the size of a insect. Some of the MAVs are bio-inspired and use flapping wings to fly, e.g. the RoboBee by Harvard University and the dragonfly-like BionicOpter MAV (Fig. 1.3). The others, which are more common to see, use fixed or rotary wings. The fixed-wing aircraft has a long history and thus is better understood. However, there is a trend in studying the rotary-wing UAVs these days. The advantage of the rotary-wing MAV, comparing to its fixed-wing counterpart, is the ability to hover over a object and to take off and land vertically. The rotary-wing UAV is also quoted as *VTOL* (*Vertical Take Off and Landing*) platform in some papers. In order to keep consistency, the same convention will be used throughout the rest of this thesis. The general configurations of rotary-wing MAVs include single rotor , three rotors (*Trirotor*), four rotors (*Quadrotor*), six rotors (*Hexrotor*) and eight rotors (*Octorotor*). Among them, the single rotor MAV was first introduced which has the same mechanism with a full-size helicopter and as the multi-rotors, the Quadrotor copter is most investigated and applied.

1.1.3 Features of Quadrotor

The VTOL platform configured with four rotors is known as quadrotor helicopter, also known as quadcopter. The use of multiple rotors solves the problem of unbalanced torque caused by using one propeller for propulsion. The advantage of a quadcopter is the using of fixed pitch propellers, so it is mechanically simpler. However, the quadcopter is inherently unstable and it is relative difficult to con-



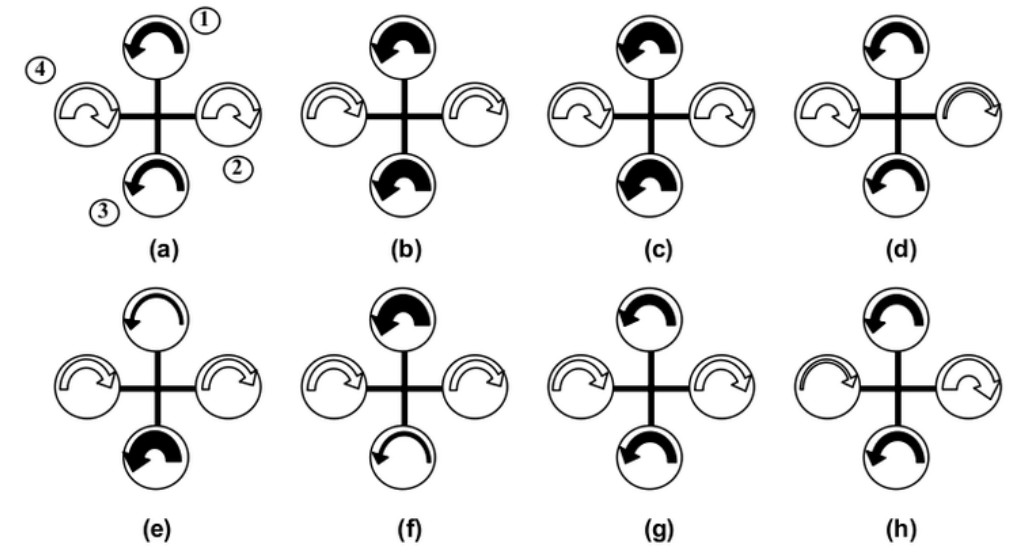
(a) (b)

Fig. 1.3: Examples of two bio-inspired micro aerial vehicle. **(a)** The RoboBee micro UAV project developed at Harvard University [13]; **(b)** The FESTO BionicOpter MAV inspired by dragonfly[11].

trol. Thus, special care should be addressed when designing the controller of a quadcopter.

The movement of the quadcopter is attained by changing the speeds of each rotor to create different propulsion forces. The mechanism of the movements against the propeller speeds is displayed in Figure 1.4. The quadcopter can fly both indoor and outdoor. When is operating outdoor, the quadcopter can gather its geographic coordinates from a GPS module. But when the quadcopter is undertaking an indoor flight, it cannot use the GPS information as well as the magnetometer to obtain its speed, position and orientation. In such scenes, the quadcopter should be able to estimate its attitude and position by using a on-board inertial navigation system (INS) or an external video capturing system.

There is a number of quadcopter platforms available in the market such as the AscTec Pelican (Figure 1.5), Draganflyer X4 (Figure 1.6), the AR.Drone and the open source quadcopter ArduCopter. These quadcopter platform usually comes with inertial measurement sensors, ultrasonic and communication equipments, and can perform aggressive maneuvers including ball juggling, flipping and flying with a pre-defined trajectory. These features and advantages of quadcopter make it an ideal platform for surveillance and monitoring in urban areas.



(a)	Yaw (anticlockwise direction)	**(e)**	Pitch (anticlockwise direction)
(b)	Yaw (clockwise direction)	**(f)**	Pitch (clockwise direction)
(c)	Take-off or take-up	**(g)**	Land or take-down
(d)	Roll (clockwise direction)	**(h)**	Roll (anticlockwise direction)

Fig. 1.4: The movement mechanism of quadcopters [3].

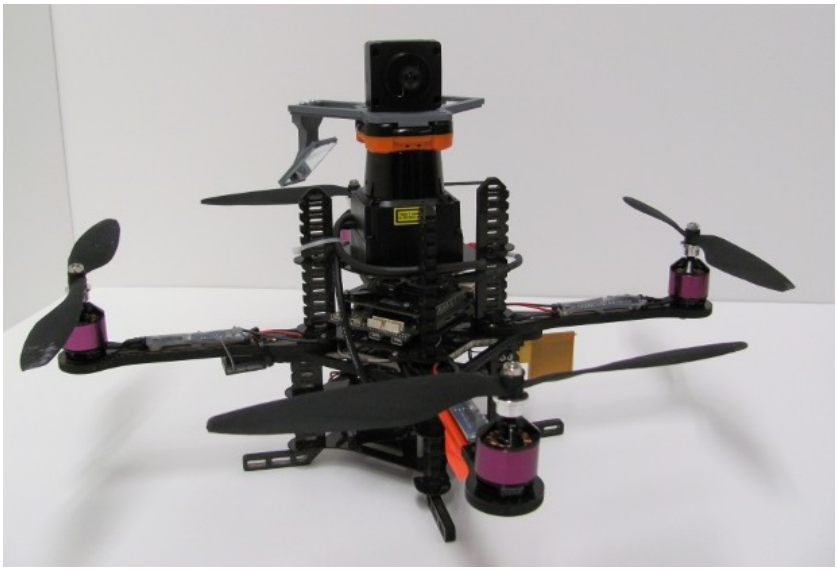


Fig. 1.5: AscTec Pelican Quadcopter. A state-of-art quadcopter comes with four 160W brushless motors, an optional laser radar and a camera mount. It has a empty weight of 620 g and a maximum payload of 650 g (Picture reproduced from `http://wiki.ros.org/Robots/AscTec`).

Fig. 1.6: The DragonFly X4-ES Quadcopter. This quadcoter is specifically designed for public safety use and has a vibration isolated camera gimble which permits to delivery high-quality real-time video[9].

1.1.4 Motivation of this Work

As mentioned above, there are many possible applications of the UAV. In most cases single UAV is enough to accomplish the task. However there are some circumstance where a UAV along is not enough for its limited payload and short endurance. To enhance the capability of single UAV, there is an emerging study that trying to make UAV(s) collaborate(s) with other robots to accomplish relative advanced tasks. Such system could be a swarm of UAVs or a hetergeneous team formed by UGV(s) and UAV(s). By using a cooperative formation, one can accomplish more complex tasks such as searching a large area, finding survivals after a disaster, building a infrastructure or moving a heavy mass. To contribute the UGV-UAV cooperative system is the main motivation of this project.

Another motivation of this work is the great potential of machine vision in robotic applications. Computer vision will boost the perception ability of robots to enable them perform more sophisticated work and it has already proven to be an important technique to industry assembling and inspection robots. UAVs, in a same manner, will have a higher level of autonomous if combined with on-board vision systems and there will be more potential use for such systems in military and civilian applications. In such a background, research into computer vision is important and meaningful.

1.2 Problem Definition

In the UGV-UAV cooperative task, it is frequently required that the UAV should be able to track and follow the ground vehicle. There are already some research in the literature on tracking a quadcopter with another quadcopter or realizing an autonomous taking-off and landing on a UGV, but there are few work been done so far to tackle the problem of tracking and following a UGV. In some circumstances, it is essential to keep the UAV hovering at a relative high altitude to provide a better sensing of the surroundings to other robots while tracking the UGV leader at the same time. Thus, the tracking problem is of equivalent importance to other issues.

The challenge of such tasks is the need of real-time image processing capability. The vision algorithm should be simple enough to be executed in real-time and a high performance controller is an essential to get a satisfactory result. Such a system can either be centralized or decentralized, depending on the computational power available on the UAV platform.

1.3 Aims and Objectives

The aim of this project is to develop a system to autonomously control an unmanned aerial vehicle (UAV) to track a ground vehicle using monocular vision in a GPS-denied environment. The UAV is designed to search and find the ground vehicle with its on-board bottom camera. Once found the vehicle, the UAV should also be able to track it with visual feedback. A ground station running ROS is designed to communicate with both the UAV and the UGV and monitor their current statuses. The Objectives of this project are:

- Design a framework for UAV visual servoing.
- Develop the software of the ground vehicle.
- Develop the vision algorithm to recognize and locate the ground vehicle.
- Develop the controller to track the ground vehicle.
- Design the strategy of the UAV to search for the UGV when lost the target.
- Develop the software of the ground station.

The contribution of this thesis is a framework of visual servoing system to enable the UAV to detect and track a UGV visually. This provides a foundation to other tasks, for instance, serving the UAV as a global sensor to provide information of obstacles or coordinating a formation team by calculating the relative positions of ground robots.

1.4 Project Management

This project (Master's Dissertation) takes a period of three and a half months and is intended to design a framework for UAV visual tracking. There are four parts of this project, namely the project proposal, the system design, the experiment tests and the dissertation report writing. The resources required by this project are: a quadcopter platform, a UGV platform, a web camera, a laboratory for flight tests, one student (40 hours / week) and one specified tutor. The expected outputs are

1. the designed software of the UGV
2. the designed visual servoing framework of the quadcopter
3. experimental results and conclusions
4. Aims and Objectives form
5. the dissertation report

Some key time points are given as:

- 12/05/2014: Starting date of the project.
- 02/06/2014: Proposal of the project aims and objectives.
- 01/07/2014: Finish the visual algorithm design.
- 03/08/2014: Finish the position controller design.
- 15/08/2014: Finish system Evaluation, improvements and experiments.
- 23/08/2014: Finish the draft of the dissertation report.
- 01/09/2014: Thesis Submission Date.
- 05/09/2014: Oral Presentation.

The risk involved in this project is the potential damage of the UAV during the flight tests. If this happens, the whole schedule could be delayed until a replacement quadcopter arrived. The other risk is that the project duration is

relative short for a hardware project and the specification may not satisfy the designed one. These problems could be overcome by careful conducting test flights and checking the work process regularly with the tutor. The revised Gantt diagram can be found in Fig.1.7. The real project progress is in accordance with most of the original plan. However, since the system improvement and experiments took too much time, the dissertation writing has been delayed for two weeks, which has been highlighted in the Gantt diagram.

1.5 Thesis Organization

The remainder of this thesis is organized as several chapters and each chapter has its own intention and focus. The second chapter presents a review of the literature and the relationship between this research and others'. This can be seen as the foundation of this work. *Chapter 3* gives a description of the main hardware and software platforms involved in this project, which includes the AR.Drone quadcopter, the 4WD Nexusrobot mobile robot platform, the Robot Operating System (ROS) and OpenCV. *Chapter 4* describes the basics of computer vision and gives a detail of all the related vision algorithms used in this work. *Chapter 5* is mainly focused on the control aspect where four PID controllers have been designed to control the motion of the drone to achieve visual servoing. *Chapter 6* gives the implementation of the system and *Chapter 7* shows some experiments of real flights and gives the performance evaluations of the designed system. Finally, the last chapter highlights the conclusion of this work as well as further work which could be done to improve the system performance.

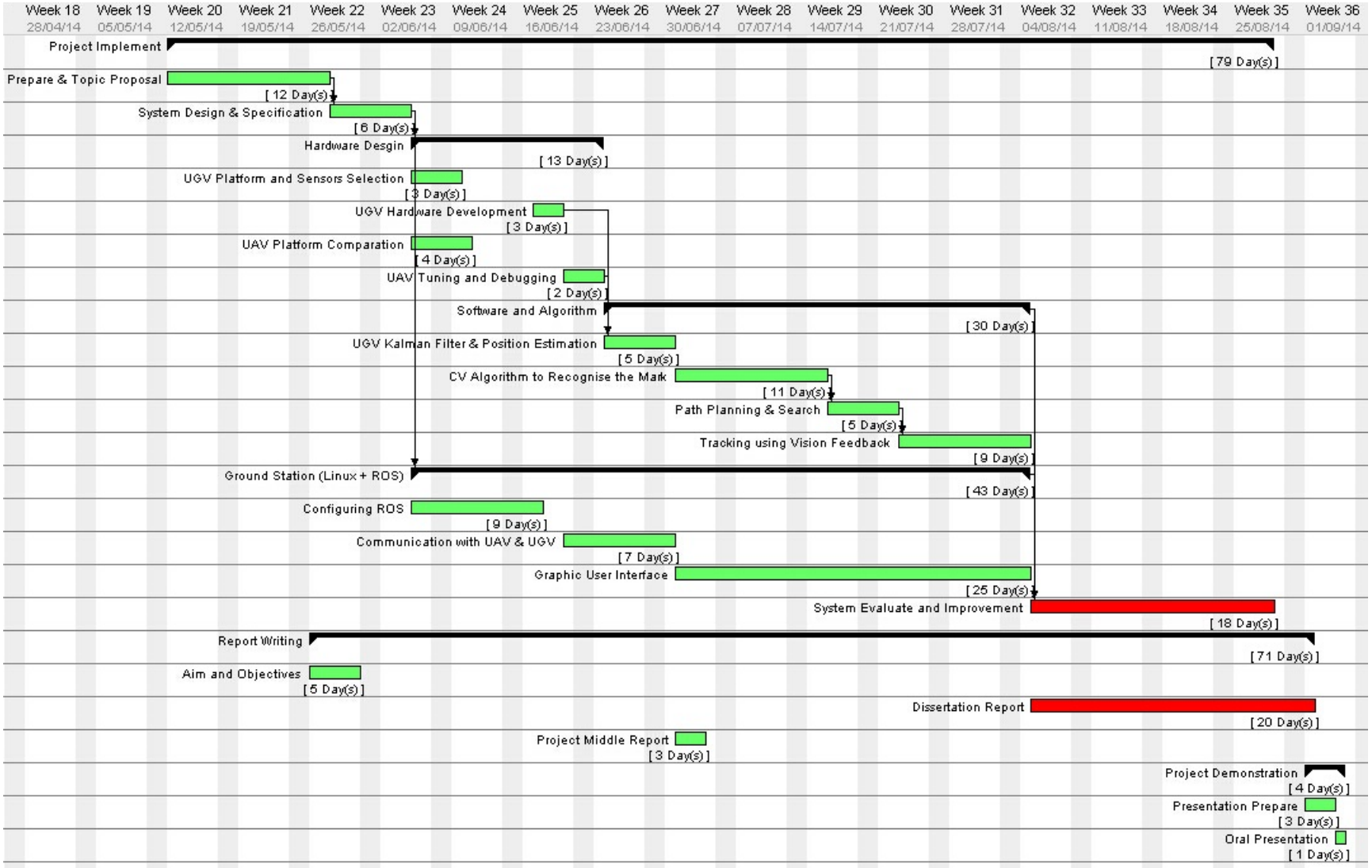


Fig. 1.7: Project Gantt Diagram (Revised)

Chapter 2

Related Work

In the last few years, enormous research have emerged in literature on the topic of autonomous quadcopters. These topics include system modelling, position estimation and control, visual servoing, Simultaneous Localization and Mapping (SLAM) and navigation with on-board sensing, etc.

This chapter is mainly focused on the literature review of the previous work. A selection of studies will be given in detail to develop a context for the research involved in this thesis. Most of these knowledge will be considered later in the design phase of visual servoing and the position controller.

2.1 Modelling of a Quadcopter

Before applied a controller for an aerial vehicle, the system model which presents the dynamics of the system should be first obtained. There are two major ways to obtain the system model. The first is using Newton's laws and Euler's equations to derive the model based on the physical characteristics of each component in the system [22] [21]. The other way, which is much practical, is to treat the system as a black box and use system identification techniques [30] [17] [14] [6].

