Multi-robot Collaborative Visual Navigation with Micro Aerial Vehicles

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Richard Michael Williams

April 2017

Contents

List of Figures

List of Tables

Notations

The following notations and abbreviations are found throughout this thesis:

AGAST	Adaptive Generic Accelerated Segment Test
BA	Bundle Adjustment
BRIEF	Binary Robust Independent Elementary Features
CCTAM	Centralised Collaborative Tracking and Mapping
CPU	Central Processing Unit
CSfM	Collaborative Structure from Motion
DCTAM	Distributed Collaborative Tracking and Mapping
DOF	Degrees of Freedom
DOG	Difference of Gaussian
DTAM	Dense Tracking and Mapping
DWA	Dynamic Window Approach
EKF	Extended Kalman Filter
EoL	End of Life
EPP	Expanded Polypropylene
FAST	Features from Accelerated Segment Test
GPS	Global Positioning System
GPU	Graphics Processing Unit
HRVO	Hybrid Reciprocal Velocity Obstacle
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
KLT	Kanade-Lucas-Tomasi
LM	Levenberg-Marquardt
LSD-SLAM	Large Scale Semi-Direct Simultaneous Localisation and Mapping
MAV	Micro Aerial Vehicle
MIMO	Multiple Input Multiple Output
ORB	Oriented FAST and Rotated Brief
P3P	Perspective-Three-Point
PID	Proportional Integral Derivative
PnP	Perspective n-Point
PTAM	Parallel Tracking and Mapping
PX4FMU	PX4 Flight Management Unit

RANSAC	RANdom Sample And Consensus
RGB	Red Green Blue
RGB-D	Red Green Blue Depth
RMSE	Route Mean Squared Error
RPY	Roll, Pitch, Yaw
RVO	Reciprocal Velocity Obstacle
SAR	Search and Rescue
SFM	Structure from Motion
SIFT	Scale Invariant Feature Transform
SISO	Single Input Single Output
SLAM	Simultaneous Localisation and Mapping
SSD	Sum of Squared Differences
SSI	Sequential Single Item
SVD	Singular Value Decomposition
SWaP	Size Weight and Power
tf-idf	term frequency-inverse document frequency
UAS	Unmanned Aerial Systems
UKF	Unscented Kalman Filter
VFH	Vector Field Histogram
VO	Velocity Obstacles
ZSSD	Zero-Mean Sum of Squared Differences

Preface

This thesis is primarily my own work. The sources of other materials are identified.

Abstract

Micro Aerial Vehicles (MAVs), particularly multi-rotor MAVs have gained significant popularity in the autonomous robotics research field. The small size and agility of these aircraft makes them safe to use in contained environments. As such MAVs have numerous applications with respect to both the commercial and research fields, such as Search and Rescue (SaR), surveillance, inspection and aerial mapping. In order for an autonomous MAV to safely and reliably navigate within a given environment the control system must be able to determine the state of the aircraft at any given moment. The state consists of a number of extrinsic variables such as the position, velocity and attitude of the MAV. The most common approach for outdoor operations is the Global Positioning System (GPS). While GPS has been widely used for long range navigation in open environments, its performance degrades significantly in constrained environments and is unusable indoors. As a result state estimation for MAVs in such constrained environments is a popular and exciting research area. Many successful solutions have been developed using laser-range finder sensors. These sensors provide very accurate measurements at the cost of increased power and weight requirements.

Cameras offer an attractive alternative state estimation sensor; they offer high information content per image coupled with light weight and low power consumption. As a result much recent work has focused on state estimation on MAVs where a camera is the only exteroceptive sensor. Much of this recent work focuses on single MAVs, however it is the author's belief that the full potential and benefits of the MAV platform can only be realised when teams of MAVs are able to cooperatively perform tasks such as SaR or mapping. Therefore the work presented in this thesis focuses on the problem of vision-based navigation for MAVs from a multi-robot perspective. Multi-robot visual navigation presents a number of challenges, as not only must the MAVs be able to estimate their state from visual observations of the environment but they must also be able to share the information they gain about their environment with other members of the team in a meaningful fashion. The meaningful sharing of observations is achieved when the MAVs have a common frame of reference for both positioning and observations. Such meaningful information sharing is key to achieving cooperative multi-robot navigation. In this thesis two main ideas are explored to address these issues. Firstly the idea of appearance based (re)-localisation is explored as a means of establishing a common reference frame for multiple MAVs. This approach allows a team of MAVs to very easily establish a common frame of reference prior to starting their mission. The

common reference frame allows all subsequent operations, such as surveillance or mapping, to proceed with direct cooperative between all MAVs. The second idea focuses on the structure and nature of the inter-robot communication with respect to visual navigation; the thesis explores how a partially distributed architecture can be used to vastly improve the scalability and robustness of a multi-MAV visual navigation framework.

A navigation framework would not be complete without a means of control. In the multi-robot setting the control problem is complicated by the need for inter-robot collision avoidance. This thesis presents a MAV trajectory controller based on a combination of classical control theory and distributed Velocity Obstacle (VO) based collision avoidance. Once a means of control is established an autonomous multi-MAV team requires a mission. One such mission is the task of exploration; that is exploration of a previously unknown environment in order to produce a map and/or search for objects of interest. This thesis also addressed the problem of multi-robot exploration using only the sparse interest-point data collected from the visual navigation system. In a multi-MAV exploration scenario the problem of task allocation, assigning areas to each MAV to explore, can be a challenging one. An auction-based protocol is considered to address the task allocation problem. The two applications discussed, VO-based trajectory control and auction-based environment exploration, form two case studies which serve as the partial basis of the evaluation of the navigation solutions presented in this thesis.

In summary the visual navigation systems presented in this thesis allow MAVs to cooperatively perform task such as collision avoidance and environment exploration in a robust and efficient manner, with large teams of MAVs. The work presented is a step in the direction of fully autonomous teams of MAVs performing complex, dangerous and useful tasks in the real world.

Acknowledgements

The work presented in this thesis represents the completion of one of the biggest challenges of my life to date. I would not have reached this point without the guidance, understanding and support from all the people who have been involved over the last few years. A lion's share of that gratitude must go to my supervisory team Prof. Boris Konev and Prof. Frans Coenen. To Boris, I owe not only thanks for his insight and guidance throughout the years but also for giving me the opportunity to pursue a PhD in the first place. It has been a journey filled with many challenges, surprises, successes and failures but thanks to you it has been a fruitful one which I will cherish in the years to come. To Frans, I owe a lot for believing in me even when I had doubts, for always helping me stay on course and keep focus which has been invaluable while working to complete my thesis. I would also like to thank my advisers Prof. Wiebe van der Hoek, Prof. Michael Fisher and Dr. Muhammad Khan for their insightful questions and helpful comments that helped me see my work in a different light.

I would also like to thank Prof. Karl Tuyls for opening up new opportunities for me; both for your help in securing a teaching position and allowing me to participate in the Robocup and Rockin competitions. I will always be grateful to you for the invaluable lessons those experiences have taught me. To Prof. Simon Parson and Dr. Betsy Sklar, I owe thanks for their guidance and kindness in allowing me to use their lab space for my experiments. To my friend and colleague David Geleta I owe a special thanks, for being there for the whole epic journey from our Undergraduate days to the final days of our PhDs. You have been my Samwise and my Frodo, and I will forever be in your debt. To my fellow PhD students Eric Schneider, Jeffery Rafael, Gabrielle Dos Santos, Matoula Kotsialou, Bastian Broecker, Daniel Claes, Joscha Fossel and Dr. Daan Bloembergen who have been there for all the late nights, coffee breaks, wonderful celebrations, enjoyable and sometimes surreal University events, each of you have made these experiences all the more enjoyable by your involvement.

Finally and most importantly to my family. It is to my father, Michael that I owe thanks for sparking my love of technology. For putting up with my persistent, misguided attempts to electrocute myself which characterised many of my early attempts to emulate your mastery of all things electronic. For your patience and guidance throughout the years and for always being there when I had a difficult problem to solve. It is to my my mother I owe thanks instilling in me the courage, determinations and stubbornness

that got me through some of the most difficult parts of my academic career so far. My siblings, Chris and Natalie, for being always being there to talk, laugh or cry with.

Chapter 1

Introduction

1.1 Micro Aerial Vehicles

For several decades research into Unmanned Aerial Systems (UAS) has been dominated by the military and aerospace industries. The barrier for entry into these fields being the ability to deploy and support a large unmanned aircraft. However with advances in sensor and battery technologies, powered largely by the mobile phone market, it has become possible to develop smaller UAS. These small UAS come in a variety of configurations from fixed-wing aircraft (Figure 1.1 (left)) capable of long duration, high altitude flight, to highly stable and manoeuvrable rotary-wing craft such as helicopters and quad, hexa and octocopters. More recently several hybrid Vertical Take-Off and Landing (VTOL) systems have been developed capable of transitioning between hovering and fast forward flight (Figure 1.1 (right)).

The number of civilian and humanitarian applications of these craft has also been growing, systems such as the SenseFly eBee (Figure 1.1 (left))see have been used in the aftermath of several natural disasters for rapid damage assessment and monitoring of temporary settlements [35]. Several companies including Google and Amazon have begun development of small UAS for tasks such delivering medical supplies to remote locations as well as commercial goods delivery in urban environments.

FIGURE 1.1: Small UAS: The SenseFly eBee (left) and Project Wing by Google (right) a VTOL craft delivering a package in the Australian outback. Image credits to Sensefly and Google respectively.

Micro Aerial Vehicles (MAVs) are a class of small UAS typically with limitations on size and payload. The term MAV is sufficiently ambiguous to have been used to describe craft weighing from 100 grams to several kilograms as shown in Figure 1.2. In the context of this thesis a MAV is defined as an aerial vehicle which weighs under 5 kilograms, is less than 1 metre in length and is capable of operating safely within a typical indoor environment such as office buildings. In this thesis the focus is on multi-rotor MAVs, specifically quadcopters; however the work presented is applicable to most rotary-wing aircraft and with some modifications could be applied to fixed wing aircraft as well.

FIGURE 1.2: Various MAV platforms: The Crazyflie Nano [22], a 20 gram MAV platform (left); the Flyabillity Gimball [27], a coaxial rotor driven platform surrounded by a gimbal mounted protective cage for enhanced collision protection (middle); and the Honeywell RQ-16 T-Hawk [81], a ducted fan Vertical Take-off and Landing (VTOL) MAV developed by the United States military (right).

There are numerous applications for MAVs such as: Search and Rescue (SaR), aerial inspection, exploration and conservation activities such as wildlife or crop monitoring [37]. These applications all have one thing in common, a requirement for a robust and reliable navigation system. The standard navigation solution for MAVs utilises the Global Positioning System (GPS) as the main localisation solution. However, GPS has several limitations in terms of both accuracy and coverage (more in Chapter 3) and there are many applications where GPS cannot be used, for example indoor SaR [80].

Given that SaR is an ideal application for MAVs their actual deployment in real SaR situations is surprisingly infrequent as noted by a recent study by Murphy et al. [80]. Murphy et al.s study covers all cases of robots being deployed in disasters between 2001 and 2013 and shows that of the 34 documented incidents there were only 10 incidences of MAVs being deployed. Additionally, Murphy et al. highlight the fact that while there have been cases where MAVs have had autonomous capabilities they were never used in any of these 10 cases. Murphy et al. give several reasons for the lack utilisation of autonomous capabilities, she notes that in several cases the craft were operating close enough to structures to cause interference to the crafts (GPS) based navigation systems. In general it comes down to a lack of trust in the autonomous systems; the pilots did not feel comfortable in delegating control to an autonomous navigation system even in cases where the autonomous system was more than capable of achieving the current objectives. This is evidenced by the fact that while there is increasing use of MAVs in real world

applications these tend to be either manually controlled or in highly structured settings where lack of advanced capabilities such as collision avoidance and scene perception are not required.

One major difficultly in multi-rotor research is the significant investment of both time and capital required to set-up a safe, reliable framework with which to conduct research. Some of the most common MAV research platforms are the AscTec[1] line of multi-rotor MAVs. These platforms typically cost in the range of €5000-8000. These platforms are also by no means complete solutions as additional sensors and on-board computers are still required. This also does not include any localisation system; which, depending on the aims of the individual researchers, may not be an issue however it is argued here that, for safety reasons, some form of localisation system is required. This is mainly due to the highly dynamic nature of MAVs and the lack of robust emergency recovery. It's common to include an emergency stop button on a ground robot which will immediately cut-off power to the actuators in the case of a runaway robot. However this is not possible with MAVs; indeed immediately cutting power to the actuators on a multi-rotor MAV virtually guarantees significant damage to the craft. However with the inclusion of a localisation system the safety of the craft immediately improves. They allow the MAV to perform controlled emergency landings or restrict their movements to only a specified area (this is often called fencing). The specifics of multi-rotor localisation systems will be discussed in Chapter 3. However, given the above the barrier for entry into the field of practical MAV research in general, and multi-MAV research in particular, is very high from both a monetary and safety standpoint. This has been noted by other researchers as a recent survey by Farid Kendoul of unmanned rotor-craft systems (a general term used to refer to both small and large rotary wing craft) [51]. Kendoul noted that while significant theoretical work has gone into autonomous navigation of UAS there is a gap between the theoretical work and the practical experimentation done. This thesis proposes that the barrier for entry, as discussed above has influenced development of this gap. By focusing on the development of solutions for low-cost, computationally constrained platforms the intention is to not only address a challenging research topic, but also help close the gap between theory and experiment and encourage more researchers to conduct practical experiments.

1.2 Autonomous Navigation for MAVs

In this Section the problem of autonomous navigation is introduced and discussed in the context of MAVs. The autonomous navigation problem is typically divided into three main challenges:

- Localisation: Where are the MAVs relative to their environment and each other?
- Mapping: What does the environment look like?

[1] `http://www.asctec.de/`

- Navigation: What path must the MAV follow in order reach a target location? Also, given a path, what control commands are required to follow it?

The localisation problem entails determining the pose (position and orientation) of a robot with respect to its environment based purely on the processing of sensor data. A reliable means of achieving this to give the robot a model of the environment in the same, or similar format, as its sensor data. For example if a robot, equipped with a camera, is given a model of its environment consisting of visual features, then it can solve the localisation problem by comparing the features it sees to those in the model. Constructing a model by hand is not always easy, how does one construct a model of the visual features in a room by hand? One way to solve this problem is to make use of an existing means of localisation. Then a model of the environment can be constructed by incrementally fusing observations of the environment together. This is referred to as mapping the environment. This requires some existing, usually external, localisation system and a separate mapping phase before the system can be deployed to do anything useful. This is not ideal with respect to many applications. It would be ideal if the robots were to be able to localise in novel environments, this means being able to simultaneously localise within and construct a map of a previously unknown environment. This is commonly referred to as the Simultaneous Localisation and Mapping (SLAM) problem [5].

The navigation problem is also divided into two parts: path planning and trajectory execution. Path planning is the problem of determining the route from a robot's current location to it's goal location, typically in the shortest time possible while avoiding all obstacles. The output of the path planner is a trajectory to be executed by the robots control system; depending upon the type of robot platform the trajectory execution problem can have its own challenges and constraints. For example, on fixed-wing MAVs trajectory execution can be complicated by the fact that the craft must keep moving in order to stay in the air. The path planning and trajectory execution problems can also be tightly coupled with the SLAM problem, as whatever environment model used for mapping and localisation is typically used for path planning. The reliability of the localisation method also has a big impact on the trajectory execution. For example a visual localisation approach may be affected by motion-blur; therefore, during trajectory execution, the MAV should avoid rapid accelerations as these may result in localisation failure. The typical high-level architecture of a robot navigation system is show in Figure 1.3. As the work in this thesis aims to explore the navigation problem from a multi-robot perspective "Other MAVs" have been included in the diagram to highlight those components in the navigation system that are affected by the inclusion of other robots into the system. The multi-robot navigation problem is explained in more detail in the next section.

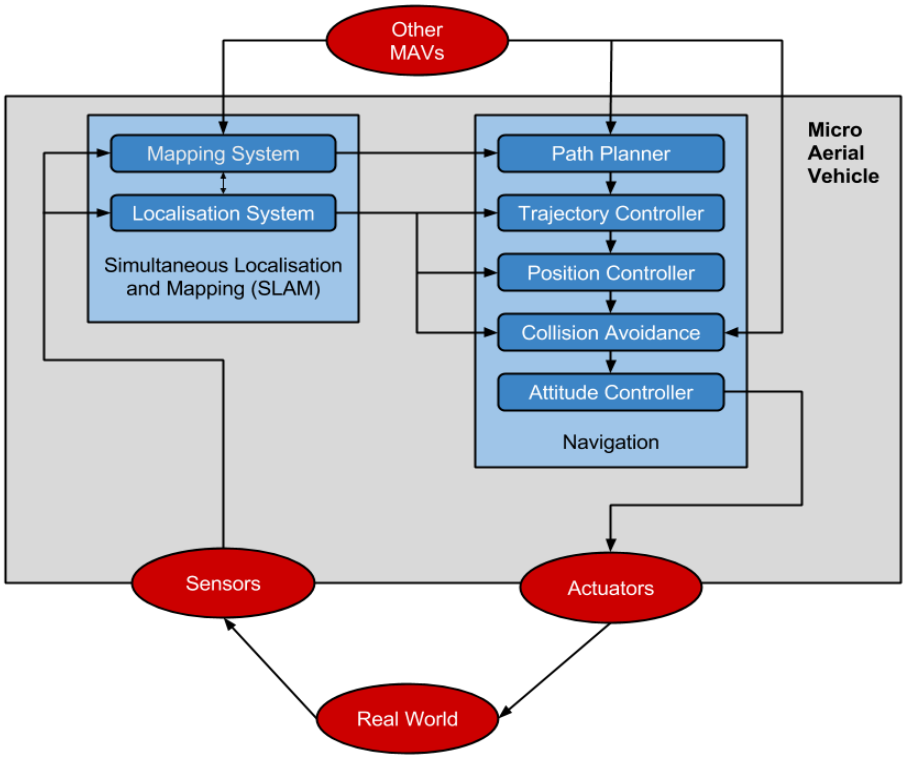


FIGURE 1.3: The general high-level system architecture for an autonomous MAV [119]

1.3 Multi-robot Navigation and Coordination

In this thesis the focus is the problem of multi-robot visual navigation on MAVs and how this relates to the problem of multi-robot coordination for tasks which are tightly coordinated. There is much work, particularly in the field of swarm robotics, which investigates loosely-coordinated solutions in which individual robots have either limited or no awareness of the explicit goals and behaviours of other robots, interaction is limited and coordination is emergent rather than strictly defined. The work presented in this thesis is concerned with direct coordination between multiple robots for tightly coupled tasks which require significant interaction and coordinated execution. This places additional constraints on the navigation system as not only must the robots be able to localise themselves (while mapping) but they must be able to localise themselves within a common coordinate system.

This becomes apparent when a simple search and rescue scenario is considered: two MAVs are tasked with searching a building to find a person in need of medical attention. MAV1 is smaller and faster so is tasked with exploring the building to find the person. MAV2 is larger and can carry a heavier payload and is tasked with both exploring the building and bringing the medical supplies. Both MAVs have no map of the building so are required to build one as they go. MAV1 explores the building and eventually finds the person and communicates to MAV2 the location of the person according its own map. However MAV2 is unable comply as the communicated coordinates are in MAV1s map frame which is unknown to MAV2. Even if MAV1 were to share its map with

MAV2 there is no guarantee this would result in success as it requires MAV2 to combine the two maps which is only possible if there is a common point of reference. In this thesis two multi-robot coordination problems are explored and use is made of them as case studies to verify the performance of the proposed visual navigation approaches. The first is a market-based approach to environment exploration. This tackles the problem of environment exploration using an auction-based protocol to assign exploration goals to individual MAVs. Another essential coordination task when dealing with groups of aerial vehicles in close proximity is that of MAV-to-MAV collision avoidance. A multi-robot distributed approach to collision avoidance based on velocity obstacles is also explored.

FIGURE 1.4: The MAV platforms used for the work presented in this thesis, the standard Parrot AR Drone (left) and a custom 3D printed, MAV platform (right).

1.4 Research Questions and Contributions

Multi-robot Visual Navigation is a complex problem. In this thesis the problem is explored from the standpoint of two important requirements, those of scalability and robustness. The goal is to develop a robust, scalable multi-robot visual navigation system capable of being deployed using low cost MAVs (see Figure 1.4). The research goals that the work presented in this thesis seek to address are summarised by the following research questions:

1. How can a Visual SLAM and appearance based localisation be used to support the autonomous navigation of large teams of low cost Micro Aerial Vehicles?

2. Given the above how does the architecture of the proposed navigation system affect its scalability and robustness?

3. The barrier for entry for practical research using multiple MAV systems is still high due to the cost of the most commonly used external localisation systems. Can this be addressed by the introduction of alternate approaches based on visual SLAM and low cost platforms?

4. Can a visual SLAM based navigation approach be used to support more high level research such as multi-agent coordination?

In the context of the above research questions the following is a summary of the key contributions of the work presented in this thesis.

1. An examination of the concept of using place recognition to enable multi-robot visual navigation for teams of MAVs which is presented in Chapter 4. This work explores this idea with a proof-of-concept implementation of a centralised multi-robot visual navigation system for the Parrot AR. Drone. The approach is based on Parallel Tracking and Mapping (PTAM) [54], a ground-breaking visual SLAM approach developed by Klien and Murray. This facilitates the use of an extremely low cost ($< £300$) and light weight (< 500 grams) MAV platform for multi-robot research.

2. A general, scalable, partially distributed tracking and mapping system for teams of MAVs which is presented in Chapter 5. Here our previous fully centralised approach is built upon to achieve a more robust, highly scalable, distributed visual navigation system. A more general state estimation approach together with alternate stereo initialisation methods make the distributed approach applicable to a range of MAV platforms.