A. Modelling from Physics

Deriving the quadcopter model from its physic laws is not a straight forward work, since the UAV is a non-linear MIMO (Multi-Input-Multi-Output) system and has aerodynamic characteristics. The dynamic model of the UAV is defined as the

relationship between the input T_n and the output $y = (\phi, \theta, \psi, x, y, z)^T$, where T_n is the thrust of each propeller and $\phi, \theta, \psi, x, y, z$ stands for roll, pitch, yaw and position, respectively. A typical model of the quadcopter can be described as [30]:

$$\begin{cases} I_x\ddot{\phi} = \dot{\psi}\dot{\theta}(I_y - I_z) + l(T_4 - T_2) \\ I_y\ddot{\theta} = \dot{\phi}\dot{\psi}(I_z - I_x) + l(T_1 - T_3) \\ I_z\ddot{\psi} = \dot{\theta}\dot{\phi}(I_x - I_z) + \sum_{i=1}^{4} M_i \\ m\ddot{x} = (sin\phi sin\psi + cos\phi sin\theta cos\psi)\sum_{i=1}^{4} T_i \\ m\ddot{y} = (cos\phi sin\theta sin\psi - sin\phi cos\psi)\sum_{i=1}^{4} T_i \\ m\ddot{z} = mg - cos\theta cos\phi \sum_{i=1}^{4} T_i \end{cases} \tag{2.1}$$

where m is the mass of the vehicle, (ϕ, θ, ψ) denotes the Euler Angles, (I_x, I_y, I_z) is the inertia moment of the rotor, T_i and M_i represents the force and torque generated by the ith propeller and l is the distance between two opposite propellers.

B. Modelling using System Identification

System identification methods can also be used to obtain the model of a UAV. Unlike building models from physics, this class of methods treat the system as a black box and analyse the system only with the input(s) and output(s) data. In [17], the drone is modelled with its internal controller in the loop and then this model is decomposed into four first-order or second-order systems. Each system is identified separately by means of least squares and verified with simulated data. PEM (*Prediction Error Methods*) is used in [14]. In this work, the quadrotor is treated as a Linear-Time-Invariant (LTI) system. A Pseudo-Random Binary Signal (PRBS) with a amplitude of $\pm 0.3v$ is generated by MATLAB as the input for identification. The pitch is represented by a first order transfer function and the altitude is estimated as a second-order transfer function. The experiments shows a fitting rate of 55.34% and 68.8% for pitch and altitude, respectively.

2.2 UAV Platform Design

There is an increasing interesting in developing UAV systems for different applications. Such systems can either be a new one developed from the ground up or a modification of an off-the-shelf platform. The applications of the UAVs differ, so there is no general solution to all these issues. Thus, development or modification of a UAV system becomes an important and crucial work for researchers to handle different circumstances.

A. Develop a New Platform

In some situations, there is no ready-to-use platform exists. Thus, it is essential to design a new platform to adapt a certain application. In Klas Nordberg et al. [2002], an autonomous helicopter platform namely the WITAS UAV platform was designed [25]. The goal of this project is to realize a fully autonomous UAV for tasks such as monitoring the traffic, emergency assistance and surveillance. The system is consisted of a helicopter platform with a on-board vision system, a software architecture for deliberative/reactive behaviour, a geographical database and a interface for ground operator. A PC104 board is being used as the main controller and a Sony FCB-EX470LP RGB video camera is used mounting on a stabilized gimbal. Another project is the DragonFly experimental platform by Stanford University [10]. This platform include two fixed wing UAVs, modular avionics hardware and a coordinate system for multiple UAVs. Other examples can also be found in the aforementioned RoboBee project [35], the AuRoRa hardware-in-the-loop platform [32] and the Berkeley UAV Platform [36], etc.

B. Modify the Hardware

While developing from the ground ensures the best opportunity for customization, modifying the existing hardware is more often the case. The KingLion rotorcraft platform is such a platform that build upon the ESky Big Lama Co-Axial helicopter in S.K. Phang et al.[2010] [31]. The designed autonomous helicopter has an on-board processor, an inertial measurement unit, an ultrasonic range finder and a Xbee-Pro wireless module.

In the work of [18], a Gumstix Overo Fire computer with a 700MHz Cortex A8 core is attached to an AR.Drone for real-time image processing, and in [39] an ATMega 1284P microcontroller is used additional to the Hummingbird Autopilot. This microcontroller is used for Zigbee communication, reading distance measurement from the IR sensors and obtaining data from a Wii Remote Camera to realise P3P (*Perspective-3-Point*) pose estimation.

Jacobo Lugo and Andreas Zeil exploited a low cost autonomous navigation framework with the AR.Drone in 2013 [19]. A proxy program is running on the AR.Drone to serve as a bridge between the AR.Drone and an AVR microcontroller. The drone and the controller is connected by the debugging port of the AR.Drone. The microcontroller undertakes the work of high-level control and it is consisted of pose estimation, path planning and position control module. The experimental results showed that this system is effective for trajectory following without the support from a ground station.

2.3 Using the AR.Drone as a Research Platform

The AR.Drone quad-rotor copter is widely used for its low cost and abundant sensors and there is a large number of researches using the AR.Drone as the platform [18] [28] [19] [33]. The paper [17] shows the usability of the AR.Drone to perform basic tasks such as position control, object following and navigation in an autonomous way. More advanced topics are also covered in this paper, for instance, formation coordination and autonomous surveillance.

A. Position control

The objective of position control is to move the drone from its current position (x, y, z) with a raw angle ψ to the desired pose (x_d, y_d, z_d, ψ_d). In [17], the yaw angle is simply controlled by a P control for its fast response and the altitude is controlled by a PD controller. For the position control, the desired position is first transformed into a relative distance to the drone and then this relative error is feed into a PD controller as the reference input. It is assumed that the yaw angular rate is small so that the forward speed is only influenced by pitch and the

sidewise speed is only controlled by the roll angle.

B. Visual-based Navagation

It is proposed in [17] that a visual navigation can be done by using the forward camera of the AR.Drone. In this work, Speeded Up Robust Features (SURF) are extracted from the camera. Firstly, the drone travels through a predefined path with the guidance of a human operator and creates a landmark map of the taught path. The map is then divided into several segmentations and each segmentation will be used as a sub-map. When travelling autonomously later, the drone will match its current view with the mapped ones to estimate a relative angle. The distance between segmentations is purely determined by dead reckoning. A experimental flight is shown in Figure 2.1.

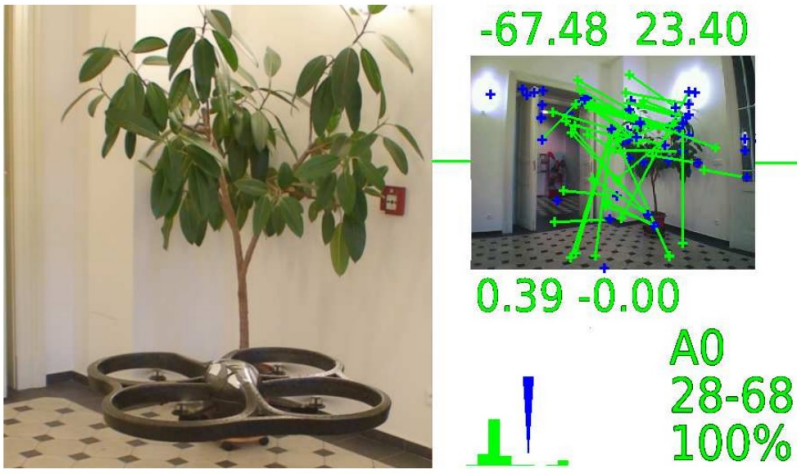


Fig. 2.1: Autonomous Visual Navigation of the AR.Drone [17]. The quadcopter was first travelled through the path and created a landmark map. In the autonomous flight, the quadcopter will match these features to find its relative position to a certain segment.

C. Mobile Localization System

In [17], the AR.Drone is also used as a mobile localization system. This system involves a lead-follower formation team consisting of a leader UGV, two followers and an quadcopter (Fig.2.2). The AR.Drone is used as the coordinator to provide the followers with their relative position to the leader. By doing this way, the

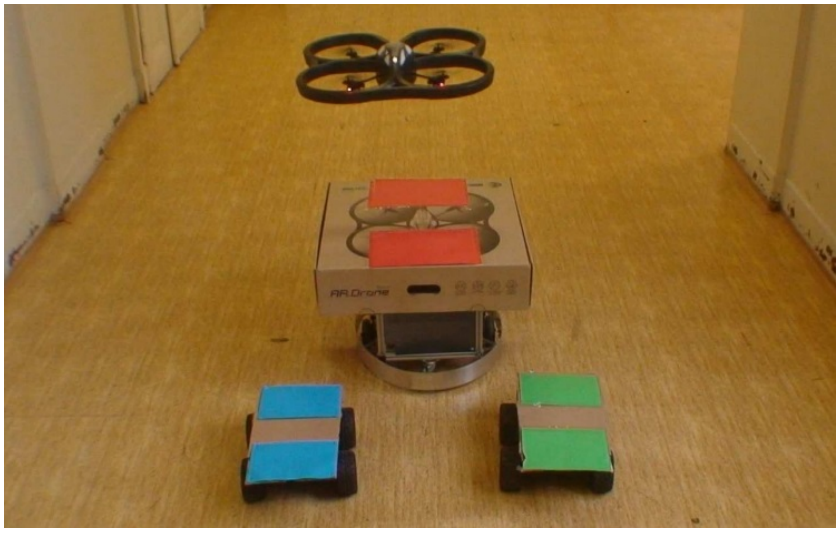


Fig. 2.2: AR.Drone used as a coordinator of a formation team [17]. In this experiment, the AR.Drone is served as the coordinator of the formation. The quadcopter calculates the relative position of each robot to the team leader and provides this information to other robots to adjust their positions.

team can adaptively change its shape to cope with obstacles while still keep the formation.

2.4 Control Architecture inside the AR.Drone

The AR.Drone has proven to be a low-cost and effective quadcopter for education and research and there is a increasing trend to design applications with the AR.Drone as the platform. In the work [5], the control architecture of the AR.Drone is introduced. While other study just regard the AR.Drone as a black box, this work is one of the rare ones that reveal the algorithms inside the AR.Drone platform.

A. Vision Algorithms

The bottom camera of the AR.Drone is able to estimate the speed using two complementary algorithms. The first method computers the optical flow over the whole image and the other uses a FAST-type corner tracking method. These two algorithms are able to be switched on-the-fly but can only be used either. From the experimental results, the optical flow method is less robust but can handle

low contrast scenarios while the corner tracking method is more accurate but has more requirements of the environment [5].

B. Attitude Estimation

In order to achieve a stable flight, the current state $q = (\dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\psi})$ of the drone should be estimated. In theory, the data from the accelerometers and gyroscopes can be integrated together to obtain the attitude angles and velocities.

Define the velocity vector of the central of the mass in the body frame as $V = [U\ V\ W]^T$, the Euler angles as $Q = [\phi\ \theta\ \psi]^T$ and the angular speed of the drone in the body frame as $\Omega = [p\ q\ r]^T$. The governing equation will then be obtained as:

$$\dot{V} = \Omega \times V + F \tag{2.2}$$

$$\dot{Q} = G(\Omega, Q) \tag{2.3}$$

where,

$$G(\Omega, Q) = \begin{bmatrix} p + (q\sin\phi + r\cos\phi)\tan\theta \\ q\cos\phi - r\sin\phi \\ (q\sin\phi + r\cos\phi)\sec\phi \end{bmatrix} \tag{2.4}$$

This equation, however, only described an ideal case. While in a real situation, there is always noise in the measurement of the accelerometer and the gyroscope. Meanwhile, the accelerometer is influenced by the gravity and the gyroscope may also suffer from drifts and becomes biased. To address this problem, the information from the vision system is used as an combination to fuse with the inertial sensors [5]. There are various algorithms which can be applied based on this idea, e.g., the KF (Kalman Filter), the EKF (extended Kalman Filter) or the complementary filter.

C. Control Loops

Figure 2.3 shows the internal control loop of the AR.Drone [5]. The Autopilot has a three-level nested structure. The inner loop controls the angular rate of the quadcopter. The error between the reference and the current angular rate measured from the gyrometers is calculated and fed to the mix PWM block to

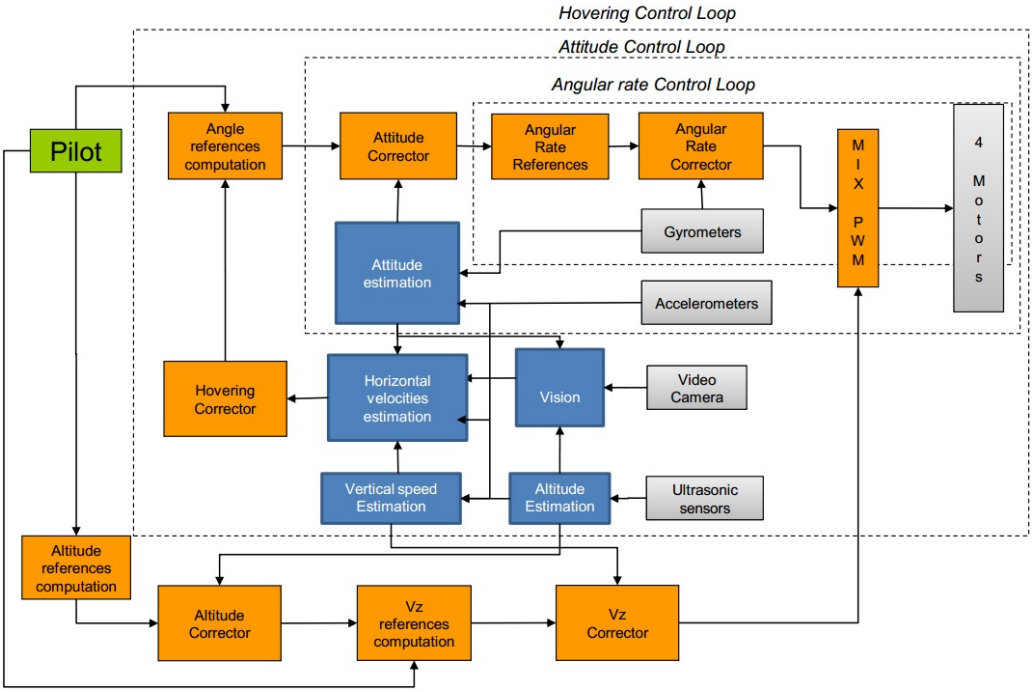


Fig. 2.3: Data fusion and control structure inside the AR.Drone quadcopter [5]

driver the motors. The attitude loop uses the IMU data from accelerometers and gyrometers to estimate its current pose in 6 DOF and applies a corrector to the angle reference. The outer loop calculates the measurement from the ultrasonic to provide an altitude estimation of the drone. Visual features are fused with the aforementioned pose estimation from the IMU to estimate the horizontal velocities. When in the hovering mode, the control system stabilizes the drone at the desired altitude and regulates the velocities to zero.

2.5 Visual Servoing

Visual servoing (VS) is referred to the technique that uses the information extracted from a visual sensor to control the robot motion in a closed-loop way. In most cases a RGB camera is used as the vision sensor and guides the aircraft with the extracted visual features. Typical applications in this category include following a line/path [31] [28] , taking-off and landing [37] [2] [16] and object tracking [39] [18] [29].

2.5.1 Path Following

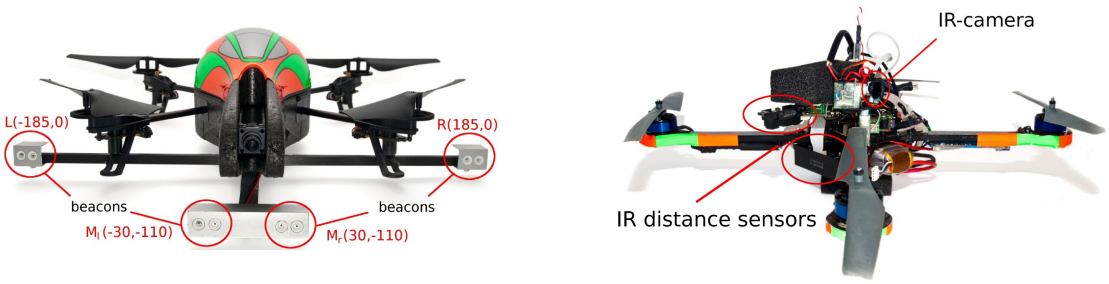
Path following is a typical application of visual servoing. In the project KingLion [31], a co-axial helicopter system with a webcam module was designed to follow the coloured tracks on the ground. An adaptive colour edge detection algorithm is used to segment the image and a colour classifier is applied to differ red, green and blue. Once the direction of the line is decided, control can be implied to steer the heading of the UAV. This work proves that the usage of coloured track is a simple but effective way for UAV visual navigation. However, this navigation method can only applied in a known environment and the UAV can only navigate through the pre-defined path. Trung Nguyen et al. further improved this by using visual-based path-following method based on Funnel Lane theory in [24].

2.5.2 Autonomous Landing

Autonomous taking-off and landing is another classic application of visual servoing for VTOL platforms. An autonomous UAV with monocular vision is designed to land on a kayak in [37]. MATLAB was used in this study for image processing and an adaptive PID controller which shows a strong robustness in the presence of wind was designed. Another work can be found in [2], in which a quadcopter system was designed to hover and landing on a visual pattern. In the work of [16], the scenario is quite similar to the proposed one in this study where a UAV tracks a UGV and autonomously lands on a circular pattern on the UGV. The proposed system uses adaptive thresholding and Hough Transform techniques which improve the robustness of the visual detection at the cost of more computational power.

2.5.3 Using a Quadrotor to Follow another UAV

In the work of Martin Saska et al.[39], a quadcopter is used to track another quadcopter. Both systems use on-board processing and no wireless communication is used. The leader is a modified AR.Drone while the follower is a Asctec Humming-bird. The AR.Drone is equipped with four pairs of LED patterns for pose estimation. The configuration of the LED beacons is shown in Figure 2.4a. This setup allows for an offset angle of about 45° with a distance up to $3m$.



(a) (b)

Fig. 2.4: **(a)** The leader AR.Drone with four pairs of LEDs. These eight infra-red LEDs provide four blobs for the follower to track. **(b)** The follower Humming-bird with two infra-red distance sensors (One Sharp GP2D12 for short range and one Sharp GP2Y3A003 for longer range) to measure the height from the ground and a special IR-camera to detect the LED pattern on the leader [39].

The follower has a IR-camera and two infrared sensors for height measurement (Fig. 2.4b). An additional 8-bit microcontroller is used to obtain blobs' positions from the Wii Remote camera via I^2C and calculate the relative pose of the leader.

The major issue involved in this research is the solving of PnP (*Perspective-n-Point*) problem. The PnP problem is defined as calculating the relative pose from a camera to a set of n markers (the position relationship of these markers is known). A unique algorithm is designed in this study which is 20 times faster by giving several constraints. This algorithm can be run with a rate of 40 Hz on the ATMega microcontroller and 60 Hz on a Gumstix processor.

Another work was done in [18]. Instead of using infra-red LEDs and low-cost sensors, this system uses passive makers, three orange table tennis balls, for the follower to trace. Again, by solving the P3P problem, the follower is able to estimate the position of the leader and maintain a desired distance between them.

2.5.4 Railway Track Following with the AR. Drone

The study [28] shows the usability of a quadcopter in a civilian application to follow a railway track with the extracted visual information. Image processing is applied to detect the vanishing point (VP) in the image and then Kalman Filter is used to track the VP over consequent frames. The behaviour of the vanishing

point is described as a constant velocity model:

$$\begin{cases} x_k = x_{k-1} + v_{k-1} \cdot \delta_k + \omega_{k-1}^x \\ v_k = v_{k-1} + \omega_{k-1}^v \end{cases} \tag{2.5}$$

where x_k is the current x position in pixels of the VP, v_k is the velocity of x_k, ω_k presents the process noise and finally δ_k is the sampling interval between two samples. The control objective of this system is to make the horizontal distance between the image center and the VP maintained at zero. The output of the estimated VP is used to control the yaw angular velocity of the drone while the forward speed is kept constant.

2.5.5 General Object Tracking with a UAV

There is also a interesting study that implies a system in which a UAV is able to track a larger number of general objects in sub-urban areas [29]. The system diagram can be seen from Fig.2.5. This system has two main modules: an

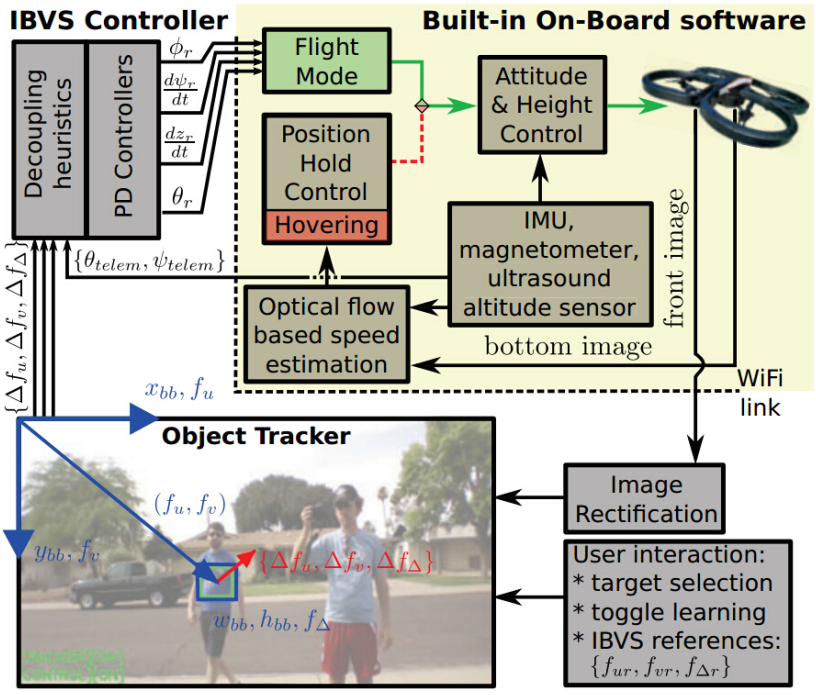


Fig. 2.5: An architecture for general object tracking using OpenTLB library and IBVS controller. Introduced by [29].

object tracker and an IBVS (Image Based Visual Servoing) controller. The open source OpenTLB library is used as the object tracker and the system is tested

for tracking different stationary objects as well as a moving person in sub-urban areas. The disadvantage of this architecture is that it can not estimate the depth of the image, so the size of the object of interested should be known. This can be solved by using stereo vision instead of a monocular camera but may also involve huge amount of computations.

2.6 Summary

The aforementioned works give a solid foundation of this study and some of the idea are further improved in this project. The modelling of quadcopter is important since it is involved in the design of control systems. It is also described that different platform could be selected, either a new platform or a modified one, based on the certain application. The popularity of using AR.Drone in research is also highlighted for its low cost and abundant on-board sensors. Finally, different tasks based on vision have been mentioned and these works have prove the feasibility of using quadcopter platforms to execute visual servoing tasks. There are other topics in the literature, such as SLAM, autonomous navigation, path planning or even distributed flight array [27]. Since these topics just have a little connection with this work, they are not going to be discussed here.

Chapter 3

System Overview

In this system, an AR.Drone quadcopter is designed to locate and follow a UGV with its integrated bottom camera. It is assumed in this study that no position information is provided by the UGV and no measurement sensors (infra-red, LIDAR, ultrasonic) are available on the quadcopter. In this situation, the drone has to locate and track it only with visual features. Figure 3.1 shows a simulated scenario where the drone is tracking the UGV. The motion of the AR.Drone is controlled in a centralized way where a ground station is used for image processing and PID controls.

This chapter gives a brief view of the system framework and presents the hardware and software platforms involved in developing of this project. The AR.Drone 2.0 is described as the VTOL platform to perform visual servoing with its integrated camera. The description of the UGV platform is followed, giving a introduction of the Nexusrobot 4WD mobile robot platform. The Robot Operating System (ROS), which provides a basic software environment, is also introduced in a separate section. Finally, the features of the OpenCV image library is given as the last part of the chapter.

3.1 System Framework

Figure 3.2 shows the overall configuration of this system. The control of the drone is achieved in a centralized way. Firstly, the drone transmits the raw image obtained from the camera to the ground station. Then the ground station, which is

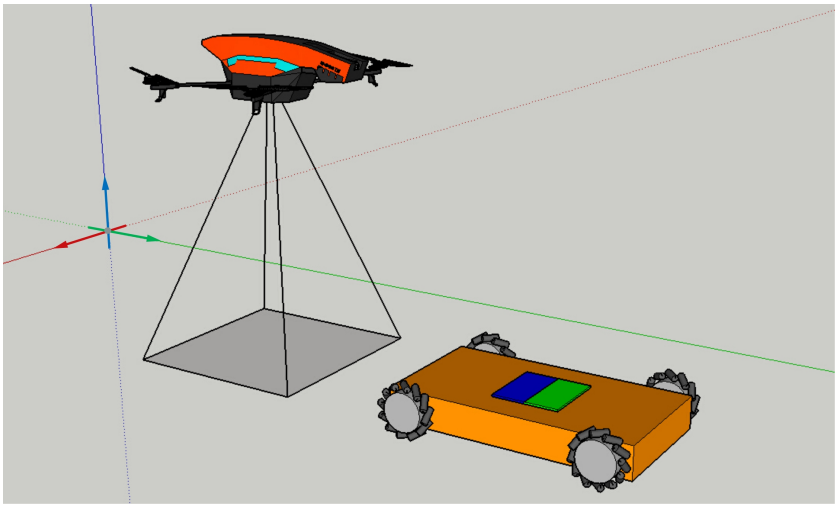


Fig. 3.1: A 3D model demonstrating the scenario in which an AR.Drone is chasing a UGV. The UGV is equipped with a coloured marker and the drone searches the ground for this marker with its bottom camera.

running Linux and ROS, analyses the image and extracts the features to evaluate the position of the UGV. Finally, high-level commands are calculated and send back to the drone as the reference of the internal controllers. The UGV communicates with the ground station using a ZigBee module to receive speed commands from the human operator. A joystick is also used to manually control both the UAV and the UGV in a remotely way.

3.2 AR.Drone 2.0

Parrot's AR.Drone (Figure 3.3) is initially defined as a high-tech quadrotor platform for augmented reality games. It was released in 2012 and is a successor of the AR.Drone 1.0. Compared to its previous version, the new AR.Drone 2.0 has two cameras with higher resolutions and a DSP co-processor for achieving on-board image processing and estimating its ground speed. The electronic and the software interface of the AR.Drone 2.0 will also be described in this section.

3.2.1 Main Features

The AR.Drone was designed to be able to fly both indoor and outdoor. It has a total weight of $380g$ ($420g$ with the outdoor hull) and a size of about $55cm$.

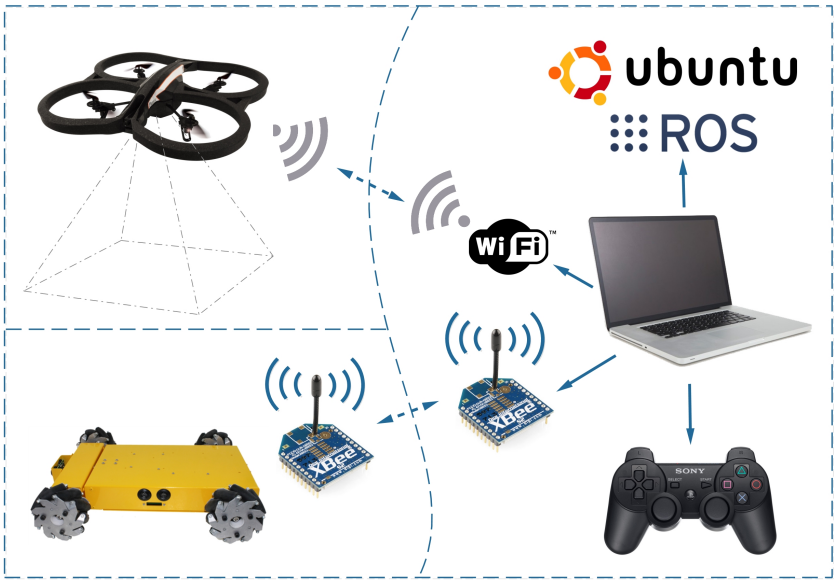


Fig. 3.2: The overall framework of the designed system. Three major parts are included in this system: the AR.Drone, the UGV and the ground station. The AR.Drone is connected with the PC with WiFi while the UGV uses Zigbee to communicate with the ground system.

Fig. 3.3: The AR.Drone 2.0 quadcopter developed by Parrot SA, France

The main frame is made of polypropylene and the crossed arm is made of carbon fiber tubes. The main processor of the AR.Drone is a 1 GHz 32 bit ARM Cortex A8 and the RAM is 1 Gbit running at 200 MHz. The new version AR.Drone 2.0 also has a video processing co-processor, a 800 MHz DSP TM320DMC64x, which reduces the computational burden of the main processor. A light-weight Linux 2.6.32 distribution is running on the AR.Drone to schedule different tasks.

It also has a WiFi 802.11 b/g/n module, a USB high speed 2.0 for extensions and a control interface which allows the drone to be controlled by a phone, a tablet or a Linux computer.

3.2.2 Sensors

AR Drone 2.0 is equipped with abundant sensors on-board. These sensors vary from inertial measurement unit, ultrasonic, barometer sensor to video cameras. It has a 9 degree-of-freedom (DOF) inertial measurement unit composed of a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. The precision of the accelerometer is $+/-50mg$ and the maximum supported angular speed of the gyroscope is $2000°/second$. The measurement from the magnetometer is served as a fusion data for state-estimation as well as providing the orientation to the north. The drone estimates its height and vertical speed from the barometer and the ultrasonic altimeter. When the height is higher than $5m$, the barometer will be selected as the source of altitude estimation. In other case, the ultrasonic sensor will be used instead. There are also two on-board cameras available for use. The frontal camera has a resolution of 720P at 30 fps and a 93° field of view while the bottom camera is pointed vertically and has a resolution of 320×240 with a view angle of 63°.

3.2.3 Software Interface

The AR.Drone provides a programming interface, namely the AR.Drone SDK (*Software Development Kit*), for developers to achieve secondary development. The role of this SDK is to provide a channel to send control commands, obtain navigation information, which includes IMU raw data, altitude, ground speed estimation etc., and camera streams from the drone.

In this work, the *ardrone-autonomy* is used as the programming interface. *Ardrone-autonomy* is a ROS driver for both AR.Drone 1.0 and AR.Drone 2.0. It is a wrapper of the AR.Drone SDK thus has a similar functionality. This driver exchanges data with the AR.Drone hardware and runs as a normal node in the ROS environment. It publishes navigation data and video streams to other nodes and also receives control commands from others.

3.3 Nexusrobot 4WD Mobile Robot

A four-wheel-drive autonomous ground vehicle developed by Nexusrobot is used as the tracking target of the AR.Drone (Figure 3.4). This platform has a dimension of $400mm \times 360mm \times 100mm$ with a total weight of $4.2kg$. The use of four Mecanum aluminium wheels allows the robot to move in all directions and the suspension system ensures the wheels are always adhered to the ground.

Fig. 3.4: The front view of the 4WD Mecanum Wheel Mobile Robot platform from Nexusrobot. This platform has four 100mm mecanum wheels with encoders, four ultrasonic sensors and an Arduino 328 controller with IO expansion board.

The mobile robot uses DC brushed motors and each motor has a separate encoder with a resolution of 12 CPR (counts per round). Motors are driven by two on-board motor driver ICs and are powered by a 12V NiMh battery. The behaviour of the vehicle is controlled by an Arduino microcontroller. The output of the encoders are also connected to the microcontroller for speed feedback. Fig. 3.5 shows the detail of internal hardware of this robot platform.

3.4 The Robot Operating System

The *Robot Operating System* (ROS) is an open source robot framework designed to be used in robotic research and robotics industry. ROS was originally developed by the Stanford Artificial Intelligence Laboratory (SAIL) with the support of the

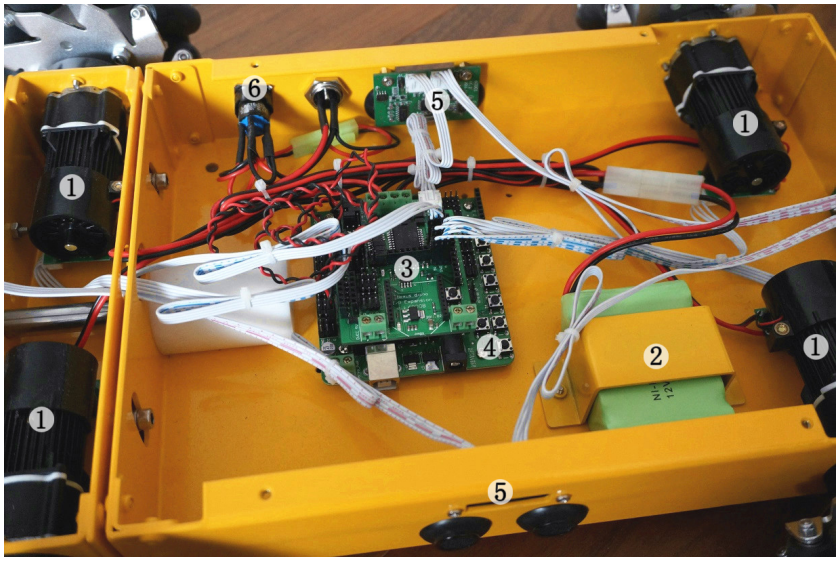


Fig. 3.5: The Hardware of the Nexusrobot 4WD Mobile Robot. Components marked in the figure are: **(1)** Four brushed DC motors, **(2)** NI-MH battery, **(3)** Expansion board, **(4)** Main control board, **(5)** Ultrasonic sensors and **(6)** Power switch.