3. Experiments to analyse the performance of both the centralised and distributed approaches in terms of localisation performance, scalability and robustness which are presented in both Chapters 4 and 5. It is shown that both approaches exhibit on average a Route Mean Squared Error (RMSE) of less than 10 centimetres and it is demonstrated that the centralised approach is capable of scaling to teams of up to 4 MAVs. In contrast experimental results are presented which show the distributed approach is capable of scaling up to 20 MAVs. Finally the increased robustness to network delay afforded by the distributed approach is demonstrated. This increased robustness is shown to improve the reliability of real-time motion tracking and localisation.

4. A demonstration of how the precision and reliability of the proposed visual navigation approach, in combination with a reciprocal velocity obstacle based position controller, can be used to solve the difficult problem of mid-air collision avoidance. This work is presented in Chapter 6.

5. An application utilising the sparse feature-based map produced by our visual navigation system to implement an auction-based multi-robot environment exploration system. This work is presented in Chapter 6.

6. Open source implementations of all software and hardware developed for this thesis aimed at lowering the bar for entry into this line of research. It is the author's hope that the available of the software developed in this thesis will encourage more researchers to conduct real physical experiments using MAV platforms and help bridge the knowledge gap discussed previously. This is discussed in Appendix A.

1.5 Thesis Outline

The remainder of this thesis is organised as follows. Chapter 2 introduces preliminaries including the notation used as well as the theoretical background of the camera model, feature detection, structure from motion, state estimation and control. Chapter 3 looks at state of the art solutions to MAV navigation and motivates the use of vision as the primary navigation sensor. Further common visual localisation methods, for both single MAV as well as teams of MAVs are reviewed. While the main focus of this thesis is visual navigation two multi-robot coordination case studies: robot-to-robot collision avoidance and multi-robot environment exploration are also considered in Chapter 4. The ideas and algorithms developed to address these case studies are discussed as well as their applicability in the evaluation of the visual navigation approaches developed for this thesis.

In Chapter 5 the idea of appearance based localisation to enable multi-robot visual navigation is explored; a proof of concept implementation is developed which forms the basis of one of the contribution of this thesis namely a centralised multi-robot visual navigation system. A complete quantitative evaluation of the framework is also presented including localisation, mapping and scalability experiments. In addition the framework performance in two multi-robot coordination tasks (collision avoidance and exploration) is evaluated.

In Chapter 6 the idea of a partially distributed approach to multi-robot visual navigation is explored. The chapter explores the benefits in terms of performance, robustness and scalability compared to the centralised approach. A platform agnostic implementation of this distributed visual navigation approach is also presented. Chapter 5 also presents an evaluation of the new distributed approach in terms of localisation accuracy, robustness to delay and bandwidth requirements. The distributed approach is also put to the test in the two multi-robot coordination tasks (collision avoidance and exploration). Finally Chapter 7 concludes the thesis with an overall analysis of the research as well as a discussion of future work.

Chapter 2

Preliminaries

In this chapter the foundational concepts of geometry, computer vision and Simultaneous Localisation and Mapping (SLAM) used throughout this thesis are introduced. In Section 2.1 we start by fixing the notation for geometric primitives and describe a number of relevant transformations. Section 2.2 explores how we represent images, in particular we look at perspective projection and the pinhole camera model. In section 2.3 we look a extracting interest points from images and how they may be used to solve problems such as image stitching and camera tracking. Section 2.4 covers theory and algorithms for deriving 3-dimensional (3D) structure from 2-dimensional (2D) images often referred to as photogrammetry or Structure from Motion (SfM). We conclude the chapter in Sections 2.5 and 2.6 by looking at two related problems in robotics: state estimation using Kalman filters and feedback control. The material in this Chapter in based on the excellent book by Szeliski [109] (in particular Chapters 2, 4 and 7).

2.1 Points and Vectors

In this section we fix the notation and briefly describe the geometric primitives used in this work. Starting with points, we represent a point in terms of either: (i)n-dimensional Cartesian coordinates or (ii) $n+1$ homogeneous coordinates where points differing by only their scale factor are equivalent:

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \tilde{\mathbf{X}} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Where

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}$$

To improve clarity we distinguish between a 2D and 3D point/vector with capitalisation. For example a 2D point x in homogeneous coordinates or a 3D point X in homogeneous coordinates:

$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \qquad \tilde{\mathbf{X}} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

2.1.1 Rigid Body Transformation

Points and vectors are typically represented with respect to a designated coordinate frame. Where relevant we will denote the coordinate system of a point by a leading superscript, for example the 3D point X in the world coordinate frame $\mathcal{W}$ will be represented as ${}^{\mathcal{W}}X$. Rotations can be expressed most generally by a 3×3 rotation matrix in the rotation group: $\mathbf{R} \in SO(3)$.

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

This representation has some nice properties:

$$\mathbf{R}^{-1} = \mathbf{R}^T det(\mathbf{R}) = 1$$

The rigid body, or 3D Euclidean transformation is one which translates point(s) from one coordinate system to another and is described by a transformation matrix of the form:

$${}^{\mathcal{WC}}\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where: (i) ${}^{\mathcal{WC}}\tilde{\mathbf{H}}$ denotes the transformation from the world coordinate frame $\mathcal{W}$ to the camera coordinate frame $\mathcal{C}$, (ii) $\mathbf{R}$ is the 3×3 rotation matrix and (iii) $\mathbf{t}$ is a 3×1 translation vector. The rigid body transform is used throughout the work in this these to describe the pose (position and orientation) of a MAV with respect to a global coordinate system as well as for transformations between coordinate frames. The rigid body transformation ${}^{\mathcal{WC}}\tilde{\mathbf{H}}$ is a member of the Lie group $SE(3)$, which allow the rigid body transformation to be minimally parametrised by a six dimensional vector μ via the exponential map. This representation has many desirable properties including being trivially differentiable. Full details of the Lie group SE(3) and its applications can be found in [116].

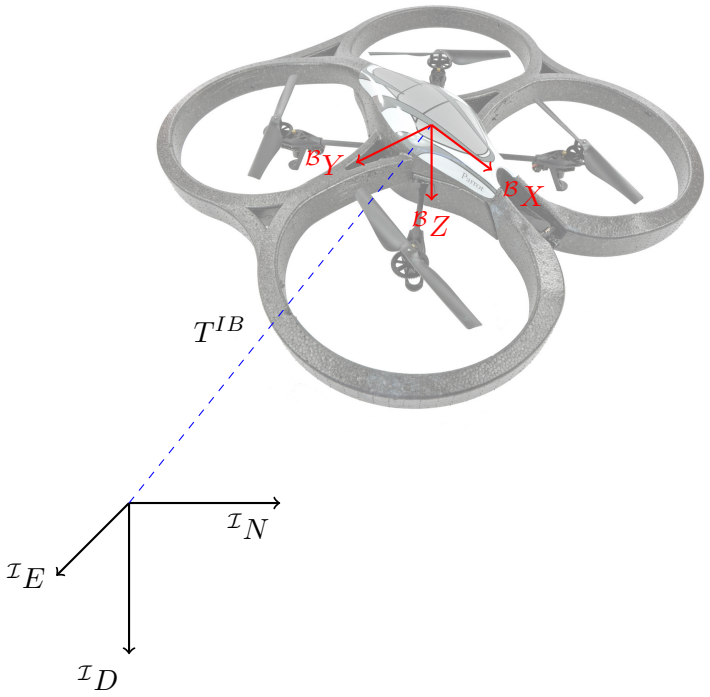


FIGURE 2.1: MAV Coordinate Frames.

Figure 2.1 shows the common coordinate frames for a MAV as used in this thesis. The inertial frame $\mathcal{I}$ is the earth fixed coordinate system with the origin defined as the starting or home location. The x-axis points north, the y-axis points east and the z-axis points into the earth. The coordinate system of a MAV is described by the body frame $\mathcal{B}$, where the origin is located at the centre of mass of the MAV. The x-axis points towards the front of the MAV, the y-axis points to the right of the MAV and the z axis points downward from the MAV. The transformation from the inertial frame to the body frame is given by the rigid body transform ${}^{\mathcal{IB}}\tilde{\mathbf{T}}$.

In some reported work, particularly that relating to MAV control, additional frames are introduced which separate the rotation and translation components of the transform ${}^{\mathcal{IB}}\tilde{\mathbf{T}}$. For example a vehicle frame $\mathcal{V}$ can be introduced where the origin is located at the centre of mass of the MAV but each axis is aligned with the inertial frame. This means the translation of the MAV with respect to the inertial frame can be expressed by the transform ${}^{\mathcal{IV}}\tilde{\mathbf{T}}$ and the rotational component of the transform can be expressed as a pure rotation from the vehicle frame to the body frame ${}^{\mathcal{VB}}\mathbf{R}$.

2.1.2 Similarity and Affine Transformations

The **Similarity** or **Scaling Transform** is part of the group containing similarity transforms in three dimensional space $Sim(3)$. It has the same representation as the rigid

body transformation with the addition of a scale factor:

$$\tilde{\mathbf{X}}' = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix} \tilde{\mathbf{X}}$$

The addition of the scaling factor to the similarity transform is useful, particularly for monocular SLAM where the scale is unknown or estimated using data from metric sensors. Optimisations such as loop closures, pose graph optimisation and bundle adjustment (see Section 2.4.5 can be done using the similarity transform instead of the standard rigid body transform. This allows us to take the scale ambiguity into account during optimisation [107].

The Affine Transformation goes a step further by adding a shearing factor along each axis: $\tilde{\mathbf{x}}' = A\tilde{\mathbf{x}}$ where:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

The affine transformation is useful to describe the transformation of points between images. In such cases, where image features are observed from different viewpoints, the rigid body or similarity transform is not sufficient to describe the transformation of the points.

2.2 Pinhole Camera Model

This Section focuses the foundational concepts of computer vision used for the work presented in this and how they relate to the problem of visual navigation. In particular this Section will cover how visual observations using cameras are modelled using the pinhole camera and distortion models. The section concludes with a discussion of geometric image features and feature matching.