Stanford AI Robot project in 2007 and is now developing at Willow Garage, a robotics research institute, collaborated with more than 20 institutions. ROS is released under the business friendly BSD (*Berkeley Software Distribution*) license and it is free for commercial and research use. In this study, the version ROS Hydro is used and is running upon Ubuntu 12.04 LTS.

Some basic concepts in ROS are Master, Node, Topic, Message and Service, all of which provide different information to the ROS graphic level. Here gives a list of their definitions and functions:

- **Master**: The master is the core of a ROS program. It provides naming and registration services for topics to communicate and exchange data. It also tracks publishers / subscribers and services.

- **Nodes**: Nodes can be seen as a basic execution program running in the ROS. Nodes can communicate by topics or services and by doing this way, different nodes with single functionality can be managed together to build a more complex robot system.

- **Topics**: Topic is like a communication channel for nodes to exchange data.

A node can publish (or subscribe) a topic to send (or receive) data from other nodes. The topic can be transmitted by using either TCP/IP or UDP/IP.

- **Messages**: Messages are containers of data which can be send by topics. There are many different types of built-in messages available and the programmer can also define his (her) own message(s).

- **Services**: Service is another way to send information to other topics. Unlike a topic, an answer will be given as soon as the service received a request from another node. Hence, Services can be seen as a way for synchronous communication.

The techniques in ROS are enormous and complex. This section is just a small fraction of all of them, but it is already enough for the reader to understand the rest of this study. To learn more about ROS, please refer to the book written by Aaron Martinez et al. [20].

3.5 OpenCV

OpenCV (Open Source Computer Vision) is an image processing library which was designed for real-time vision tasks such as product inspection, security monitoring, interactive art and advanced robotics [26]. It is also released under the BSD license and it is free for commercial use. A OpenCV application can be programmed with C/C++ [4], Java and Python [15] and it supports different platform including Windows, Linux, Android and Mac OS.

The latest version OpenCV 2.4.9 is used in this study and just before the release of this work, the version 3.0 beta came out with some new GPU acceleration features that may boost many vision algorithms. However for the time limitation, the OpenCV 3.0 is not used in this work but will be considered in the future.

Summary

This chapter discussed the system framework which is consisted of a UAV, a UGV and a ground station. The features of the AR.Drone quadcopter is introduced. It is a low cost platform but its abundant on-board sensors and well-developed firmware make it an ideal platform for research and thus is selected as the quadcopter of this work. The UGV platform is a 4WD platform which has an 8-bit AVR microcontroller as the main processor and it can communicate with the ground station via ZigBee. The ground station is running a Linux distribution and uses ROS and OpenCV to control the UAV in a centralized way. ROS is widely used in robotic research and OpenCV is also a famous library for real-time image processing, thus the system framework and algorithms used in this work can be easily referenced by others. This chapter is a brief introduction of the used hardware platforms and software tools and the system implementation detail will be discussed in later chapters.

Chapter 4

Vision-Based Target Detection

Visual sensors can provide much more environmental information than other types of sensors and they can be easily accessed these days. However, computer vision is still a challenging work to do [12] [7], especially in situations where time constrains are required. This work involves a vision algorithm for an autonomous aerial vehicle to locate a passive artificial marker. Similar to industry robot manipulator systems, this system is an eye-in-hand application in which the camera is mounted at the end of the rigid body and will share its dynamics. The movement and attitude of the quadcopter have an impact on the camera, thus elaborate filtering should be used to remove the movement noise and specific transformation should be applied to cancel the effect of tilting (pitch, roll and yaw).

The algorithms for marker detection introduced so far include model based methods (or geometric methods), probabilistic methods and non-model based methods. Model based methods are these which depend on the shape features of the object such as squares, circles, rectangles or a combination of them. A typical and widely used algorithm in this category is the Probabilistic Hough Transform. On the other hand, non-model based methods does not rely on the structure features of the object but only uses colour features. The stability and robustness of non-model method is not as good as model based method. However, non-model approach is much faster and requires relative low computational power. In order to achieve a real-time autonomous system, a non-model based method, *Blob Detection*, is selected as the algorithm of this work.

There are roughly five stages involved in the process of detecting the marker,

which are **1)** Visual Perception, **2)** Image Filtering, **3)** Image Segmentation, **4)** Blob Detection and **5)** Object Coordinates Calculation. Details of these steps will be discussed individually in this chapter.

4.1 Visual Perception

The input of the visual system is the image obtained from the bottom camera of the AR.Drone. The image from the camera was scaled into 640×360 by the internal firmware before transmitted to the ground station. The Pinhole Camera Model is adapted as the camera model. Two important parameters in this model are the Intrinsic Matrix and the Extrinsic Matrix, which are used together to project a point from pixel frame to world frame. The Intrinsic Matrix is defined as

$$\boldsymbol{M_I} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where α_x, α_y represent the focal length, γ is the skew coefficient which is normally zero and (u_0,v_0) is the principal point which should be ideally equalled to the image center. This matrix, as well as the Extrinsic Matrix, depends on the parameters of a certain camera and needs to be identified individually.

- ***Camera Calibration***

The characteristic of camera varies and each camera has different level of distortion. Thus in the early stage of visual detection, the camera should be calibrated by a procedure known as *Geometric Camera Calibration*. This process is used to evaluate the two aforementioned parameters. Calibrating a camera in ROS is straightforward and there is a build-in tool *camera_calibration* which can be directly used. Figure 4.1 shows a screen shot of this procedure with a 8×6 chess board pattern.

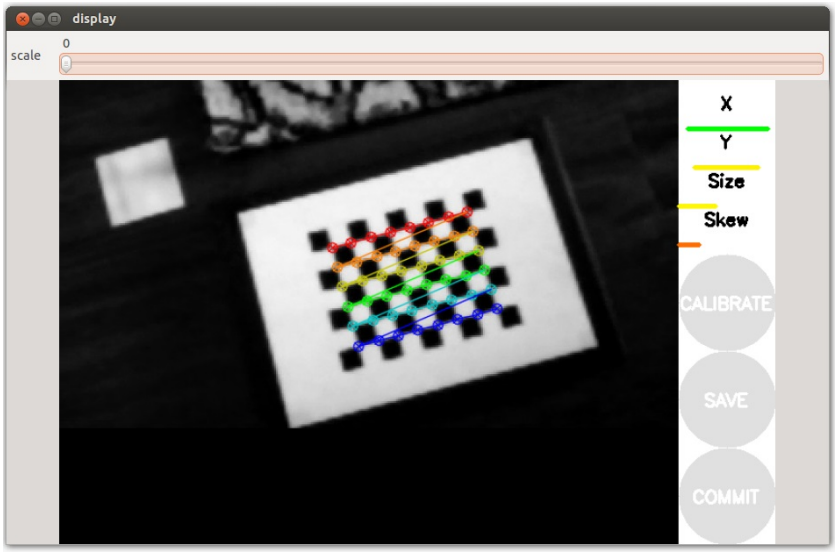


Fig. 4.1: Calibrating the Bottom Camera of the AR.Drone. The chessboard pattern has 8×6 inner corners and each square has an area of $0.04m$. The calibration tool used is a built-in package *camera_calibartion* in ROS.

- ***Passive Marker Pattern***

In order to simplify the process of image processing, a passive artificial marker is designed and used. A coloured marker with two squares (Figure 4.2b) is selected from four candidates and is mounted on the top surface of the UGV. There are a few reasons to select this pattern. Firstly, the two sections in this pattern are equally sized and has the same shape, which means the same method and parameters can be adapted for both parts. Secondly, the orientation of this pattern can be easily obtained by calculating the angle between the central points of two sections. The final choice between **(a)** and **(b)** is simply compared by experiments.

4.2 Image Filtering and Segmentation

Image filtering deals with eliminating noise from the image. Before applying any algorithm, it is required that the image should be filtered to remove any possible noise, e.g. the vibration noise caused by the movement of the AR.Drone. The OpenCV function *medianBlur()* is therefore used.

The segmentation is then done by a sequence of algorithms. Firstly, the RGB image is transformed into HSV (*Hue, Saturation and Value*) space and then thresh-

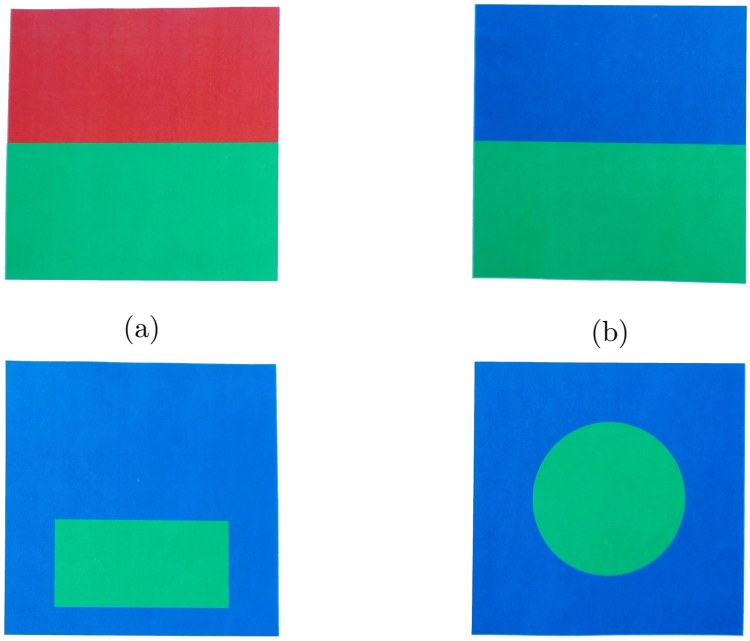


Fig. 4.2: Coloured Pattern Candidates for the Passive Marker. **(a)** Two Squares (red and green); **(b)** Two Squares (blue and green); **(c)** Two Nested Squares; **(d)** A Circle inside a square

olding is applied to transform the HSV image into a binary one. As the last step in this stage, eroding and dilation are done to remove separated pixels.

4.2.1 Colour Thresholding

The RGB image is transformed into HSV space through a non-linear transformation. The RGB space is a cubic space while the HSV space is a cone like space. The motivation for doing this step is that the colour in HSV space is organised in a more natural way and can be easier threshold than that in RGB space. Normally, the range of Hue is $[0, 360]$ and the range of Saturation and Value is $[0, 100]$. However in OpenCV, the range of HSV is defined differently as: $H = [0, 180]$, $S = [0, 255]$, $V = [0, 255]$. There is a *cvtColor()* function in OpenCV for handling the HSV transformation by setting the third parameter as CV_BGR2HSV.

The converted image is then applied a colour thresholding procedure to find interested regions in a certain colour range. The range of the colour was tested by several experiments and test flights to find an optimal one. This range is

crucial to the vision algorithm and should be selected properly. An aggressive value may improve the success rate of the detection but also may introduce false measurements.

4.2.2 Eroding and Dilating

Erosion and *Dilation* are two basic Morphological transformations which can be used in pairs to remove separate regions of a binary image. The use of eroding followed by dilating is also known as *Opening*. A 5×5 Morph Ellipse is used as the kernel, which takes the form of

$$\boldsymbol{K} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

To further eliminate unwanted areas, one more stage of Opening is done. After this step, this image will be processed into a binary image with just a few separated segmentations. A demonstration of the whole segmentation process is given in Figure 4.3. It can be concluded from the diagram that the colour threshold method is effective and the Opening has a satisfactory performance for removing noises caused by thresholding.

4.3 Blob Detection

The aim of blob detection is to find one or some contour(s) in the segmented image which satisfies a certain criteria. The rule applied in this application is to firstly find the maximum contour and then evaluate if its area S satisfies the criteria $S\_min < S < S\_max$.

After applied the Opening operation to the threshold images, there may still have a number of separated pixel blocks in the binary images. The Blob Detection algorithm will examine these so called *contours* iteratively to find the maximum one as the candidate which has the maximal possibility to be the real object. The

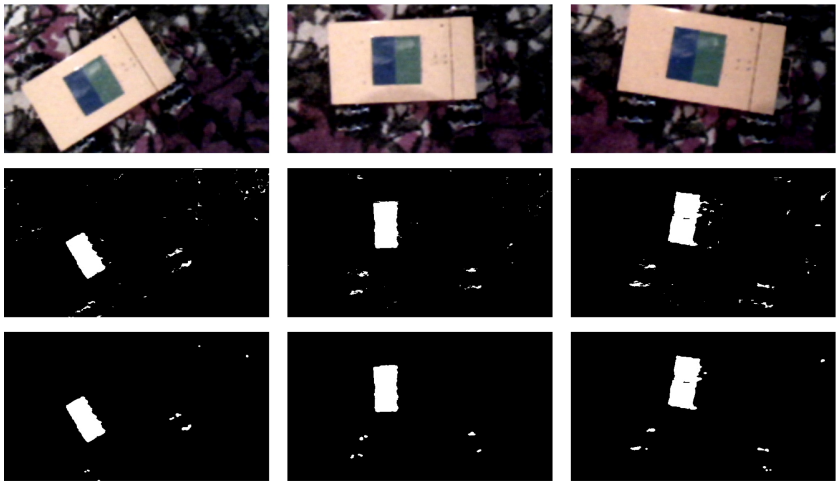


Fig. 4.3: Image Segmentation Result. These three images were viewed from different perspectives and under varying lighting conditions. The images in the second row are thresholded in HSV space and the images in the last row are the result after applied the Opening operation.

next step is to calculate the $Moment$ of this contour from the formula:

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y) \tag{4.1}$$

where $I(x, y)$ is the intensive of the gray image at point (x, y). The zeroth order moment $\{M_{00}\}$ represents the area of the contour while the first order $\{M_{01}\}$ and $\{M_{10}\}$ is the centre of mass of the image in x and y directions. The program will check the area criteria to see if this contour is satisfied with the range limitation. If so, the central point can be obtained as:

$$\bar{x} = \frac{M_{10}}{M_{00}}, \bar{y} = \frac{M_{01}}{M_{00}} \tag{4.2}$$

The Figure 4.4 shows the result after applied this algorithm for both the blue and green part of the featured marker. The program outlined the selected contours from which it could see that the algorithm proposed has a high accuracy even for images in a relative complex environment (with the carpet as the background).

4.4 Target Position Estimation

The final step of visual detection is to determine the position of the object of interest based on the result from visual algorithms. Assume point $P_1 = (x_1, y_1)$

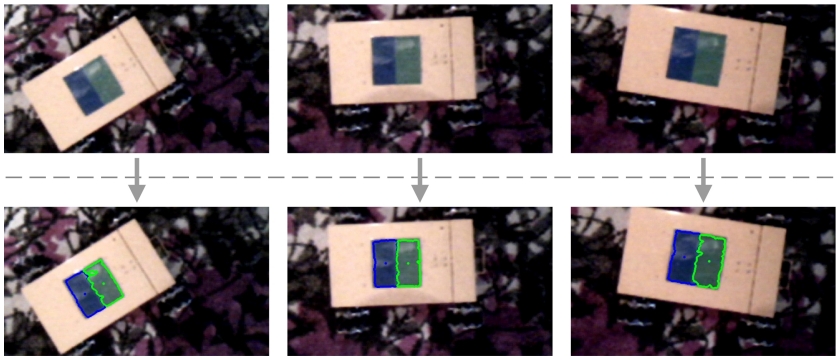


Fig. 4.4: Blob Detection Result. The coloured outlines represent the detected contours and the circular points in the middle of each area are the estimated central points calculated by the Moment of that contour.

is the detected center of the blue square and $P_2 = (x_2, y_2)$ is the central point of the green part. Since these two squares are aligned and have the same size, the estimated point $\hat{P_d}$ can be calculated as the central point of P_1 and P_2. The coordinate $(\hat{x_d}, \hat{y_d})$ as well as the direction $\hat{\theta_d}$ of the featured marker in the pixel frame can be then estimated as:

$$\begin{cases} \hat{x_d} = (x_1 + x_2)/2 \\ \hat{y_d} = (y_1 + y_2)/2 \\ \hat{\theta_d} = artan((y_2 - y_1)/(x_2 - x_1)) \end{cases} \tag{4.3}$$

For fault tolerance, the program will also check the line distance $L(P_1, P_2)$ to see if it is a reasonable value. The pixel line distance L is defined as:

$$L_{(P_1,P_2)} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.4}$$

The computed position and direction will then be published as a ROS Message of the type *geometry_msgs/Pose* as the reference for the position controller.

Summary

This chapter is mainly focused on the visual algorithm of this system. The camera calibration issue and the selection of the coloured marker are first described. The detail of segmentation and Blob detection are then followed. In this algorithm, the raw image is first transformed into HSV space before applied a colour thresholding procedure. Erosion and Dilation are then used to remove separated pixel blocks. Finally, the moments of each contour in the binary image are calculated and the position of the most possible candidate is calculated. This position will be transformed into a 3-D relative position to be used as the reference for the position control. Details of this process will be described in the next chapter.

Chapter 5

Real-Time Visual Servoing

Visual Servoing (VS), as mentioned before, is referred to the use of visual feedback to control the locomotion of a robot. Once the relative position of the ground vehicle to the UAV is solved by the coordinate transformation, a motion controller can be implied to control the position of the quadcopter for tracking the vehicle in real time. This chapter is focused on the control aspects and gives the design details of the IBVS (Image-Based Visual Servoing) controller used in this project.

5.1 System Dynamics

In the controller design phase, the model and parameters of the AR.Drone is unknown and thus is treated as a black box (Fig. 5.1). This model incorporates the dynamics of the on-board autopilot and it is a non-linear MIMO system, which can be represented as:

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ y(t) = Cx(t) \end{cases} \tag{5.1}$$

The system state $x(t) := (z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi})^T$ reveals the internal states of the quadcopter and most of them can be directly measured by the inertial measurement unit. Other states, such as $(\dot{x}, \dot{y})$, can be estimated from the known states fusing with other measurements, e.g. the magnetometer and the bottom camera. However, not all of these states are important in the IBVS system. Define the state of interest as $X(t) := (z, \psi, \phi, \theta)^T$ and the input as $u(t) := (u_{\dot{z}}, u_{\dot{\psi}}, u_{\phi}, u_{\theta})^T$ which is scaled into $[-1.0, 1.0]$, a simplified and linearized

model can be obtained as:

$$\begin{cases} \dot{X}(t) = AX(t) + Bu(t) \\ Y(t) = CX(t) \end{cases} \tag{5.2}$$

where A is the system matrix, B is the control matrix and C is the output matrix which is diagonal.

Furthermore, it is known that the pitch θ and roll ϕ angles are related to the ground speed $\dot{x}$ and $\dot{y}$ and if keep these angles significantly small, the system can be decoupled as four individual SISO systems: $G_{\dot{x}}(s)$ as u_θ for input, $G_{\dot{y}}(s)$ as u_ϕ for input, $G_z(s)$ as $u_{\dot{z}}$ for input and finally $G_\psi(s)$ with $u_{\dot{\psi}}$ as the input. It is important to realize that this decoupled model is just an approximation of the real dynamics in order to simplifier the controller design. If the pitch and roll angles are not neglectable, the coupling effects will lead to a unstable behaviour.

To identify the $G_{\dot{x}}(s)$, $G_{\dot{y}}(s)$, $G_z(s)$ and $G_\psi(s)$, system identification methods could be used. However in this work, this is not possible due to the lack of precise equipments for state measurements and a relative large test ground to accept the unpredictable drift of the drone during the identification process.

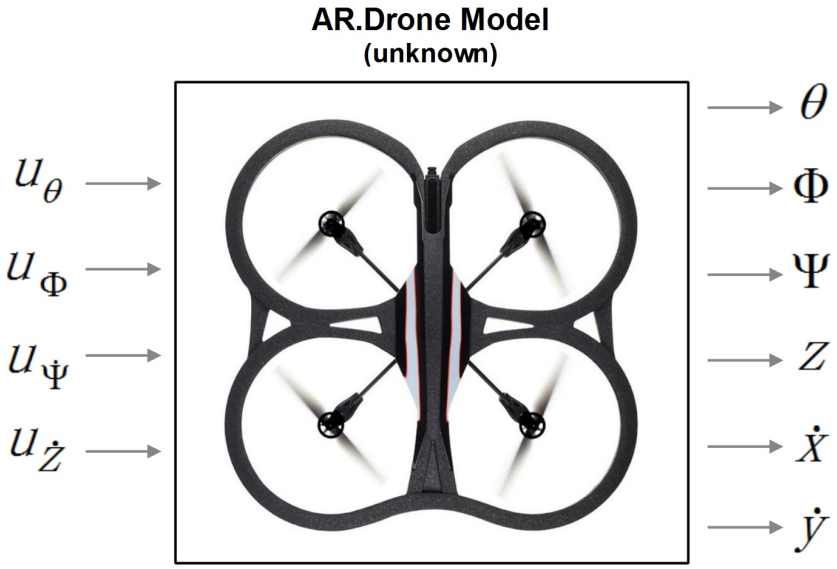


Fig. 5.1: The input and output model of the AR.Drone system. The internal dynamics of the AR.Drone is unknown, thus it is treated as a black box in the stage of controller design. The range of the inputs are scaled into $[-1.0, 1.0]$ and all the outputs can be obtained directly from the AR.Drone SDK.

5.2 Pose Estimation

In order to achieve the visual servoing, the current pose (x, y, z, ψ) of the drone should be estimated. A good estimating of the current position will lead to a great performance improvement. Since only one of the cameras on the AR.Drone can be used at a certain time and the bottom one has already be used for target searching, visual based navigation method, e.g. V-SLAM and PTAM, cannot be implied. Ground truths sensors used for outdoor flying such as the GPS are also not available for indoor flights. Thus, the quadcopter have to estimate its current 3-D position purely based on its on-board sensors. The altitude z and yaw angle ψ can be directly obtained from the drone, but the x and y offsets should be estimated from $\dot{x}$, $\dot{y}$, and ψ. The best way to estimate current states is to use Kalman Filter or Extended Kalman Filter. However in this case the x and y velocities have already been filtered by a complementary algorithm on the AR.Drone before transmitted [17], thus a simple Dead Reckoning algorithm could be precise enough to obtain the current position in x and y axes. The formula of Dead Reckoning used in this work is described as,

$$\begin{cases} {}^{W}\dot{x}(k) = cos(\psi) \times v_x - sin(\psi) \times v_y \\ {}^{W}\dot{y}(k) = sin(\psi) \times v_x + cos(\psi) \times v_y \end{cases} \tag{5.3}$$

and

$$\begin{cases} {}^{W}x_{k+1} = {}^{W}x_k + {}^{W}\dot{x}(k) \times \delta_k \\ {}^{W}y_{k+1} = {}^{W}y_k + {}^{W}\dot{y}(k) \times \delta_k \end{cases} \tag{5.4}$$

The Equation 5.3 transforms the x and y velocities into inertial frame and the Equation 5.4 integrates the velocities in discrete time. It is assumed that the measurement noise of the speed estimation from the AR.Drone is white noise and unbiased and the yaw angle ψ is kept at a small level, so that

$$\sum_{k\to\infty} (v_x(k) + \omega_{v_x}(k)) = \sum_{k\to\infty} v_x(k) + 0 = \sum_{k\to\infty} v_x(k)$$

and

$$\sum_{k\to\infty} (v_y(k) + \omega_{v_y}(k)) = \sum_{k\to\infty} v_y(k) + 0 = \sum_{k\to\infty} v_y(k)$$

Thus in this situation, the obtained estimations of x and y are unbiased.

5.3 Coordinates Transformation

The coordinates computed from the image processing algorithm is in pixel frame and thus depend on the attitude of the quadcopter. Stability problems may rise if feed these coordinates directly as references into the position controller. Imagine that if the object is in front of the quadcopter, the quadcopter will increase its speed in the x axis. But the issue is that the x speed is driven by tilting the θ angle but this will make the object further in the view of the camera. This will result a positive feedback and lead to unstable behaviours. To solve this problem, one way is to transform the coordinates of the object of interest from pixel frame into world frame. This transformation is important but not well documented in other works. The transform formula in this study is based on the one in [17] and made some changes:

$$\begin{bmatrix} {}^{W}x_0 \\ {}^{W}y_0 \end{bmatrix} = \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \left[\begin{pmatrix} k_u(c_v - v) \\ k_v(c_u - u) \end{pmatrix} + \begin{pmatrix} -k_\theta sin\theta \\ k_\phi sin\phi \end{pmatrix} \right] z + \begin{pmatrix} x \\ y \end{pmatrix}$$

where $(x_0,\ y_0)$ is the coordinate in the 3-D world frame, ϕ, θ and ψ are Euler angles, (u, y) is the pixel position in the camera frame and (x, y, z) is the current position of the quadcopter. The other four parameters in this equation stand for the calibrating coefficients. k_u and k_v are related to the field view and resolution of the camera while k_θ and k_ϕ were obtained by real experimental tests.

5.4 Controller Design

The objective of the visual servoing controller is to transfer the system states from the current state $X_c := (x_c, y_c, z_c, \psi_c)$ to the desired one $X_d := (x_d, y_d, z_d, \psi_d)$. As described in the modelling section, the quadcopter can be decomposed into four SISO sub-systems and thus can be controlled individually. A number of candidate methods exist to control the quadcopter. Model-based techniques such as Internal Model Control (IMC) [14] and Model Predicive Control (MPC) could have a satisfactory performance. But the lacking of a valid system model in this work makes model based methods unimplementable. On the contrary, the

classic PID (proportional-integral-derivative) controller is used for its simplicity and effectivity.

5.4.1 Control Structure

Four PID controllers are used in this system, namely the X Controller, Y Controller, Z Controller and the Yaw Controller. The overall control structure is depicted as Figure 5.2. It should be addressed that the inputs and outputs of the system are transmitted via WiFi and a high latency would thus be involved into the system model. Special care should be given when designing the parameters of the controllers in order to avoid any possible system instability.

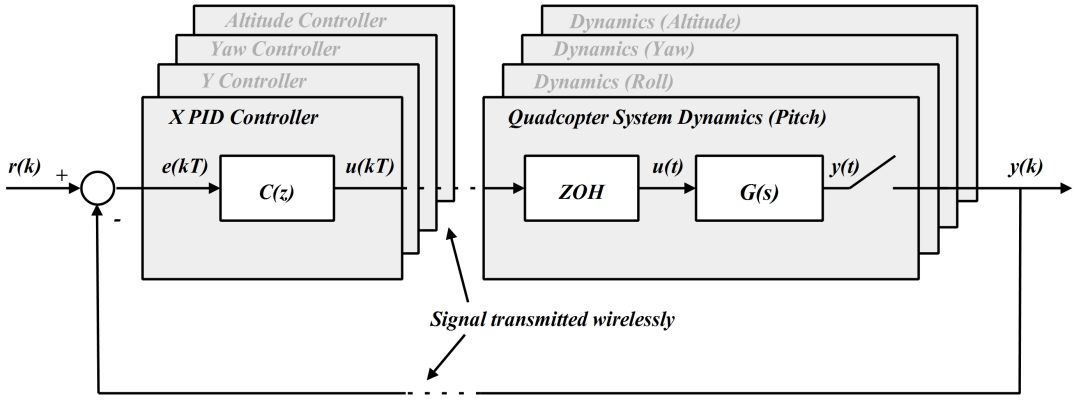


Fig. 5.2: Control System Structure. $C(z)$ is the transfer function of the PID controller in z plane and $G(s)$ stands for the system transfer function in time domain. The dashed line in the graph represents signals transmitted via wireless.

The continuous PID controller takes the form of:

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau) \cdot d\tau + K_d \cdot \dot{e}(t) \tag{5.5}$$

while in this work, the discrete form of PID controller has used:

$$\begin{cases} u_k = e_k \cdot \kappa_p + (e_k - e_{k-1})/\delta_k \cdot \kappa_d + e_k^i \cdot \kappa_i \\ e_k^i = e_{k-1}^i + e_k \cdot \delta_k \end{cases} \tag{5.6}$$

where e_k is the error between the sample k and the reference, κ_p, κ_i, κ_d are controller parameters and δ_k is the sampling time which is calculated every sample. The pseudo code of this discrete PID controller is given as Algorithm 1 in the Appendix.

5.4.2 Controller Parameters

The frequency of the received sensory data is $200Hz$ while the processing rate of the visual detection is only around 20 FPS, thus the control frequency is selected to be $20Hz$. Although the quadcopter has a rapid dynamics, a higher control frequency is not a necessary since the control commands are of high-level and the on-board autopilot will handle the low-level control which needs much higher loop rate. In order to achieve a 3-D position control, four PID controllers are implied:

- **X and Y Controllers:** The X and Y controller structures and parameters are identical for the fact that the pitch and roll dynamics are symmetric. The designed controllers are two PD controllers without a Integral part since there is already a integration in the system. The Derivation gain κ_d is set to be relative high to cancel the effect of transmission delays involved by WiFi communication.
- **Z (Height) Controller:** The height has a slow dynamics and may have steady state error. Thus a PID controller with small κ_i is used for altitude control. The height velocity has an impact to θ and ϕ, thus the κ_p should be selected elaborately to have minimal influence to X and Y controls while keep the desired height.
- **Yaw Angle Controller:** The yaw sub-system has a high response time so a simple Proportional control could already have a good performance.

When a simulator is available, Ziegler-Nichols or the Åström-Hägglund methods can be used for tuning the PID parameters. But in this study there is no valid model of the quadcopter, so the tuning has to be done by trial-and-error. The designed controllers are computational effective while still robust to external disturbances which can be seen from further experiments.

5.5 Summary

This chapter deals with the controller design process. The model of the quadcopter is first introduced followed by the Dead Reckoning algorithm for pose estimation. The transformation formula is given to transform the position from pixel frame into world frame. Finally, a position controller has been designed to achieve 3-D position control with four PID controllers. The control frequency is selected as $20Hz$ and the PID gains are obtained by trail-and-error.

Chapter 6

System Implementation

The vision and control approaches used in this project have already been introduced in Chapter 4 and Chapter 5. This chapter, however, will mainly focus on the system implementation details. The diagram in Figure 6.1 shows the overall software structure of this system. A four-layer software structure is applied in this system. The lower three layers have already been mentioned in Chapter 3 so they will not be discussed here. The top layer, Applications, is a set of programs managed by ROS and is the main concern of this chapter.

This chapter is organised as follows: the node graph and message flow is first discussed. A finite state machine is then introduced for control and decision making of the AR.Drone. A low-cost global vision system was developed to assist the controller design and its features and specifications will be mentioned. Finally, the developed UGV embedded software and the human interface of the ground station will also be discussed.

6.1 Nodes and Data Flow

The ROS system provides an environment for the robot program to be distributed organised. In ROS, different nodes can be ran in parallel and has their own loop rates. Figure 6.2 shows the node graph of this system. It can be seen that there are seven nodes in the ground station and each node has its major task to do. Different nodes are communicated by topics which are managed by ROS. The use of ROS simplifies the data exchanging between processes and makes it possible to

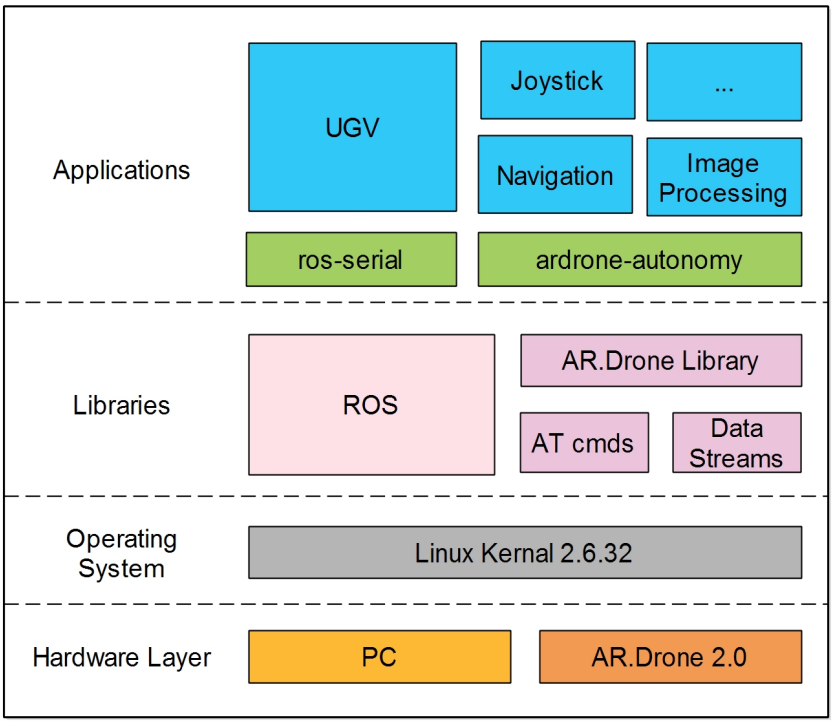


Fig. 6.1: Software Layers of the System

construct a complex system with a number of single-task programs.

The nodes can be roughly divided into three groups based on its role in the system. The first group includes these which handles the visual servoing of the UAV, which are *ardrone_driver*, *dronevt_rtip*, *dronevt_nav* and *dronevt_core*. The *ardrone_driver* node is the aforementioned *ardrone-autonomy* driver linked to the AR.Drone hardware. The *dronevt_rtip* executes the task of real time visual detection of the target of interest and the node *dronevt_nav* handles decision making and position control. The *dronevt_core* is simply a coordinate node while the other two nodes, the *control_interface* and the *xbee_com* are the nodes to handle user input and to teleoperate the UGV, respectively.