The pinhole camera model describes the projection of 3D world features into the image plane of an ideal pinhole camera. It is used throughout computer vision to model the transformation from the 3D world of objects to the 2D world of images. This model illustrated in Figure 2.2 where the 3D world point $\tilde{\mathbf{P}}$ is projected onto the image plane of the camera at image point $\tilde{\mathbf{x}}$. The pinhole camera model is an example of the most general type of transformation, perspective projection. Given a 3D point in the world observed by a camera we often want to compute its corresponding 2D point in the image plane. The pinhole camera model can be described in matrix form by:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix}$$

Where: f_x and f_y define the focal length of the camera (in pixels) (this representation also implicitly encodes the aspect ratio of the image sensor), c_x and c_y define the optical

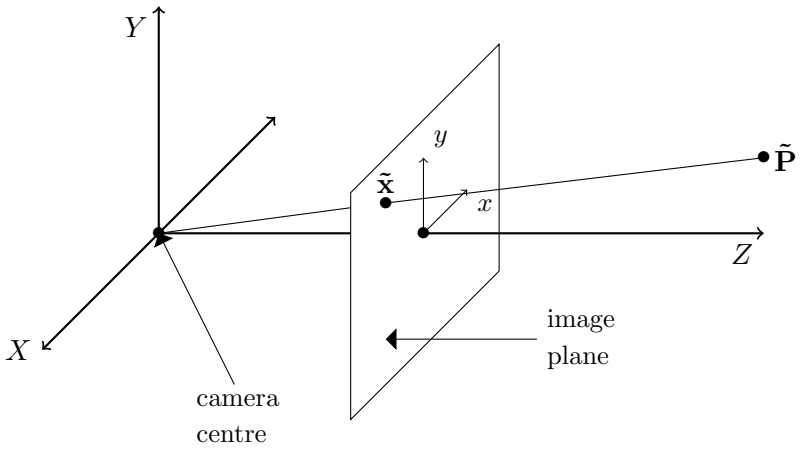


FIGURE 2.2: An illustration of the pinhole camera model. In reality the image plane is behind the camera centre and the true projection results in an image that is upside down and has to be rotated, this is done by the camera sensor so the generate images appear right way up.

centre (also called the principal point) and s defines the skew factor to account for the sensor not being mounted perpendicular to the optical axis [109]. The skew factor is often omitted as most image sensors do not induce any axis skew. These parameters can be provided by the manufacturer or more commonly obtained via calibration (see Section 2.4.1). This model assumes the 3D coordinates describe the point relative to the camera centre. If coordinates describe the transform relative to some other coordinate system e.g. ${}^{\mathcal{W}}\tilde{\mathbf{P}}$ in the world frame $\mathcal{W}$ the point must first be transformed to the camera coordinate system. This means we must know the camera transform in the world coordinate system, this is generally described as the extrinsic parameters of the camera:

$$\mathbf{M}_{ext} = \begin{bmatrix} {}^{\mathcal{WC}}\mathbf{R} & {}^{\mathcal{C}}\mathbf{t} \end{bmatrix}$$

Thus a 3×4 matrix describing the ${}^{\mathcal{WC}}\mathbf{R}$ world to camera rotation where ${}^{\mathcal{C}}\mathbf{t}$ is the world origin in camera coordinates. Therefore the full camera matrix containing both the intrinsic and extrinsic parameters of the camera is required to translate a point from world coordinates to $(u,\ v)$ pixel coordinates:

$$\tilde{\mathbf{x}} = \mathbf{K}\mathbf{M}_{ext}\ {}^{\mathcal{W}}\tilde{\mathbf{P}} \tag{2.1}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x/w \\ y/w \end{bmatrix} \tag{2.3}$$

2.2.1 Lens Distortion

The model above assumes no distortion occurs i.e. a perfectly shaped lens. However in practice this is not the case and imperfect lenses and sensor alignment can introduce significant distortion. This distortion results in image points appearing in significantly different positions than the expected projection given by the pinhole camera model described in the previous section. This is particularly noticeable when observing lines, straight lines in the world appear curved in the image. This effect is even more pronounced with wider angle lenses as shown in Figure 2.3. A common approach is to use a low order polynomial to model the radial distortion:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.4}$$

$$y_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.5}$$

Where $r^2 = x^2 + y^2$ accounts for the increasing distortion the further from the centre point and k_1, k_2 and k_3 are the radial distortion coefficients estimated via calibration. For wider angle lenses such as those used in this thesis it is more efficient to model the distortion using Devernay and Faugeras' Field of View (FOV) model[18]. In the FOV

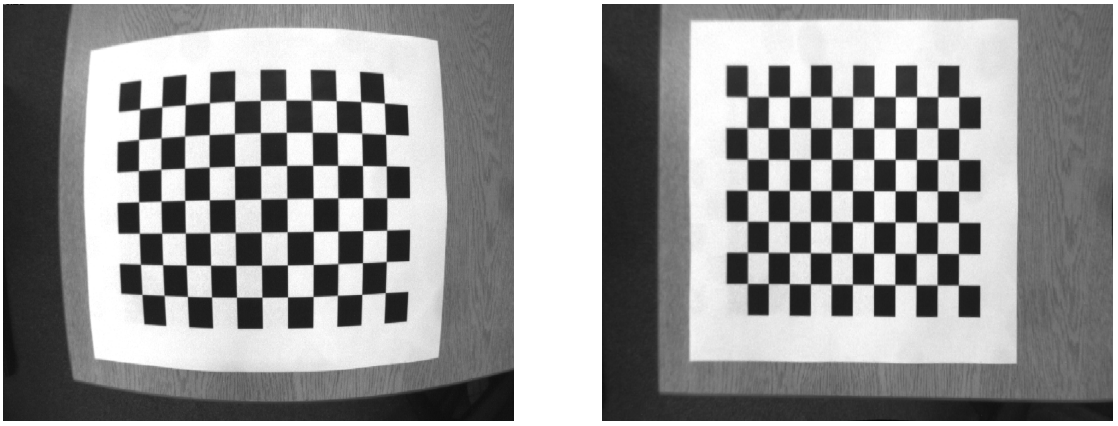


FIGURE 2.3: An example of the lens distortion resulting for a wide angle lens(left) and the same scene after camera calibration and image rectification(right).

model only a single parameter is used to describe the field of view of the *ideal* wide-angle lens. It is assumed that the distance between an image point and the principal point is proportional to the angle between the corresponding ray connecting the 3D point with the optical centre and the optical axis in the rectified image. The radial distortion function is given as:

$$r = \sqrt{\frac{x^2 + y^2}{z^2}} \tag{2.6}$$

$$r' = \frac{1}{w} arctan(2r\ tan\frac{w}{2}) \tag{2.7}$$

Where w is the field of view parameter obtained via calibration. Recovering the undistorted pixel coordinates can be done using the following transformation:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{r'}{r} \begin{bmatrix} x/w \\ y/w \end{bmatrix} \tag{2.8}$$

Some cameras also exhibit tangential distortion caused by misalignment between the lens and the image sensor, but it has been shown for the machine vision cameras used in this thesis that the effect is negligible [18].

2.3 Image Features

One of the fundamental problems in computer vision is feature detection and matching. Visual features are subsets of image data that describe unique or interesting regions within the image. Good features have desirable properties such as distinctiveness and repeatability. The ability to match unique features between two images allows a number of interesting applications such as image stitching, object tracking or, for the work in this thesis, case motion estimation. Features can describe geometric shapes such as lines and corners or less rigidly defined regions such as coloured/textured blobs. Corner points have been widely used in computer vision as interest points as they are highly constrained in both axes (as opposed to lines) and are therefore easier to match.

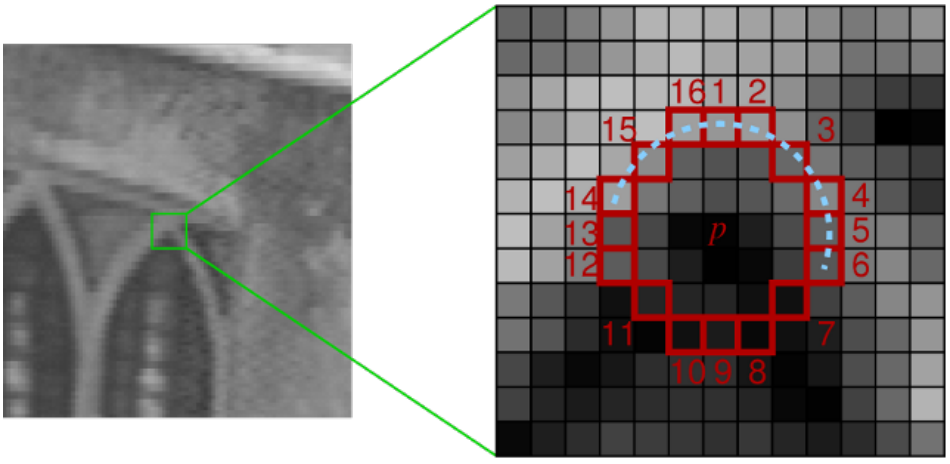


FIGURE 2.4: The segment test used in the FAST corner detector, here $r = 3$ and $n = 12$. The dashed arc highlights the 12 pixels brighter than p therefore p is a corner feature [97].

2.3.1 Corner Feature Detection

A feature detector is an algorithm which extracts visual features from images. A good feature detector is one that is both computationally efficient as well as reliable. For

real-time tracking applications such as visual SLAM, where feature extraction will be carried out on every image the importance of an efficient detector is vital. As such many tracking approaches use the Features from Accelerated Segment Test (FAST) detector.

The FAST feature detector, first presented in [96], uses the Segment-Test algorithm to determine if there is a corner feature at some pixel location p. The Segment Test is as follows, in a Bresenham [9] circle centred on pixel p and of radius r, if more than n pixels are either brighter or darker than pixel p by some threshold t then p is a corner feature see Figure 2.4. Adjusting the value of t determines the sensitivity of the corner detector, higher values of t results in a detector that finds fewer, but stronger, corners; a lower value increases the number of corners detected, but the resulting corners have smoother gradients which may affect repeatability.

The segment test can be accelerated as it provides a method to quickly reject pixels that definitely aren't corner features. In the example in Figure 2.4 where $n = 12$, the pixels at each compass point (i.e. 1, 9, 5 and 13) can be checked first; if the intensity of these pixels is close to p then p cannot be a corner feature. This begs the question, whatever the value of n what is the fewest number of pixel intensity tests we can do to determine if p is a corner feature and what is the best value for n. The value of n determines the maximum angle of the corner features detected while still rejecting edges (i.e. $n = 8$). Therefore $n = 9$ is the ideal value as it will detect corners with the largest variety of angles while still rejecting edges. In [97] Rosten demonstrated that a $n = 9$ detector is up to twice as fast as the original detector as well as being highly repeatable when compared to other state-of-the-art detectors.

The FAST-9 detector made use of machine learning to improve the performance of the segment test algorithm, for a given value of n the aim was to test the fewest number of pixels to determine if p is a corner feature or not. Rosten used the full Segment Test to extract corner features to create a training set for a decision tree classifier. The aim being to produce a decision tree which can determine if a pixel contains a corner point by testing the fewest number of pixels. Rosten was able to produce an $n = 9$ detector which only needed to test an average of 2.83 pixels to determine if a corner feature is present. The reliance on supervised learning to build the decision tree means for the best results the FAST-9 detector should be trained on features collected from the operating environment. This reduces the adaptability of the detector, an issue addressed by Mair et al. with their Adaptive Generic Accelerated Segment Test (AGAST) detector [73]. The AGAST detector uses a more generic binary decision tree instead of the learned ternary tree used in FAST-9. It also combines two decision trees, one optimised for homogeneous or cluttered image regions and the other optimised for uniform surfaces or high structured regions with texture. The AGAST detector switches trees based on the local structure of the image, making it adaptable to different environments without training. This adaptability is particularly useful in the application of Visual SLAM in outdoor scenes as the environment can change from highly cluttered scene (e.g. buildings and infrastructure) to uniform, highly textured regions such as fields or roads.