6.2 High-level Finite State Machine

The behaviour of the AR.Drone is controlled by a finite state machine (FSM) in the node *dronevt_nav*. There are totally six states in this FSM transmitting between each other whenever needed, which are *Landed*, *Taking-off*, *Manual*, *Tracking*, *Hovering* and *Searching for the Target*. The function of each state is described as follows:

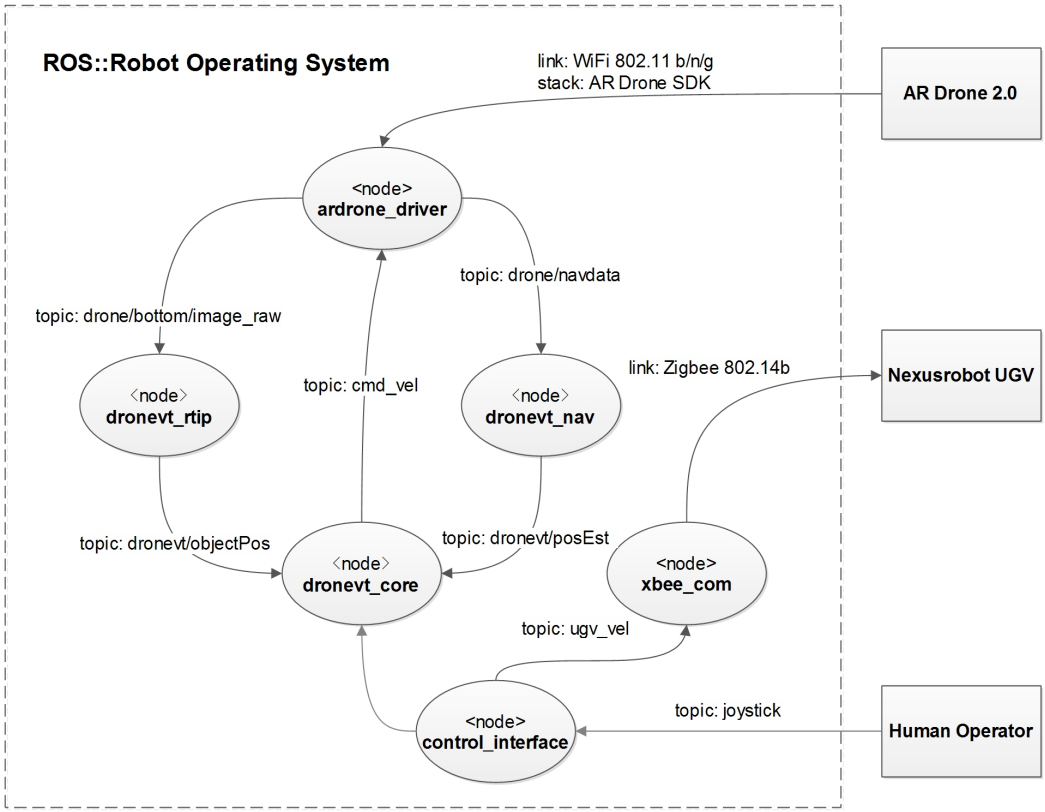


Fig. 6.2: ROS Nodes and Messages Graph. Nodes in ROS are communicated by publishing and subscribing topics. The texts inside the circles are the names of the topics and the text on each connection line indicate the name of that topic.

- **Landed**: In this state, the drone is landed on the ground and no control need to be implied. Every time the drone landed on a flat surface, the *flattrim* process should be called to calibrate the sensors.

- **Taking-off**: After sending the take-off command, the AR.Drone will execute an automatic taking-off procedure. During this period, no control should be imposed and it is recommended that at least $5s$ should be given for the drone to stabilize itself. After this time, the drone will be switched into the Manual mode.

- **Manual**: In this mode, the user can control the movement of the quadcopter and send velocity commands to control its movement with the joystick.

- **Tracking**: The main state in which the quadcopter will search the ground for the featured mark, estimate the relative position of the UGV and try to minimize both x and y distances to the UGV.

- **Hovering**: To minimize unnecessary oscillation around the target location, the system will be switched into hovering mode when the distance error is below a certain threshold ($0.02m$). During the hovering mode, the PID controllers will be switched off and only the internal controller of the drone will work.

- **Searching for the Target**: Under normal operating condition, the drone will hover above the UGV at a height of $1.6m$. When the target was marked lost by the visual system, the drone will gradually increase its height at a rate of $0.1m/s$ to increase its field of view. The maximum height permitted is $z = 2.2m$. The drone will stop rising when hitting the top value and will hold at this height afterwards. However, if the target was lost for more than $10s$, the drone will stop tracking and set itself into standby mode and wait for further commands.

6.3 Human Interface

The *control_interface* is the node providing a channel for the user to interfere the system. For autonomous system, a human interface is an essential to monitor the system status, print out messages or switch off the system when necessary. The human inferface is consists of three parts: log display, processed image output and joystick input. This system has no graphic user interface at the moment, log information and warning messages are directly printed out on the screen and processed images are shown in a separate window. The input device for operating the UAV and the UGV is a SPEEDLINK Play Station 3 wireless joystick. The function designed for each button is shown in Figure 6.3. This external input is designed to manually take off and land the drone, control the movement via the analogue sticks and stop the autonomous flight to avoid an accident or in an emergence of crashing.

Fig. 6.3: The joystick used for teleoperating the UAV. The operator can control 4 DOF of the AR. Drone and send taking off / landing command to the drone. It is also able to switch between autonomous and manual flight mode by pressing the corresponding buttons.

6.4 UGV Software Design

Figure 6.4 shows the diagram of the UGV. In this project, the UGV is tele-operated by a human operator. The high-level control command is send from the ground station to the microcontroller on the UGV with a ZigBee network. The embedded system on the UGV has to resolve this command into speed references for each of the four motors. PID controllers are designed for each motor to control its rotary speed based on the feedback from the speed encoder. Constant acceleration and deceleration are also implied to eliminate the shock involved by any possible sudden speed change.

6.5 Global Reference System

The odometry of the drone based on on-board sensors has a low accuracy and will drift significantly with time. In order to achieve precisely movements such as trajectory following, an external reference system is of great importance. A commercial video capturing system, e.g. the Vicon Motion Capture System [38],

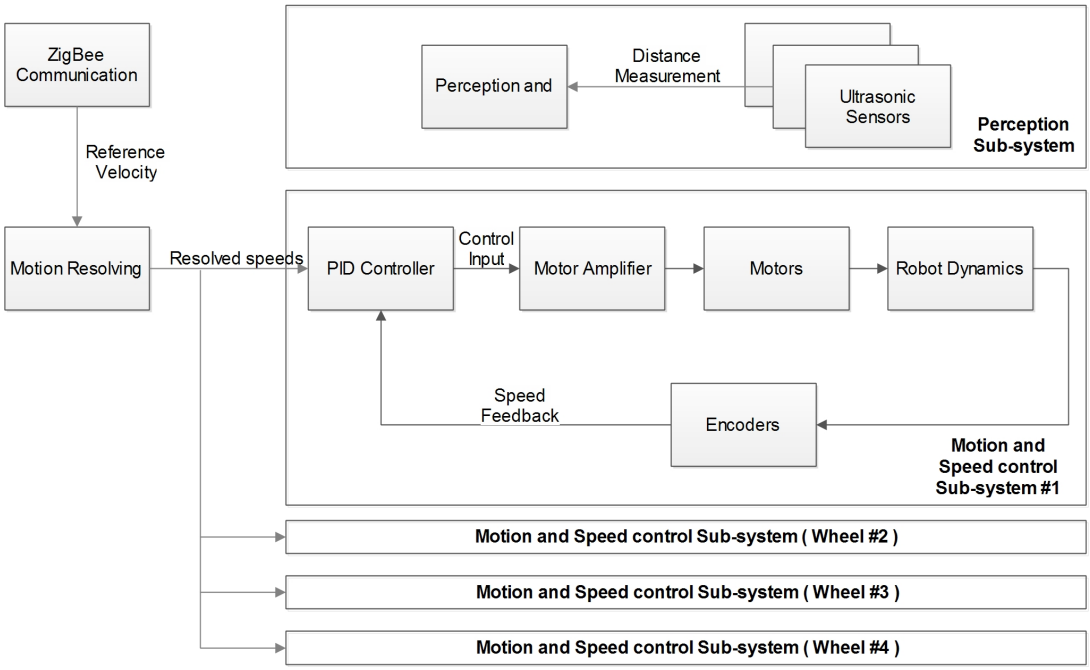
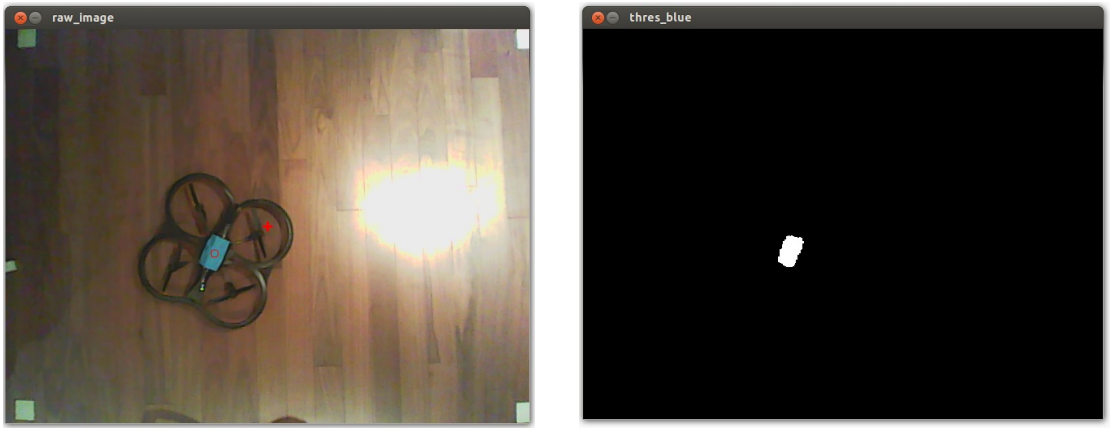


Fig. 6.4: The Functional Block Diagram of the UGV System

is powerful and can be easily applied for global reference. However, such system costs thousands of pounds thus is not always available to researches with limited budget. In such a background, a global vision system using a low cost web camera is designed during the process of this project.

The global reference system uses a *Logitech C170* web camera which only worth £12. The webcam is mounted underneath the ceiling at a height of 2.32 m and can provide a valid working area of $1.5m \times 2.1m$. Four yellow square notes are used to mark the boundary of the working space and no observation can be obtained beyond this region.

It is notable that the frame rate of this system varies according to the lighting condition. In daytime it can perform a real-time image processing as fast as 20 FPS (frame per second), but when the system is under low illumination the performance is only around 8-9 FPS. This is not caused by the vision algorithm but the characteristic of the used web camera. A better camera should be able to handle this issue. As mentioned before, the output from this system is fused with the measurement of the drone to get a better position estimation. However, this system is only employed in the trajectory following problem but failed to be used in the UGV tracking task since the height of the UAV exceeded the view of



(a) View from the Global Camera (b) The Segmentation Result

Fig. 6.5: Global Vision System. The red cross in **(a)** is the central of the image coordination while the red circle is the estimated central position of the AR.Drone. Sub-figure **(b)** shows the binary image after thresholded in the HSV space.

the camera. The algorithm applied in this system is similar to the aforementioned one for detecting the marker on the UGV so it will not be discussed in detail. A demonstration of this system is illustrated in Figure 6.5. Since the estimated result depends on the altitude of drone, the position is calibrated with the height of the drone in real time.

Summary

The system implementation details are given in this chapter. The processing nodes in this system can be grouped into three classes: the nodes for UAV visual perception and position control, the node for handling the UGV movement and the node for human interface. A finite state machine is used to control the behaviour of the quadcopter and handle accidental events. The embedded system on the UGV controls the speed of four motors in closed loop and receives commands from the ground station. Finally, a low cost global reference system is introduced which runs a similar algorithm as the one for the UAV. The software framework is a integration of all the aforementioned visual and control algorithms. Thus the performance and the structure of the software is important in order to satisfy the requirement of a real-time system.

Chapter 7

Experimental Results

To evaluate the control and vision performance of the proposed system, various experiments and test flights have been done. All the experiments were conducted in a cuboid space of $2.0m \times 1.5m \times 2.0m$ with a carpet on the floor to provide textures for the optic flow odometry of the AR.Drone. Three classes of tests are designed to evaluate the system: the first experiment, Position Estimation, tested the IMU-based odometry in 3-D space. This is important since whenever the target was lost, the quadcopter has to estimates its current position to compute the relative orientation of the UGV. The second experiment evaluated the PID controller performance. Hovering at a fixed point as well as figure following are tested in this class. The last class of experiment is to verify the visual tracking of both a stationary target and a vehicle in moving. During all the experiments, the yaw angle reference is fixed to be $0°$ and the height is set to be $0.6m$ for the first two experiments and $1.6m$ for the visual servoing test.

7.1 Position Estimation

To test the precision of the IMU-based estimation, the Global Reference System introduced earlier was used to serve as the ground truth and was compared with the estimated value. During the test, the drone took a manually flight in a random pattern for $120s$. The Figure 7.1 plots the estimated position trajectory as well as the global truth. It can be seen that the estimations from the quadcopter does not fit the ground truth well. There is an average offset of about $0.5m$ in both

x and y directions. To further address the problem, the errors of the estimated and the measured position in x, y axes as well as the relative distances are plotted in Fig. 7.2. It is assumed in earlier chapter that the velocity estimation of the quadcopter is unbiased, however what this diagram shows is that both x and y speeds are suffering constant offsets. The estimation errors are then fitted into three third order functions using MATLAB polynomial curve fitting. The fitted functions are:

$$\begin{cases} f(x_{err}(t)) = 0.0008t^3 - 0.1435t^2 + 17.4777t - 116.6664 \\ f(y_{err}(t)) = -0.0008t^3 + 0.1518t^2 - 1.6310t + 204.0367 \\ f(d_{err}(t)) = 0.0001t^3 + 0.0023t^2 + 10.3049t + 136.7847 \end{cases} \tag{7.1}$$

from which one can see the estimation error has an increasing trending in terms of time. Take the derivative of the first two functions to obtain the approximate error features of x and y speeds, which are given as:

$$\begin{cases} f(\dot{x}_{err}(t)) = 0.0248t^2 - 0.2870t + 17.4777 \\ f(\dot{y}_{err}(t)) = -0.0023t^2 + 0.3036t - 1.6310 \end{cases} \tag{7.2}$$

take the mean of the function values sampled between t = $[0, 120]$ and obtained $\bar{\dot{x}}_{err}$ as 11.55 mm/s and $\bar{\dot{y}}_{err}$ as 5.44 mm/s. This result could be used to eliminate the offset in v_x and v_y in order to achieve a unbiased position estimation. However, there is no evidence that these offsets were involved by other factors, e.g. the textures on the ground, the drift of the yaw angle, etc. Thus no more conclusion can be made before more experiments are conducted.

It also can be seen that, for a short time interval of $10s$ the estimated distance error is relative small (less than $25cm$). Thus the selected $10s$ as the maximum searching time when lost the target is fairly reasonable.

7.2 Autonomous Position Control

The indention of doing the position control experiment is to evaluate the performance of the PID position controllers. Two experiments were done in this test. The first is to set the quadcopter autonomously hover at a fixed point in 3-D space and then imposed disturbance forces. The other is let the quadcopter follow

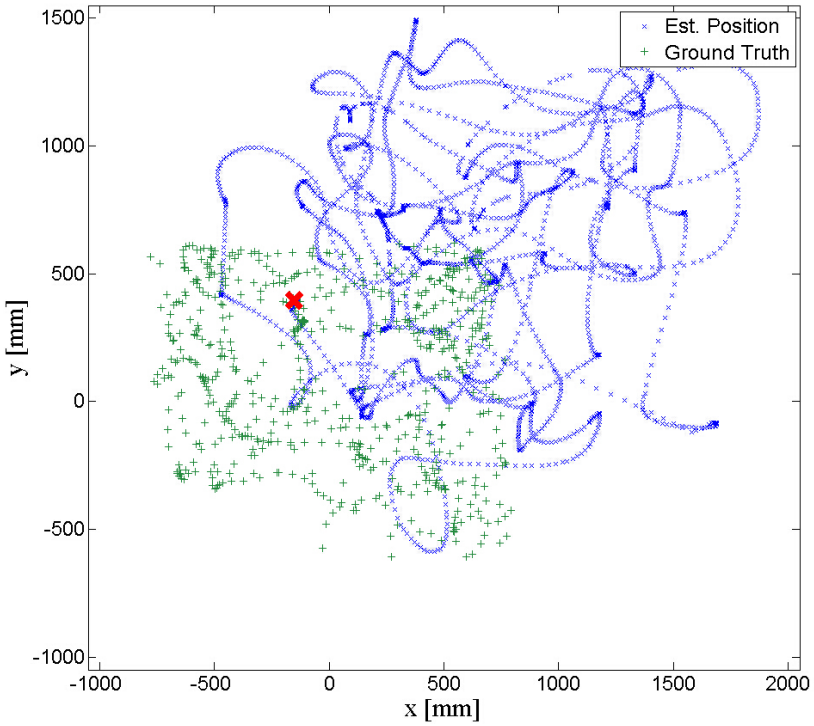


Fig. 7.1: Pose estimation compared with the ground truth in a test flight of $120s$. The blue 'x' markers stand for the estimated value and the green '+' markers are the measured ground truth. The red cross at $(-150, 400)$ is the starting point.

a predefined square path. These two experiments are designed for different purposes: the first experiment tests the impulse response of the control system while the successive waypoints in the second test can be seen as a ramp input to the system. To avoid any possible drift in the position estimation of the quadcopter, the global reference system is used to provide an external true measurement.