2.3.2 Corner Feature Selection

A potential issue with feature detection is the problem of features detected adjacent to one another. This can lead to issues when matching features from other views in that the adjacent features could be erroneously matched given their proximity. A common solution is non-maximum suppression, where features adjacent to one another are filtered by their relative intensity. This is done by defining a scoring function for feature points, $V(p)$ where V returns the sum of absolute difference between p and the 16 surrounding pixels. Scores of adjacent feature points are computed and the point with the lower score is discarded as illustrated in Figure 2.5. Another scoring function is the Shi Tomasi Scoring [103] function. This is based on the Harris Stevens [39] corner detector which uses the image gradients to detect corners. Specifically it computes a sum of squared difference matrix M over a small rectangular or Gaussian window function w

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{2.9}$$

Here I_x and I_y are the image gradients in x and y directions. This works well for axis-aligned corner features; to obtain rotation invariance the corner response is computed using the Eigenvalues of M:

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \tag{2.10}$$

Where R is a matrix of the eigenvectors and λ_1 and λ_2 are the eigenvalues of M. The Harris Stevens detector computes the corner response function R with a tuneable sensitivity parameter k as

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda 2)$$

Indeed Harris and Stevens presented a method to approximate the corner response without explicitly calculating the Eigenvalues which are more computationally intensive.

$$R = detM - k(traceM)^2$$

Where $detM$ is the determinant of the matrix M and is equivalent to the expression $\lambda_1 \lambda_2$ and $traceM$ is the trace of matrix M which is equivalent to $\lambda_1 + \lambda 2$. Shi and Tomasi noted that for reliable tracking, the features with the largest Eigenvalues that do not differ significantly are best [103]. These represent features with strong intensity profiles i.e. the intensity of the feature point differs to a larger extent than the mean intensity of the surrounding pixels. Given that the eigenvalues are bounded by the maximum intensity of the image it serves to take the smaller of the two to compute the corner score:

$$R = min(\lambda_1, \lambda_2) > \lambda \tag{2.11}$$

Where λ is a threshold parameter. An example is shown in Figure 2.6 where the image on the left shows the FAST corners detected in the image and the image on the right contains only those corners with a Shi Tomasi score above 50.

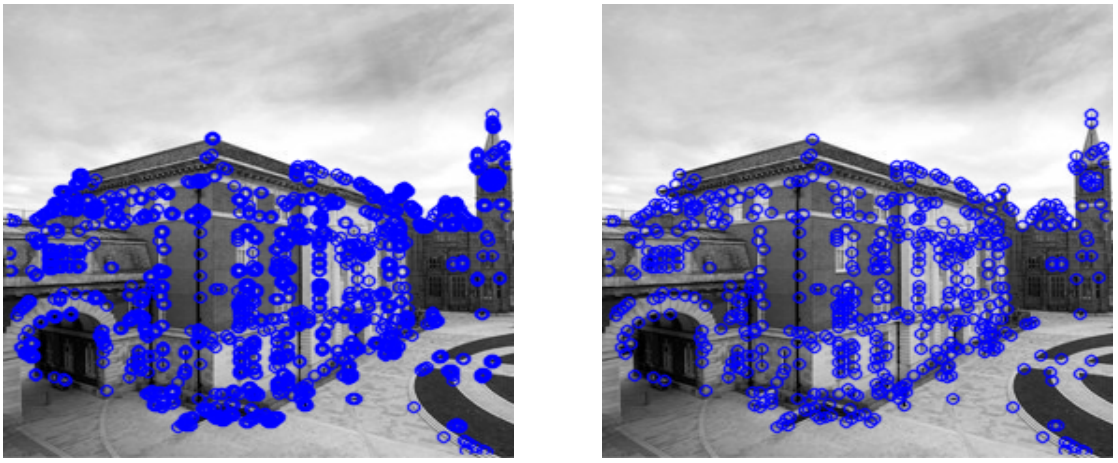


FIGURE 2.5: FAST corner features extracted from the image (left) and the corner features after non-maximum suppression(right).

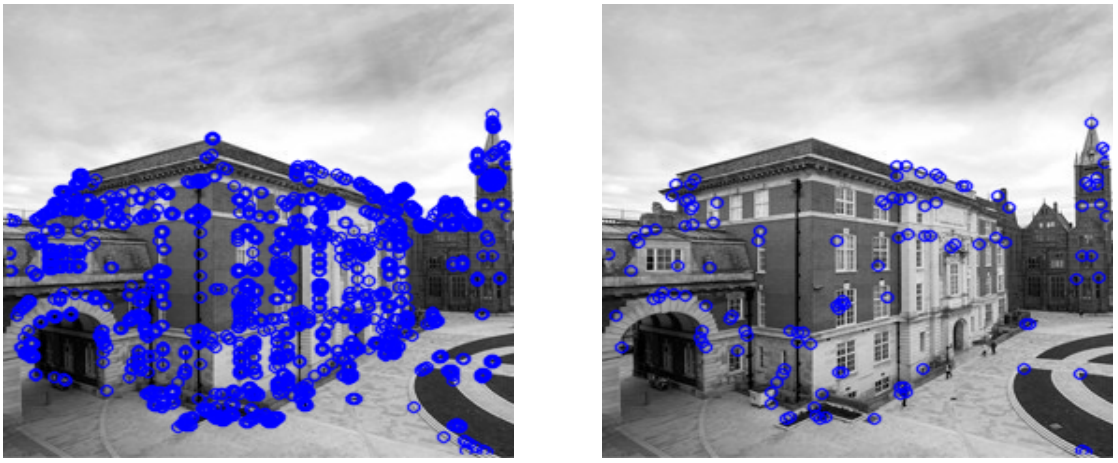


FIGURE 2.6: FAST corner features extracted from an image (left) and corners with a Shi-Tomasi score $>$ 50 (right).

2.3.3 Scale Invariant Feature Transform (SIFT)

Another interest point detector is SIFT[68]. SIFT is a combination of the Difference of Gaussian (DoG) feature detector and descriptor that uses gradient histograms. For the DoG detector the image is blurred using a Gaussian filter of increasing σ and computes the pairwise difference of the blurred images. Then for each pixel and a small window of neighbouring pixels it looks for the local extrema over all these Difference of Gaussian images, effectively searching for the scale at which the feature response is highest. The candidate points are filtered using non-maximum suppression, eliminating edges. Finally the precise centre point of the feature is computed at sub-pixel accuracy by fitting a quadratic function to the feature response function.

To achieve rotation invariance the dominant orientation(s) of the feature is computed using a histogram of local gradient directions. The highest peak and all peaks within a threshold of the highest peaks are selected. In the case of multiple peaks a feature point is generated for each dominant orientation. The feature descriptor is computed based on a 16×16 pixel region around the feature points. This is divided into 4×4 pixel regions on which an 8-bin orientation histogram is computed. This results in a descriptor with 128 elements. Their invariance properties make SIFT features a popular choice for applications such as image stitching and object recognition. For real-time tracking applications SIFT features are less popular due to the increased complexity involved in both SIFT feature detection and descriptor extraction. However in recent years, with the advances in parallel computation, using computer graphics processors it is possible to compute SIFT features in real-time [42].

2.3.4 Feature Matching

Feature detection is only one part of the problem in order to be useful we must be able to establish feature correspondences, that is, determine if feature x in image i is the same 3D world point as feature y in image j. The local appearance of a feature in an image is subject to change in transformation, rotation, scale and illumination making this one of the most challenging problems in computer vision. Given the fundamental nature of the problem there are many solutions, a complete description of which, is beyond the scope of this work. The discussion will thus be restricted to discussing methods for matching the features discussed in this thesis namely FAST and SIFT. One particular method commonly used in conjunction with FAST features is template matching. Template matching involves a direct comparison of the intensity values of a small, fixed size, region of pixels around a detected feature point. It is based on the assumption that the intensity values of the templates remain consistent between frames. This assumption is valid in real-time tracking applications where the motion between frames in small. We can then compare image patches using the Sum of Squared Differences (SSD):

$$SSD = \sum_{(u,v)\in W} (I_1(u,v) - I_2(x+u, y+v))^2 \tag{2.12}$$

Where an exact match of the intensity values will return a score of 0. A more robust but computationally more intensive approach is the Zero-Mean Sum of Squared Differences (ZSSD). Here the mean intensity value for each image patch is subtracted to improve robustness to lighting conditions:

$$ZSSD = \sum_{(u,v)\in W} ((I_1(u,v) - \overline{I_1}) - (I_2(x+u, y+v) - \overline{I_2}))^2 \tag{2.13}$$

Where $\overline{I}$ is the mean intensity of the pixel values in image patch I. This technique is often combined with an image pyramid to achieve a limited invariance to scale. An image pyramid is constructed by taking the source image applying a smoothing filter

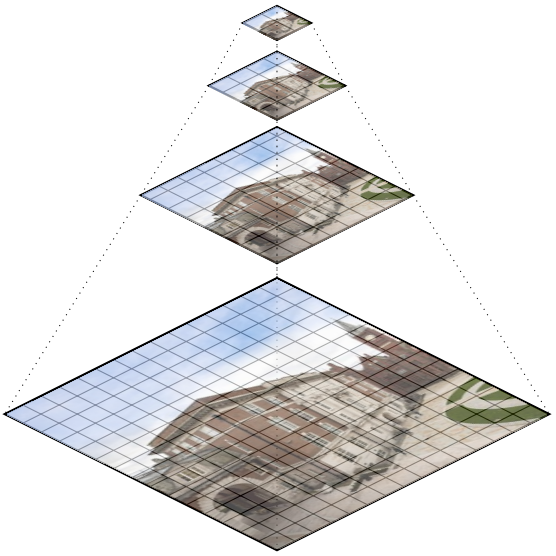


FIGURE 2.7: Image Pyramid

and sub sampling it by some factor λ along each axis. This multi-scale representation of the image allows us to detect features at different scale levels providing some invariance to scale. The scaling factor λ determines the number of pyramid levels $\lambda = 2$ results in a 4 level image pyramid (see Figure 2.7). This patch-based approach has the advantage of being computationally efficient, however it relies on the consistency of intensity values between frames. Where there is a large difference in viewpoint the local intensity of matching templates may be vastly different.

Matching SIFT keypoints (and descriptors in general) is usually done using a nearest neighbour approach where the similarity measure is the Euclidean distance in feature space:

$$d = [\sum_{i=1}^{128}(p_i - q_i)^2]^{1/2}$$

where p_i and q_i are the SIFT feature descriptors being compared. The naive approach is to set an absolute threshold for the feature distance, however this can result in false matches. Instead Lowe et al. [68] suggested that the most robust approach was the threshold between the first and second nearest neighbours. If both are close (as defined by the threshold) then the pair cannot be described as a strong match and should be discarded. This means only the strongest matches are used. This can be done in a brute force manner with in complexity of $O(m \cdot n)$ to reduce a feature set of size m to a set of size n. This makes real-time implementation infeasible but there are several approaches to reduce the complexity such as approximate nearest neighbour search [78] or the bag-of-words method [87].

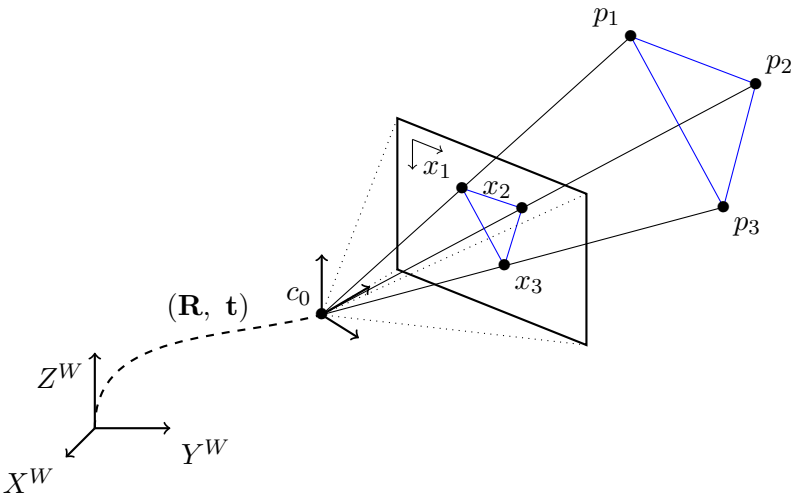


FIGURE 2.8: Perspective n-Point Problem

2.4 Structure from Motion

Recovering 3D information from a 2D camera sensor is a well known problem in computer vision, the key is exploiting either the motion of the object being observed or the motion of the camera itself to recover the missing depth information. Structure from Motion (SfM) has been extensively studied in the case of offline 3D reconstruction, however in recent years the techniques developed have been applied to real-time problems in robotics. In this section we will introduce some key problems in SfM and describe some common methods to solve them. In particular we will look at the problems of estimating the pose of a camera when observing a known shape, recovering depth information from two distinct views of the same point and estimating the relative pose between two cameras. We will conclude with a discussion of bundle adjustment, a powerful technique to iteratively refine both structure (3D points) and motion (camera poses) estimates.

2.4.1 Perspective n-Point Problem

The Perspective n-Point problem addresses the issue of estimating camera poses (and camera parameters) from observations of known object/points. Here we have a set of n 3D points together with their corresponding 2D positions in the camera frame, see Figure 2.8. Given the set of 2D-3D point correspondences $(\tilde{\mathbf{x}}_i, \ldots, \tilde{\mathbf{x}}_n) \leftrightarrow (\tilde{\mathbf{P}}_i, \ldots, \tilde{\mathbf{P}}_n)$ we want to estimate

$$C = K(\mathbf{R}, \mathbf{t})$$

where K is the intrinsic camera parameter matrix and $\tilde{\mathbf{x}}_i = C\tilde{\mathbf{P}}_i$ i.e. the re-projection of the set of 3D points $\tilde{\mathbf{P}}$ into image coordinates corresponds to the set observed features points $\tilde{\mathbf{x}}$. While inherently non-linear this problem can be solved in a linear fashion if we make the assumption that the unknown variables are independent. We can then define a set of equations describing each 2D-3D point correspondence with respect to

the matrix C.

$$x = \frac{c_{11}X + c_{12}Y + c_{13}Z + c_{14}W}{c_{31}X + c_{32}Y + c_{33}Z + c_{34}W} \tag{2.14}$$

$$y = \frac{c_{21}X + c_{22}Y + c_{23}Z + c_{24}W}{c_{31}X + c_{32}Y + c_{33}Z + c_{34}W} \tag{2.15}$$

Where: (i) x, y are the 2D image coordinates of feature point x_i, (ii) (X, Y, Z, W) are the 3D homogeneous coordinates of 3D point p_i and (iii) c_{12} denotes the element in the 1st row and second column of the matrix C. Multiplying each equation by the denominator gives the following set of linear equations:

$$0 = (x_j c_{31} - c_{11})X + (x_j c_{32} - c_{12})Y + (x_j c_{33} - c_{13})Z + (x_j c_{34} - c_{14})W \tag{2.16}$$

$$0 = (y_j c_{31} - c_{21})X + (y_j c_{32} - c_{22})Y + (y_j c_{33} - c_{23})Z + (y_j c_{34} - c_{24})W \tag{2.17}$$

These can then be rearranged into a homogeneous linear equation of the from $Ax = 0$ where A is the matrix form of the system of linear equations and x is a column vector representing the unknowns (i.e. the camera matrix C).

$$\begin{bmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -x \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -yX & -yY & -yZ & -y \end{bmatrix} x = 0 \tag{2.18}$$

Where $x = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{34} \end{bmatrix}^T \in \mathbb{R}^{12}$ representing the set of unknown values. Given that C has 12 unknowns at least 6 correspondences are required to produce a unique solution. This can be solved using Singular Value Decomposition (SVD) where x is the Eigenvector corresponding to the smallest Eigenvalue of the matrix. Once C has been obtained we can recover both the intrinsic and extrinsic camera parameters by factoring C into $K\mathbf{R}$ using QR decomposition. We can then compute $t = K^{-}1(c_{14}, c_{24}, c_{34})^T$. For increased accuracy the parameters of C can be further refined using least squares (which will be described in detail later in this Chapter); finding the parameters of C that minimise the re-projection error for all points based on the initial solution. Where points are affected by noise or bad correspondences a Random Sampling and Consensus (RANSAC) based approach can be used. The general RANSAC algorithm is as follows:

Algorithm 1 RANSAC

1: Randomly select a subset of dataset
2: Find the set of parameters that fits the selected data points
3: Compute the number of outliers for the complete dataset based on the estimated parameters
4: Repeat for fixed number of iterations or until the number of inliers reaches a given threshold

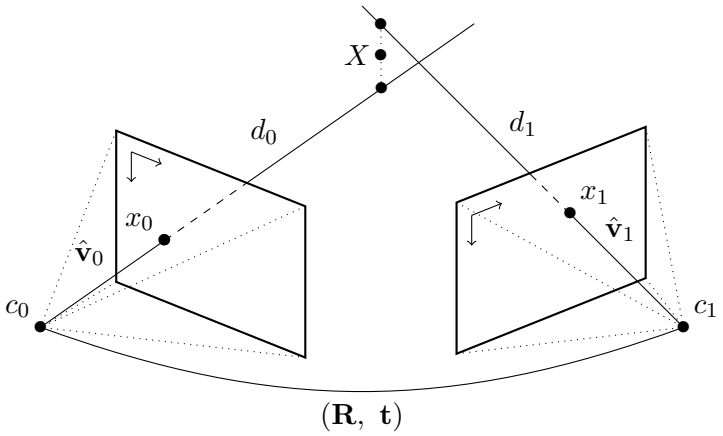


FIGURE 2.9: Triangulation

Given that we can recover the intrinsic camera parameters from the solution to the PnP problem one of the most widely used applications is that of camera calibration. This is achieved using a known calibration pattern and PnP to obtain the intrinsic parameters for a specific camera. A common calibration pattern is the checkerboard shown in Figure 2.3. The typical procedure is as follows:

1. Threshold the image
2. Detect edges and fit lines
3. Intersect lines to obtain corner points
4. Estimate camera matrix C from obtained corner points
5. Extract K from C using QR decomposition.

Another common application is assuming a calibrated camera (i.e. K is known) is camera pose estimation using fiducial markers[33, 43]. Here markers consisting of

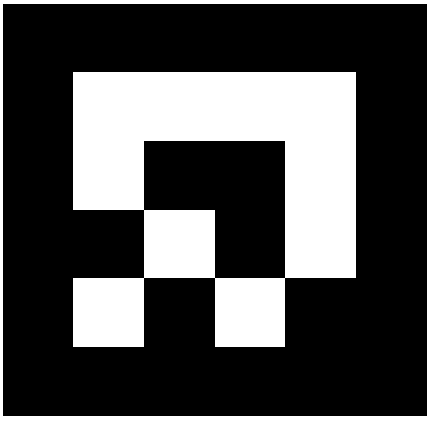


FIGURE 2.10: An example of a fiducial marker

known shape and dimensions are used, an example marker is shown in figure 2.10. The

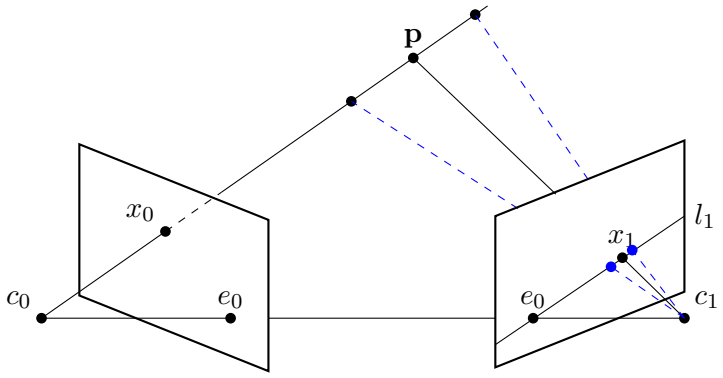


FIGURE 2.11: Epipolar Geometry

use of high contrast colour scheme and known, fixed shape make these markers trivial to detect with standard thresholding techniques in computer vision [126]. The arrangement of squares represent a binary hamming code which encodes both the identifier of the marker as well as its orientation. This allows the markers to be differentiated from each other as well as other objects of similar shape. This provides a very fast and accurate method for camera pose estimation at the cost of requiring prior knowledge of the marker shape and dimensions. Fiducial markers is the context of visual navigation will be discussed further in Chapter 3.

2.4.2 Triangulation

Triangulation can be seen as the converse of the PnP problem and the problem was first described by Longuet-Higgins [66]. Here we have two observations x_0 and x_1 of the same 3D world feature X. Additionally we know the full camera matrix for both cameras i.e both the the relative transformation between the two cameras $(R, \mathbf{t})$ is known as well as the intrinsic camera parameters K. The unknown we want to recover is the corresponding 3D point X. This can be solved using the same system of linear equations (equations 2.14 and 2.15) as before however in this case the unknowns are the X, Y, Z coordinates of the 3D point X. As before we can solve this system of equations using SVD and again increased accuracy can be obtained by taking a least squares solution.