7.2.1 Hovering at a Fixed Point

In this experiment, the drone was set to be hovering at $P_d = (0, 0, 0.6)$ for $120s$. The current position of the quadcopter was recorded at a rate of $20Hz$ and plotted in Fig. 7.3. It is quite straightforward to say that the controller achieved a good performance and all the sampled positions during the test were restrained into a cuboid space with a dimension of about $0.1m \times 0.2m \times 0.1m$. More details of the properties of this test flight are given in Tabel 7.1. To make it more persuasive, another experiment was done to test the controller performance against disturbances. In this experiment, external forces were given by pushing the quad-

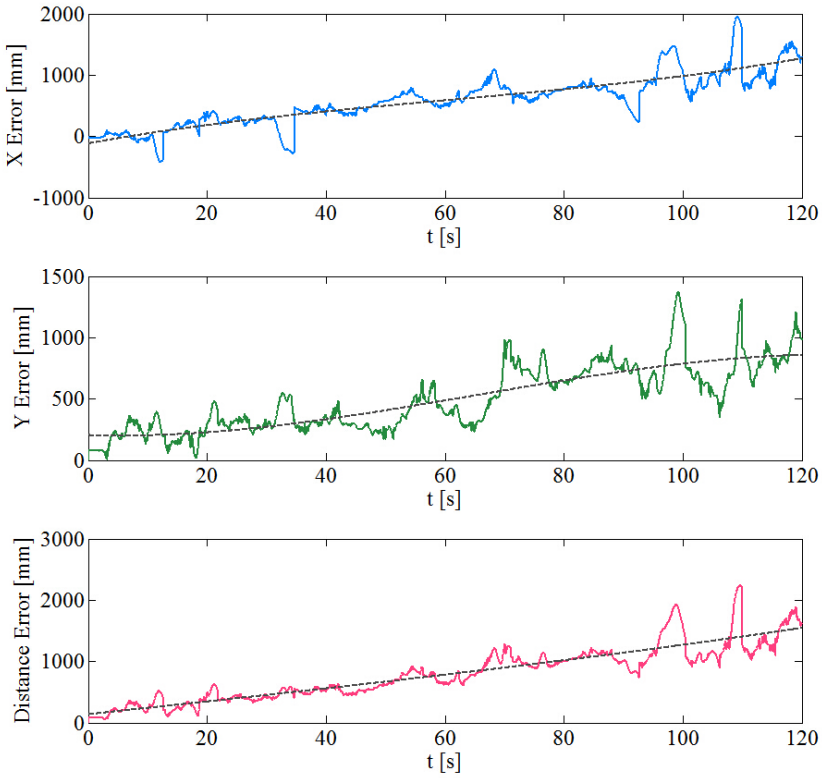


Fig. 7.2: Position Estimation Error x_e, y_e and their squared roots (distance). The black dashed curve in each sub-graph indicates the error trends varying with time which are fitted by third order functions.

copter away from its hovering point. The control inputs and outputs are given in Figure 7.4. A rotational force and a translated force were applied to the quadcopter at different moments and from the system response, it can be seen that the designed control is robust to large external forces and disturbances and the AR.Drone returned back to the original position in a relative short time.

Table 7.1: Position Errors Analysis when Hovering at a Fixed Point

	x (cm)	**y (cm)**	**z (cm)**
RMSE	4.45	6.60	2.79
Std. Dev.	4.39	6.57	2.79
Max. Error	14.30	19.80	8.50

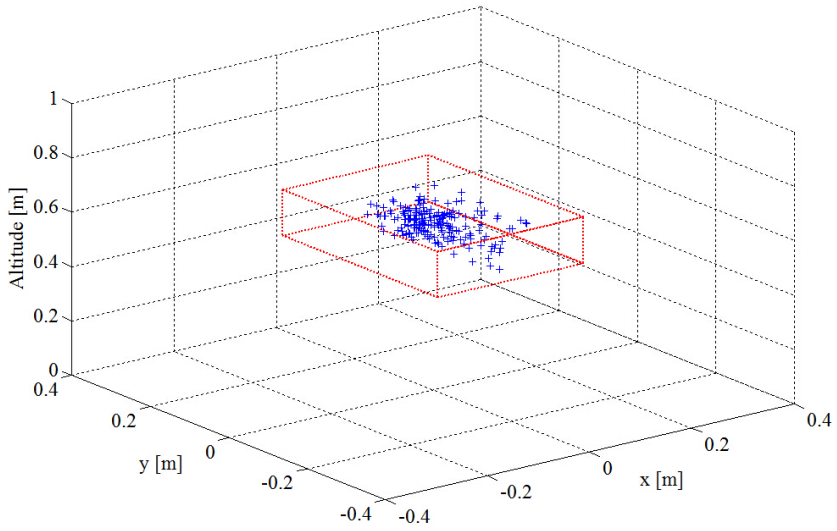


Fig. 7.3: Hovering at a fixed position where $(x_d, y_d, z_d, \psi_d) = (0.0, 0.0, 0.6, 0.0)$. The result shows that in this case the quadcopter is hovering with a high accuracy. The cuboid region outlines the maximum absolute errors on each axis.

7.2.2 Following a Square Shape

In this experiment, the drone was designed to follow a square figure. The waypoint sequences were calculated in real time and fed into the position controller as a function of time t. In the experimental test, the quadcopter followed the square for four rounds. The trajectory is plotted in Fig. 7.5 and the error analysis is given in Table. 7.2. The control performance, as could be expected, has been decreased but still keep at a good level. It also can be seen that the overall performance (Fig. 7.5a and 7.5b) is worse than that of either in x or y axis (Fig. 7.5c and 7.5d). This is because the designed controller controls x and y as two independent systems and does not consider their couple effects. These could be improved if a coupling coefficient is added into the position controller which takes the distance error as a calibrate ratio of the PID parameters.

Table 7.2: Square Shape Figure Following Error

	x (cm)	**y (cm)**	**z (cm)**
RMSE	6.61	10.71	2.33
Std. Dev.	6.32	10.72	2.33
Max. Error	17.30	27.50	7.90

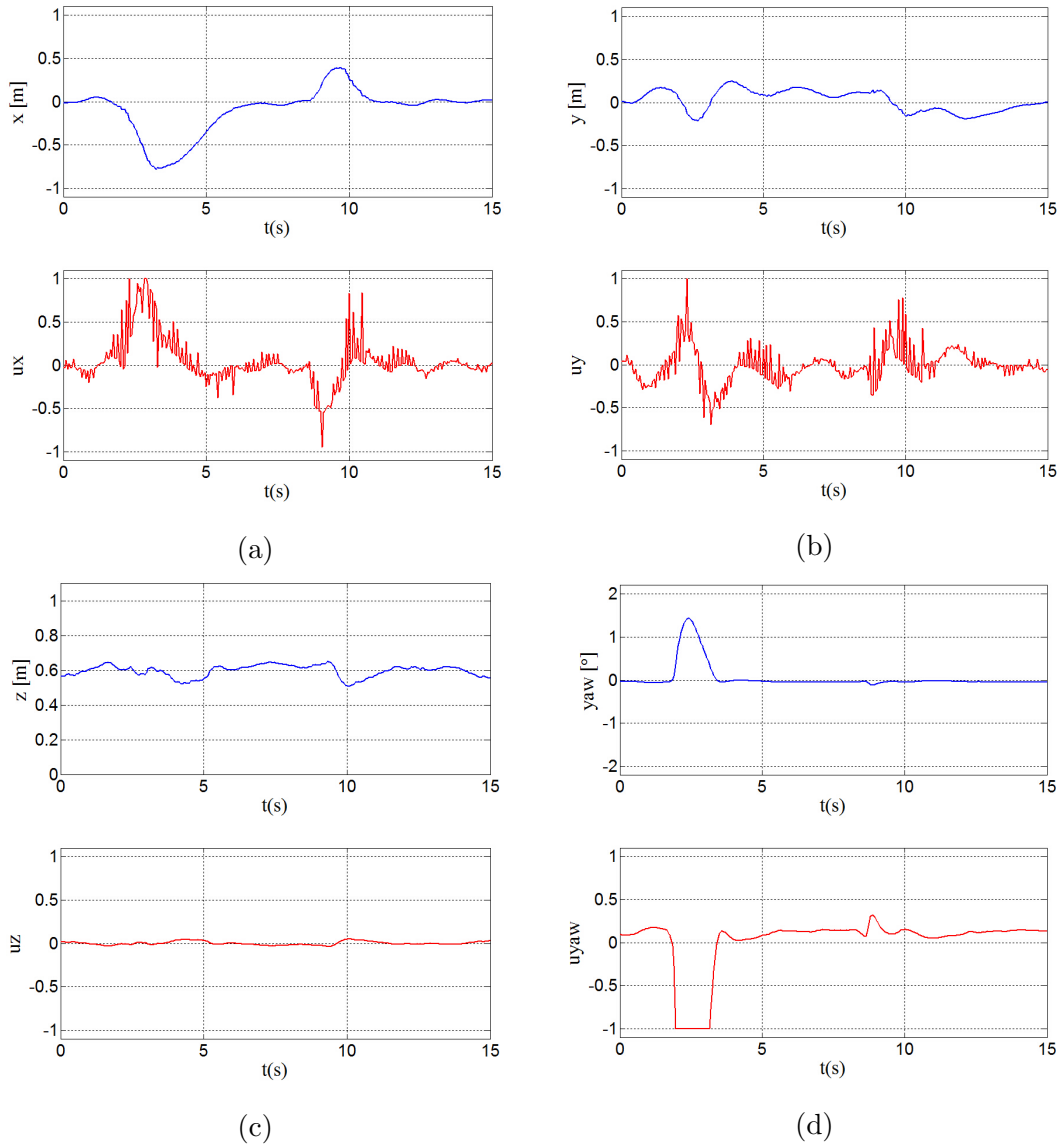


Fig. 7.4: System Inputs and Outputs under Disturbances. **(a)** X Position and U_x, **(b)** Y Position and U_y, **(c)** Z Position and U_z and **(d)** Yaw Angle and U_ψ . A twist force was given at time $3s$ which resulted a dramatic $90°$ change in Yaw angle and a translational force pushed the drone away in x direction at $9s$. From the system response after implied these disturbances, it can be concluded that the designed position controller is stable and robust to disturbances.

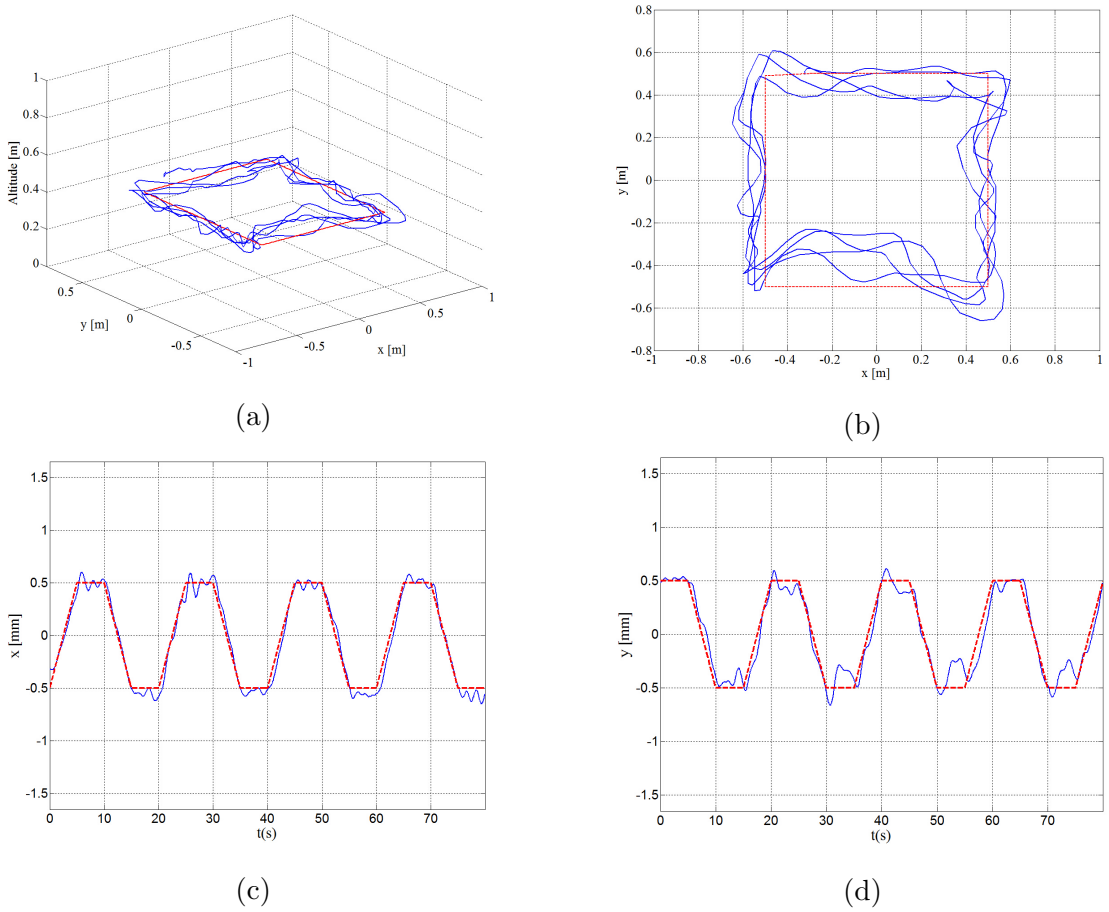


Fig. 7.5: Square Shape Figure Following Test Result. In this experiment, the quadcopter was designed to follow a square shape figure for four rounds. The red dashed lines are the desired path and the blue lines are the actual trajectory. **(a)** is the 3-D plot of the trajectory, **(b)** is the top view and **(c)** and **(d)** are the positions in x and y axes.

7.3 Visual Tracking Performance

The last experiment verified the performance of visual servoing which is a integration of visual detection and position control. In this test, the quadcopter firstly tracked a stationary UGV and then followed a UGV which was moving in a random way. The performance was evaluated by the visual detection hit rate and the estimated position error from the drone to the UGV.

7.3.1 Tracking a Stationary Target

In this experiment, the UGV is placed at a fixed position while the UAV was manually fly to a position vertically above the UGV and then set into autonomous mode. Two experiments were done and each experiment collected 1000 valid visual measurements. The second experiment was conducted with better lighting condition while the first one was under poor lighting environment.

The experimental result are given in Table 7.3. Compared with previous experiments, the RMSE, standard deviation and maximum absolute error have all risen 2 - 5 times and this increased errors were involved by instability of the vision algorithm. The issues of the visual detection include lost the target in the image frame, detected a false object and failure of recognising. A false measurement would be the worst, since this will result in a jump of the reference to the position controller and will lead to unstable behaviours. Losing the target will also increase the control error because in this case, the IMU odometry will be used to estimate the current position of the quadcopter so it can go back to the last position where the UGV was. Beacause of the aforementioned drifts, the IMU odometry will overestimating the relative position to the UGV thus lead to a higher level of error.

The hit rate is calculated by the ratio between the overall processed images number and the number of positive results. From the two contrast tests, one can make a conclusion that the performance of the visual algorithm is highly depended on the lighting condition of the environment. It is also worth mentioning that from some other experiments, the indirect sunlight could lead to a highest hit rate. Being too bright or too dim will cause noisy measurements of the camera

and takes the risk of a considerable false measurements.