2.4.3 Epipolar Geometry

In order to triangulate a 3D point from 2D point correspondences we need to first establish the point correspondence between to images. Searching for point correspondences between images can be difficult as locally many points look similar and depending upon the type of feature points and descriptors used matches can be ambiguous. However we can use the known geometry of the cameras to limit the search for point correspondences to a search along a single line rather than the entire image. This can be done by exploiting the geometry between two cameras of which the most important is the epipolar constraint. This describes the relationship of point correspondences between

two viewpoints i.e. stereo vision. First let's define an alternate representation for image coordinates, namely normalised image coordinates. If we know the intrinsic camera parameters we convert image coordinates to a normalised form where the effective focal length is 1 and the camera centre is in the middle of the image. This means we no longer require the intrinsic camera matrix to describe the projection of points into the image plane. The transformation from un-normalised coordinates to normalised is given by:

$$\mathbf{p} = K^{-1}\mathbf{u}$$

where u are the un-normalised image coordinates and K^{-1} is the inverse intrinsic parameter matrix. Consider Figure 2.11 where we observe a 3D world point from two separate views. The line connecting the centres of both cameras c_0 and c_1 is known as the **baseline**. The plane defined by the points $c_0,\ c_1,\ \mathbf{p}$ is known as the epipolar plane. The intersection between the image plane and the epipolar plane is known as the epipolar line $l1$. For each pair of corresponding normalised image points $\mathbf{x}_1, \mathbf{x}_2$ the following constraint holds:

$$\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0$$

where $\mathbf{E}$ is the essential matrix [66] which describes the transformation between the two cameras:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} \in \mathbb{R}^{3\times 3}$$

Here $[\mathbf{t}]_\times$ is the skew symmetric matrix of the form:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

Representing the matrix form of the cross product multiplication of $\mathbf{t}$ and $\mathbf{R}$. The epipolar constraint means that the corresponding feature point in the second image x_1 is guaranteed to be found along the epipolar line this limits the search space for point correspondences to a single line. The epipolar constraint is also useful for estimating the relative motion between two camera where the set of point correspondences is known. As the essential matrix is a 3×3 it appears that there are 9 unknowns, however we can arbitrarily scale $\mathbf{E}$ the essential matrix and still satisfy the epipolar constraint, meaning we can only compute $\mathbf{E}$ to scale. This leaves 8 unknowns meaning we can compute $\mathbf{E}$ from 8 or more point correspondences using the 8-point linear algorithm. Given a set of 8 or more point correspondences we can define a set of 8 equations of the form $\mathbf{p}_0^T \mathbf{E} \mathbf{p}_1 = 0$

$$\begin{bmatrix} x_0 & y_0 & 1 \end{bmatrix} \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = 0 \tag{2.19}$$

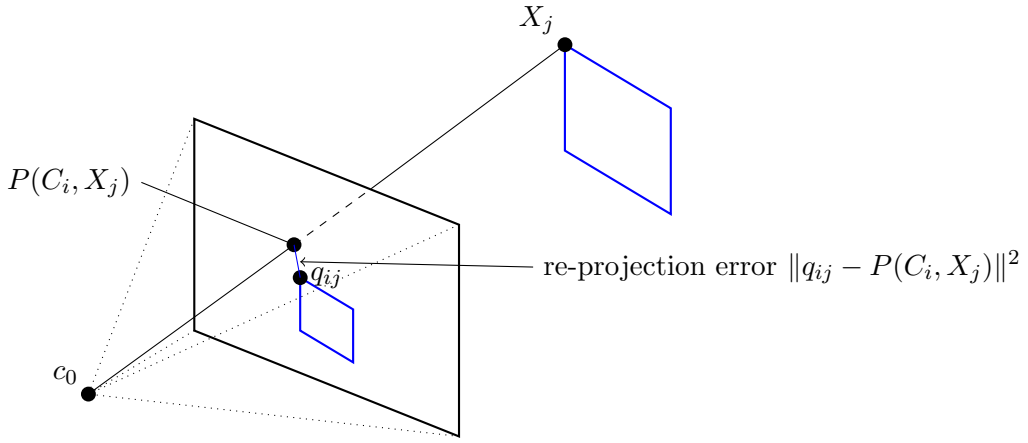


FIGURE 2.12: Re-Projection Error

We can rewrite this as:

$$E_{11}x_0x_1 + E_{12}x_0y_1 + E_{13}x_0 + E_{21}y_0x_1 + E_{22}y_0y_1 + E_{13}y_0 + E_{31}x_1 + E_{32}y_1 + E_{33} = 0 \tag{2.20}$$

We can then reformulate this as a set of homogeneous linear equations of the form $Ax = 0$ where x is the vector of unknown values from the essential matrix and A consist of a row for each point correspondence:

$$\begin{bmatrix} x_0x_1 & x_0y_1 & x_0 & y_0x_1 & y_0y_1 & y_0 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_ix_j & x_iy_j & x_i & y_ix_j & y_iy_j & y_i & x_j & y_j & 1 \end{bmatrix} \begin{bmatrix} E_{11} \\ E_{12} \\ E_{13} \\ \vdots \\ E_{33} \end{bmatrix} \tag{2.21}$$

As before we find the solution for x can be found in the null space of A. That is using SVD to decompose $A = UDV^T$, where the solution x is the rightmost column of V corresponding to the only null singular value of A. The 8-point linear algorithm has been succeeded by many solutions, notably Nisters 5-point algorithm which estimates $\mathbf{E}$ from only 5 point correspondences [86]. Of particular interest to Visual SLAM on MAVs is the work of Fraundorfer et al. who demonstrated a 3-point algorithm [32]. They assume a platform equipped with both camera and IMU and thus the orientation angles of the two views are known leaving only the three unknowns of relative transformation.

2.4.4 Least Squares

Least squares in the general case is a parameter estimation approach to find an approximate solution to an overdetermined system of equations. This is done by minimising the sum of squared errors for each equation. In many cases there is no exact solution to an overdetermined system of equations in such a case least squares can be used to

find the closest solution to the exact one. Least squares is particularly useful in the context of fitting a model to a set of noisy observations. For example given a column vector of independent data points $x = (x_1, \ldots, x_n)^T$ and vector of dependent measurements $y = (y_1, \ldots, y_n)^T$ the aim is to find a set of model parameters $p = (p_1, \ldots, p_n)^T$ which represent the best fit between the model and the measured data points. Best fit is defined as the set of parameters at which the sum S of the residuals is minimal:

$$S = \sum_{i=1}^{n} \|\mathbf{r}_i\|^2 \tag{2.22}$$

where the residuals $\mathbf{r}_i$ is the difference between the measured value and the value predicted by the model $\mathbf{r}_i = y_i - f(x_i, p_i)$. The form of the function f determines the nature of the least squares problem, that is linear vs non-linear least squares. In the case of computer vision and structure from motion problems, we commonly use the re-projection error as the residual in least squares problems. The re-projection error is the difference between the pixel location of an observed 3D point and its estimated position based on the camera pose and the points 3D position (see Figure 2.12). In this case the residual function f is the projection function based on the pinhole camera model with lens distortion (introduced in Section 2.2). As this is a non-linear function a non-linear least squares approach is required. As we have already mentioned non-linear least squares can be applied to the PnP and Triangulation problems to produce more accurate results, a more general problem is that of Bundle Adjustment which we consider in the next section.

2.4.5 Bundle Adjustment

This section provides a brief introduction to the bundle adjustment problem, for a more detailed account the reader is directed the excellent work of Triggs et al. [114]. The bundle adjustment problem can be seen as a combination of the triangulation problem and the PnP problem. We have a sequence of camera images from which we use 2D image measurements to extract both the 3D structure of the scene as well as the trajectory of the cameras. In the case of Visual SLAM we have existing estimates of both the 3D structure and camera trajectory which are estimated incrementally. The initial estimates are affected by both noise (assumed to normally distributed) as well as accumulated error from the incremental motion estimation. In bundle adjustment we seek to jointly refine the 3D structure and camera trajectory. More formally given a map consisting of n 3D map-points $\tilde{\mathbf{X}}_j$ and m camera matrices $\mathbf{M}_i$ representing m images take of a scene. In addition, for each map-point we have a set of 2D image measurements where measurement x_{ij} is an observation of map-point $\tilde{\mathbf{X}}_j$ in the camera image represented by camera matrix $\mathbf{M}_i$. We can then define the bundle adjustment in terms of a non-linear least squares problem where we seek to minimise the cost function, defined as the

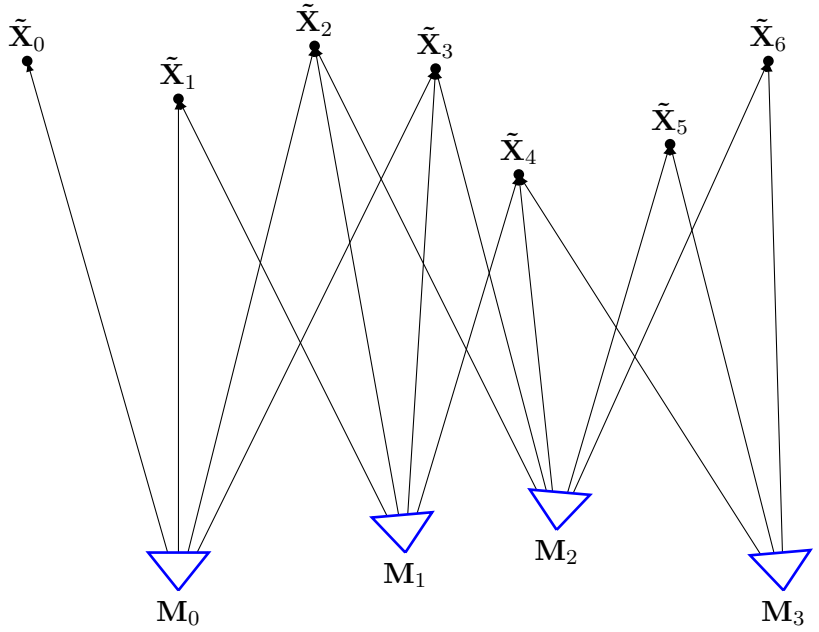


FIGURE 2.13: Bundle Adjustment Problem

re-projection error between observed and predicted image points:

$$\min \sum_{j=1}^{n} \sum_{i=1}^{m} \|h(\mathbf{M}_i, \tilde{\mathbf{X}}_j) - x_{ij}\|^2 \tag{2.23}$$

The name bundle adjustment comes from the representation of the problem shown in Figure 2.13 in which the lines between cameras and the observed points represent the "bundles" to be incrementally adjusted till an improved solution is found. Algorithms such a Gauss-Newton and Levenberg-Marquardt (LM) have been successfully applied to such problems [40, 72, 109]. The typical approach for the non-linear case is to first linearise the problem using a first-order Taylor expansion around the current solution, compute an adjustment based on this linearisation, apply it and repeat until a convergence is reached. The Gauss-Newton algorithm works well when the initial solution is close to optimal whereas LM is more effective when the quality of the initial solution cannot be guaranteed; for example where linear solutions such as the 8-point algorithm were used to determine the initial solution. In the next section we explain the LM algorithm in more detail.