Table 7.3: Visual Tracking Estimated Error Analysis (Stationary)

	RMSE (cm)	**Std.Dev. (cm)**	**Max.Error (cm)**	**Hit Rate**
Experiment 1	30.79	28.95	266.47	73.26 %
Experiment 2	18.04	17.16	147.50	96.24 %

7.3.2 Tracking a Moving Vehicle

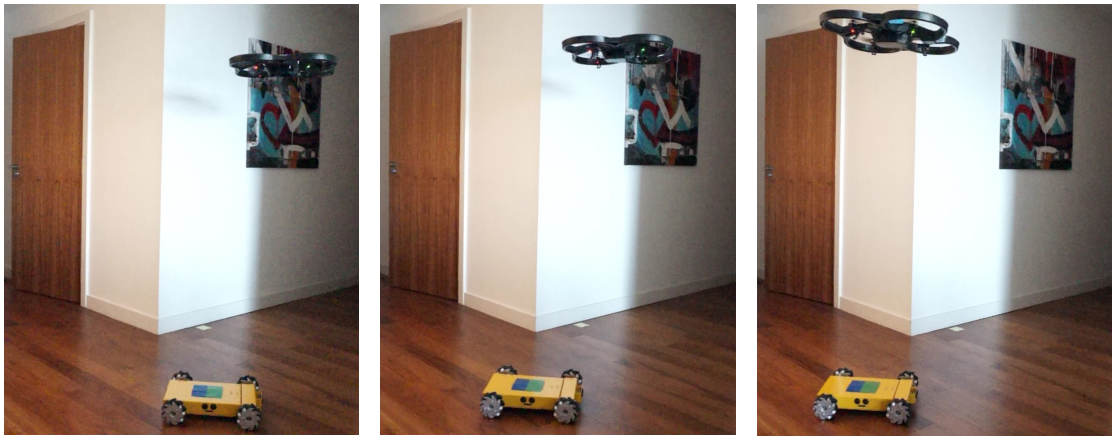
Tracking a moving vehicle is a real challenging work to do. In this case, it is difficult to estimate the current and future position of the UGV. Also, the integrated visual odemotry will lose accuracy for the reason that the movement of the UGV will confuse the optic flow in the image and result in invalid speed estimations.

In this experiment, the UGV was moved with a relative slow speed to improve the success rate of visual tracking. Figure 7.6 illustrates one of the succeed flights where the UAV was flying after the UGV with a response delay of about $1s$. This delay was introduced by the wireless latency and an internal α filter used to smooth the estimated positions of the UGV.

Two experimental tests were given and the properties are listed in Table 7.4. Compared with the stationary case, the estimation error have increased by an average rate of 45.7 % and the vision hit rate have decreased by 16.5 %. To further analyse the problem, the relationship between the estimated error and the vision availability had investigated as Figure 7.7. It can be seen from the diagram that the detection failures have a high influence to the estimated error. As mentioned before, this is caused the drift of the IMU position estimation and could be improved if a better position estimation method is used, such as vision based navigation methods PTAM and V-SLAM.

Table 7.4: Visual Tracking Estimated Error Analysis (Moving)

	RMSE (cm)	**Std.Dev. (cm)**	**Max.Error (cm)**	**Hit Rate**
Experiment 1	51.85	49.46	388.81	69.0 %
Experiment 2	24.81	22.09	128.01	72.6 %



(a) (b) (c)

Fig. 7.6: Successive frames showing that the UAV is properly following the UGV.

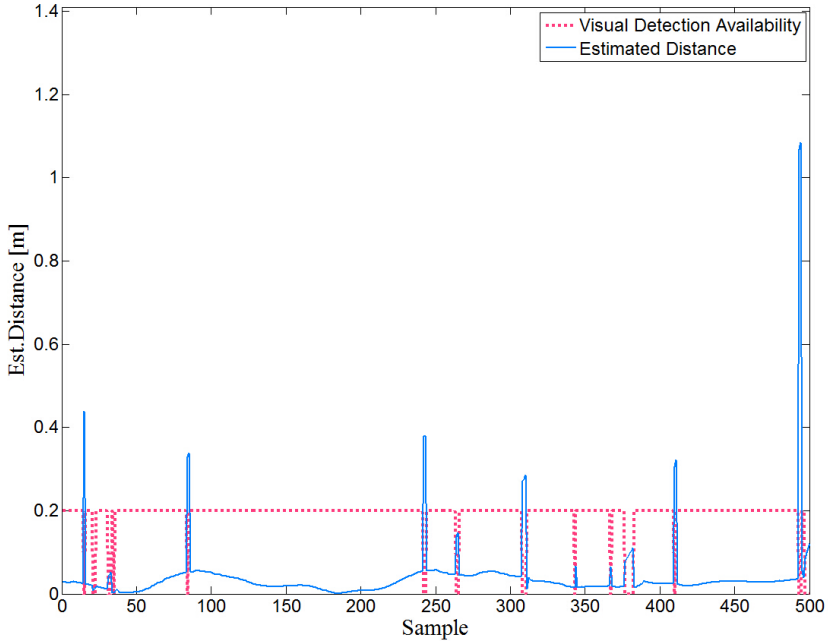


Fig. 7.7: The Analysis of the Relationship between the Estimated Error and the Availability of the Visual Detection. As a conclusion of this diagram, the estimated error has a strong link with the availability of the vision algorithm. Whenever the visual system is failed, a large estimation error will be involved into the system because of the use of Dead Reckoning after the lost moment. The distance error will return to a normal level when the visual systems recovered from failures.

7.4 Summary

Plenty of test flights have shown that this system has a satisfactory performance. The position controller has a fast response and is robust to disturbances. In terms

of the performance of the visual system, it has an average of 84.75 % hit rate with stationary target and 70.8 % in case of the moving UGV tracking. Some problems of this system also exposed during the experiments. One is the coupling effects of x and y dynamics which could lead to oscillations around the stationary point. Another issue is the visual algorithm does not have a good performance under low lighting condition or if the detecting object is moving. As a summary, the designed visual servoing system is valid and effective while more improvements could be done to achieve a better control result.

Chapter 8

Conclusion and Further Work

This thesis presents a framework for autonomous visual tracking using a low cost AR.Drone quadcopter. The AR.Drone has a competitive number of sensors and can be controlled wirelessly by a computer or a tablet with the AR.Drone SDK. The quadcopter has two on-board cameras and in this study, the camera pointed vertically has been used as the source for computer vision. It is mentionable that the quadcopter is controlled in a centralized way by a ground station running ROS system. The raw image is obtained from the drone and processed with the Blob vision algorithm. The detected position will then be transformed into world frame as the reference of the position controller. Finally, the computed $(u_\theta, u_\phi, u_{\dot{z}}, u_{\dot{\phi}})$ control inputs which are normalized into $[-1.0, 1.0]$ will be transmitted back to the quadcopter.

To achieve a fast and real-time image processing, the non-model colour based Blob Detection algorithm is used to detect the relative position to the coloured marker on top of the UGV. This algorithm analysis the colours of the image in HSV space. Erosion and Dilation are used to remove noise pixels in the threshold image. The Blob algorithm checks every contour in the processed image, evaluate its moment and selects the one with the maximum area as the measured object. The same procedure applies for both green and blue parts to obtain their central and the position of the object is simply estimated as the central of those two points.

It is mentioned in other work that, when the raw angular rate is small, the system can be decoupled into four independent SISO sub-systems. In order to con-

trol the position of the quadcopter in 3-D space, four PID controllers are designed with a control frequency of $20Hz$. The parameters of the controllers are tuned manually and are based on some test flights. According to later experiments, this controller has a good performance and is stable in the presence of disturbances and transmission delay.

The behaviour of the UAV is controlled by a finite state machine, in which the strategy for searching target is realized. A human interface with a joystick as the input is designed to provide a way to manually control the UAV when it is necessary. A low cost video capturing system is also designed during the project to evaluate the performance of the quadcopter in a more objective way.

Three classes of experiments were conducted and analysed to verify the validity of the proposed system. It could be seen that the designed visual servoing system has a competitive performance against those designed in other works. However, there are some further work which can improve the system specification to a higher level:

1. It is noticeable that in this work no valid model of the quadcopter was obtained. System identification is the key to obtain an approximate model which could used during the controller design process. Some efforts have already been done and it is promising to realize a model-base controller, such as MPC or IMC.
2. The designed system uses a centralized control structure which can be replaced by a on-board controller that can eliminate the latency introduced by wireless transmission.
3. The color-based visual algorithm is too depended on the lighting conditions of the environment and could easily get a false measurement. To address this problem, the Hough Transformation can be applied upon the result of Blob Detection to examine the shape features of the object and further improve the success rate.
4. The proposed Dead Reckoning position estimation only works for a short period of time. To achieve a long time estimation in indoor environment, visual based navigation methods should be used.

Some of these issues can be easily solved while some may take a relative long

period. System design should be an iterative way but as a consequence of the time limitation of this dissertation, the mentioned problems can not be well addressed in this work.

REFERENCES

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

[2] R. Bartak, A. Hrasko, and D. Obdrzalek. A controller for autonomous landing of ar. drone. In *Control and Decision Conference (2014 CCDC), The 26th Chinese*, pages 329–334. IEEE, 2014.

[3] S. Bouabdallah, M. Becker, V. de Perrot, and R. Y. Siegwart. Toward obstacle avoidance on quadrotors. Proceedings of the XII International Symposium on Dynamic Problems of Mechanics (DINAME 2007), Ilhabela, SP, Brazil, February 26-March 2, 2007, 2007.

[4] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[5] P.-J. Bristeau, F. Callou, D. Vissière, N. Petit, et al. The navigation and control technology inside the ar. drone micro uav. In *18th IFAC world congress*, volume 18, pages 1477–1484, 2011.

[6] I. Carro Perez, D. Flores-Araiza, J. Fortoul-Diaz, R. Maximo, and H. Gonzalez-Hernandez. Identification and pid control for a quadrocopter. In *Electronics, Communications and Computers (CONIELECOMP), 2014 International Conference on*, pages 77–82. IEEE, 2014.

[7] E. R. Davies. *Machine vision: theory, algorithms, practicalities.* Elsevier, 2004.

[8] W. M. DeBusk. Unmanned aerial vehicle systems for disaster relief: Tornado alley. In *AIAA Infotech@ Aerospace Conference, AIAA-2010-3506, Atlanta, GA*, 2010.

[9] DragonFly Inc. *Draganflyer X4-ES Four Rotor UAV Helicopter Aerial Video Platform*, 2013. [Online]. Available at `http://www.draganfly.com/uav-helicopter/draganflyer-x4es/index.php`.

[10] J. Evans, G. Inalhan, J. S. Jang, R. Teo, and C. Tomlin. Dragonfly: A versatile uav platform for the advancment of aircraft navigation and control. In *Proceedings of the 20th Digital Avionics System Conference, Daytona Beach*, 2001.

[11] FESTO. *BionicOpter Inspired by dragonfly flight*, 2013. [Online]. Available at `http://www.festo.com/cms/en_corp/13165.htm`.

[12] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[13] Harvad University. *RoboBees*, 2014. [Online]. Available at `http://robobees.seas.harvard.edu/`.

[14] A. Hernandez, C. Copot, R. De Keyser, T. Vlas, and I. Nascu. Identification and path following control of an ar. drone quadrotor. In *System Theory, Control and Computing (ICSTCC), 2013 17th International Conference*, pages 583–588. IEEE, 2013.

[15] J. Howse. *OpenCV Computer Vision with Python*. Packt Publishing Ltd, 2013.

[16] C. Hui, C. Yousheng, L. Xiaokun, and W. W. Shing. Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision. In *Control Conference (CCC), 2013 32nd Chinese*, pages 5895–5901. IEEE, 2013.

[17] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl. Ar-drone as a platform for robotic research and education. In *Research and Education in Robotics-EUROBOT 2011*, pages 172–186. Springer, 2011.

[18] J. J. Lugo, A. Masselli, and A. Zell. Following a quadrotor with another quadrotor using onboard vision. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 26–31. IEEE, 2013.

[19] J. J. Lugo and A. Zeil. Framework for autonomous onboard navigation with the ar. drone. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 575–583. IEEE, 2013.

[20] A. Martinez and E. Fernández. *Learning ROS for Robotics Programming*. Packt Publishing Ltd, 2013.

[21] M. Mohammadi and A. Mohammad Shahri. Modelling and decentralized adaptive tracking control of a quadrotor uav. In *Robotics and Mechatronics (ICRoM), 2013 First RSI/ISM International Conference on*, pages 293–300. IEEE, 2013.

[22] I. Morar and I. Nascu. Model simplification of an unmanned aerial vehicle. In *Automation Quality and Testing Robotics (AQTR), 2012 IEEE International Conference on*, pages 591–596. IEEE, 2012.

[23] NBC News. *U.S. 'Global Hawk' Drone Joins Search for Kidnapped Nigerian Schoolgirls.*, 2014. [Online]. Available at `http://www.nbcnews.com/storyline/missing-nigeria-schoolgirls/u-s-global-hawk-drone-joins-search-kidnapped-nigerian-schoolgirls-n104696`.

[24] T. Nguyen, G. K. Mann, and R. G. Gosine. Vision-based qualitative path-following control of quadrotor aerial vehicle. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 412–417. IEEE, 2014.

[25] K. Nordberg, P. Doherty, G. Farnebäck, P.-E. Forssén, G. Granlund, A. Moe, and J. Wiklund. Vision for a uav helicopter. In *International Conference on Intelligent Robots and Systems (IROS), workshop on aerial robotics. Lausanne, Switzerland*, 2002.

[26] OpenCV.org. *OpenCV Homepage*, 2014. [Online]. Available at `http://opencv.org/`.

[27] R. Oung and R. D'Andrea. The distributed flight array: Design, implementation, and analysis of a modular vertical take-off and landing vehicle. *The International Journal of Robotics Research*, page 0278364913501212, 2013.

[28] E. Pali, K. Mathe, L. Tamas, and L. Busoniu. Railway track following with the ar. drone using vanishing point detection. In *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014.

[29] J. Pestana, J. L. Sanchez-Lopez, P. Campoy, and S. Saripalli. Vision based gps-denied object tracking and following for unmanned aerial vehicles. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–6. IEEE, 2013.

[30] J. Pestana Puerta, J. L. Sánchez López, I. Mellado Bataller, C. Fu, and P. Campoy Cervera. Ar drone identification and navigation control at cvg-upm. 2012.

[31] S. K. Phang, J. J. Ong, R. T. Yeo, B. M. Chen, and T. H. Lee. Autonomous mini-uav for indoor flight with embedded on-board vision processing as navigation system. In *Computational Technologies in Electrical and Electronics Engineering (SIBIRCON), 2010 IEEE Region 8 International Conference on*, pages 722–727. IEEE, 2010.

[32] I. H. B. Pizetta, A. S. Brandao, and M. Sarcinelli-Filho. A hardware-in-loop platform for rotary-wing unmanned aerial vehicles. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 1146–1157. IEEE, 2014.

[33] M. Saska, T. Krajnik, J. Faigl, V. Vonásek, and L. Preucil. Low cost mav platform ar-drone in experimental verifications of methods for vision based autonomous navigation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4808–4809. IEEE, 2012.

[34] N. Stacy, D. Craig, J. Staromlynska, and R. Smith. 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*, volume 2, pages 699–701. IEEE, 2002.

[35] Z. E. Teoh, S. B. Fuller, P. Chirarattananon, N. Prez-Arancibia, J. D. Greenberg, and R. J. Wood. A hovering flapping-wing microrobot with altitude control and passive upright stability. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3209–3216. IEEE, 2012.

[36] J. Tisdale, A. Ryan, M. Zennaro, X. Xiao, D. Caveney, S. Rathinam, J. K. Hedrick, and R. Sengupta. The software architecture of the berkeley uav platform. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pages 1420–1425. IEEE, 2006.

[37] T. Venugopalan, T. Taher, and G. Barbastathis. Autonomous landing of an unmanned aerial vehicle on an autonomous marine vehicle. In *Oceans, 2012*, pages 1–9. IEEE, 2012.

[38] Vicon. *Vicon in Engineering*, 2013. [Online]. Available at `http://www.vicon.com/Application/Engineering`.

[39] K. E. Wenzel, A. Masselli, and A. Zell. Visual tracking and following of a quadrocopter by another quadrocopter. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4993–4998. IEEE, 2012.

[40] Wikipedia. *Northrop Grumman RQ-4 Global Hawk*, 2014. [Online]. Available at `http://en.wikipedia.org/wiki/Northrop_Grumman_RQ-4_Global_Hawk`.

APPENDIX A

Algorithm 1 The Alogrithm of Discreat PID Controller

procedure (init):
    $e_k^i \leftarrow 0$
    $e_{k-1} \leftarrow 0$
procedure (feedback):
    $e_k \leftarrow ref - measurement$
    $e_k^i \leftarrow e_k^i + e_k \times \delta_t$
    $u_p \leftarrow K_p \times e_k$
    $u_i \leftarrow K_i \times e_k^i$
    $u_d \leftarrow K_d \times ((e_k - e_{k-1})/\delta_t)$
    $output \leftarrow u_p + u_i + u_d$
    if $output > limit_+$ **then**
        $output \leftarrow limit_+$
    if $output < limit_-$ **then**
        $output \leftarrow limit_-$
    $e_{k-1} \leftarrow e_k$
    return *output*