2.4.6 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (LM) algorithm is an effective solution to non-linear least squares optimisation problems particularly in the case where the quality of the initial solution may not be close to optimal. The algorithm was developed first by Levenberg [65] and again by Marquardt [74]. In this section we provide a brief introduction the LM algorithm, more detail can be found in [40, 72, 109]. LM provides this flexibility

by a combination of the gradient descent and Gauss-Newton. The behaviour of the LM algorithm is as follows: far from the local minimum LM uses a gradient descent approach which is slower than Gauss-Newton but guaranteed to converge. Close the the local minimum the LM algorithm switches to Gauss-Newton in order to speed up convergence. The general form of a least squares problem we have a function f which describes the relation between a parameter vector $\mathbf{p} \in \mathbb{R}^m$ and a measurement vector $\hat{\mathbf{x}} \in \mathbb{R}^n$; that is $\hat{\mathbf{x}} = f(\mathbf{p})$. An initial parameter estimate $\mathbf{p}_0$ and a measurement vector $\mathbf{x}$ are provided and the aim is to iteratively refine the parameter vector $\mathbf{p}$ such that it minimises the squared distance $\epsilon^T\epsilon$, where $\epsilon = \mathbf{x} - \hat{\mathbf{x}}$. An assumption is made that f is locally linear; using the initial parameter estimate $\mathbf{p}_0$ as a starting point the algorithm generates an incremental update $\delta_\mathbf{p}$ to iteratively refine $\mathbf{p}$. To determine the update $\delta_\mathbf{p}$ the function f is approximated by $f(\mathbf{p}+\delta_\mathbf{p}) \approx f(\mathbf{p}) + J\delta_\mathbf{p}$ where J is the Jacobian of f, $J = \partial f/\partial p$ that is all the partial derivatives of f with respect to $\mathbf{p}$. Starting with the initial parameter estimate $\mathbf{p}_0$ the algorithm produces a sequence of parameter vectors $\mathbf{p}_1, \mathbf{p}_2, \ldots$ until it converges towards a local minimum for $\mathbf{p}$. Therefore at each iteration it is necessary to compute a $\delta_\mathbf{p}$ that minimises the following:

$$\|\mathbf{x} - f(\mathbf{p} + \delta_\mathbf{p}\| \approx \|\mathbf{x} - f(\mathbf{p}) - J\delta_\mathbf{p}\| = \|\epsilon - J\delta_\mathbf{p}\| \tag{2.24}$$

This $\delta_\mathbf{p}$ is then the solution to a linear least squares problem $\|\epsilon - J\delta_\mathbf{p}\|$ over $J\delta_\mathbf{p}$ which can be solved using the so-called normal equations:

$$J^T J\delta_\mathbf{p} = J^T\epsilon \tag{2.25}$$

The matrix $J^T J$ is the first order approximation of the Hessian of $\frac{1}{2}\epsilon^T\epsilon$ and $\delta_\mathbf{p}$ is the Gauss-Newton step. The gradient descent (sometimes called the steepest descent) of $\epsilon^T\epsilon$ corresponds to $-J^T\epsilon$. The LM uses an augmented version of the normal equations 2.25 where a damping factor λ is added (where $\lambda > 0$):

$$(J^T J + \lambda I)\delta_\mathbf{p} = J^T e \tag{2.26}$$

The damping factor λ is adjusted at each iteration based on the outcome of equation 2.26, if the outcome results in a reduction in the error term $\epsilon^T\epsilon$ then the value of λ is decreased for the next iteration; if the outcome results in increased error then λ adjusted until a solution is obtained that results in a reduction of the error. This means that it may require several computations of equation 2.26 before an adjustment $\delta_\mathbf{p}$ that reduces error is found and accepted. As the Gauss-Newton method uses an approximation of the Hessian of the residuals. This approximation method is valid only in cases where the error in small (i.e. the current set of parameters are close to the local minimum) or the residual function is close to linear. If either assumption does not hold the method is not guaranteed to converge. This is the motivation for the inclusion of the damping factor in the LM algorithm. When the value of λ is large the adjustment is skewed towards the steepest descent (i.e. $J^T\epsilon$) this results in smaller steps (i.e. more iterations before

convergence) but is guaranteed to reach a local minimum. If λ is initially set to a large value the algorithm will follow the steepest descent method initially, but as λ is reduced for each solution that reduces error as the solution approaches the local minimum the Gauss-Newton term dominates the adjustment resulting in faster convergence to the local minimum. This means the LM algorithm converges quickly to a minimum when the initial parameters are close to the solution but also provides some robustness (via the gradient descent method) in cases where the initial solution may not close to the minimum. It is this adaptive behaviour that makes the LM algorithm one of the most widely used in the literature for solving non-linear least squares optimisation in computer vision.

The termination criteria for the LM algorithm are typically one or many of the following:

- a maximum number of iterations has been reached.
- the magnitude of the residual is low than a threshold.
- the relative magnitude of $\delta_{\mathbf{p}}$ is lower than a threshold.
- the magnitude of the gradient falls below a threshold.
- the relative reduction if the magnitude of the residual falls below a threshold.

Typically in computer vision applications we have an estimate of the uncertainty of the measurement vector $\mathbf{x}$ in the form of the covariance matrix $\Sigma_{\mathbf{x}}$. This modifies the least squares problem to a *weighted* least squares problem. Incorporating this information into the LM algorithm can be done in the following way: the Euclidean norm error term $\epsilon^T\epsilon$ is replaced by the squared Mahalanobis distance $\epsilon^T\Sigma_{\mathbf{x}}^{-1}\epsilon$. The goal is then to minimise the squared norm $\Sigma_{\mathbf{x}}^{-1}$. The augmented normal equation 2.26 is then rewritten as:

$$(J^T\Sigma_{\mathbf{x}}^{-1}J + \lambda I)\delta_{\mathbf{p}} = J^T\Sigma_{\mathbf{x}}^{-1}\epsilon \tag{2.27}$$

2.4.7 Sparse Bundle Adjustment

A typical bundle adjustment problem can be very large in terms of the number of parameters. As discussed previously we have a map consisting of 3D map-points and camera matrices. Each map-point consists of 3 parameters (for euclidean coordinates) and each camera matrix consists of 11 parameters (6 parameters for translation and rotation and 5 intrinsic camera parameters). Given that a bundle adjustment problem may consist of several hundred camera matrices and several thousand map-points this can result in a non-linear least squares problem with hundreds of thousands of parameters. The computational complexity of the bundle adjustment problem is cubic in the number of parameters which makes directly solving the augmented normal equations computational infeasible for sufficiently large problems. To resolve this issue modern approaches take

advantage of the sparsity of the bundle adjustment problem. This sparsity comes from the fact that not every map-point is observed in every image. This results in a sparse block structure of the Jacobian matrices which can be exploited to speed up computation. More formally we define a vector $\mathbf{a}_j$ which represent the parameters for camera j and vector $\mathbf{b}_i$ which represent the parameters for 3D point i. And we define the bundle adjustment problem in terms of minimising the error between the measurement vector $\mathbf{x}$ where $\mathbf{x}_{ij}$ represents the measurement of point i in camera j:

$$\arg\min_{\mathbf{a}_j,\mathbf{b}_i} \sum_{i=1}^{n}\sum_{j=1}^{m} d(\mathbf{Q}(\mathbf{a}_j,\mathbf{b}_i),\mathbf{x}_{ij})^2 \tag{2.28}$$

Here we introduce a reformulation of the re-projection error where $d(\mathbf{x},\mathbf{y})$ is a function which returns the Euclidean distance between the two image points $\mathbf{x}$ and $\mathbf{y}$ and $\mathbf{Q}(\mathbf{a}_j,\mathbf{b}_i)$ is the projection of 3D point i onto the image plane of camera j. If we define the dimensions of the point and camera parameters $\mathbf{a}_j$ and $\mathbf{b}_i$ as α and β respectively then the general formula for the number of parameters in a given problem can be given as $n\alpha + m\beta$.

The parameter vector for this non-linear least squares problem consist of the parameters the n camera camera matrices and m feature points which can be composed as a parameter vector of the form: $\mathbf{P} = (\mathbf{a}_1^T,\ldots,\mathbf{a}_m^T,\mathbf{b}_1^T,\ldots,\mathbf{b}_n^T)^T$ we can similarly define the measurement vector $\mathbf{X}$ as the measured image coordinates for all map-points points in all cameras:

$$\mathbf{X} = (\mathbf{x}_{11}^T,\ldots,\mathbf{x}_{1m}^T,\mathbf{x}_{21}^T,\ldots,\mathbf{x}_{2m}^T,\ldots,\mathbf{x}_{n1}^T,\ldots,\mathbf{x}_{nm}^T)^T \tag{2.29}$$

We define a covariance matrix $\Sigma_{\mathbf{X}}$ which represents the uncertainty of the measurement vector $\mathbf{X}$, which can be set to the identity matrix for situations where no uncertainty information exists. If $\mathbf{P}_0$ is the initial parameter vector we can generate an estimated measurement vector $\hat{\mathbf{X}}$ which represents the vector estimated feature points $\hat{\mathbf{X}} = f(\mathbf{P})$. The estimated measurement vector has the form:

$$\hat{\mathbf{X}} = (\hat{\mathbf{x}}_{11}^T,\ldots,\hat{\mathbf{x}}_{1m}^T,\hat{\mathbf{x}}_{21}^T,\ldots,\hat{\mathbf{x}}_{2m}^T,\ldots,\hat{\mathbf{x}}_{n1}^T,\ldots,\hat{\mathbf{x}}_{nm}^T)^T \tag{2.30}$$

where $\hat{\mathbf{x}}_{ij} = \mathbf{Q}(\mathbf{a}_j,\mathbf{b}_i)$. Following our description of the LM algorithm in the previous section we aim to solve this non-linear least squares problem by minimising the squared $\Sigma_{\mathbf{X}}^{-1}$ norm (that is the Mahalanobis distance) $\epsilon^T\Sigma_{\mathbf{X}}^{-1}\epsilon$, where *epsilon* is now the vector representing the difference between the measurement vector $\mathbf{X}$ and the estimated measurement vector $\hat{\mathbf{X}}$ i.e. $\epsilon = \mathbf{X} - \hat{\mathbf{X}}$ with respect to the current parameter vector $\mathbf{P}$. As per the LM algorithm this can be solved by iteratively solving the augmented weighted normal equation:

$$(J^T\Sigma_{\mathbf{x}}^{-1}J + \lambda I)\delta_{\mathbf{p}} = J^T\Sigma_{\mathbf{x}}^{-1}\epsilon \tag{2.31}$$

where J is the Jacobian matrix of the function f and δ is the update to be applied to the parameter vector $\mathbf{P}$. A key property of the structure of the bundle adjustment problem is the interdependence between feature points and cameras. This leads to a sparse block structure in the normal equations. More formally as image point $\hat{\mathbf{x}}_{ij}$ is dependent only on the parameters of the k-th camera that means in the Jacobian $\partial\hat{\mathbf{x}}_{ij}/\partial\mathbf{a}_k = 0$, $\forall j \neq k$ and $\partial\hat{\mathbf{x}}_{ij}/\partial\mathbf{b}_k = 0$, $\forall i \neq k$. Let $\mathbf{A_{ij}}$ be the partial derivative of point $\hat{\mathbf{x}}_{ij}$ with respect to camera $\mathbf{a}_i$ that is $\mathbf{A_{ij}} = \partial\hat{\mathbf{x}}_{ij}/\partial\mathbf{a}_i$. Similarly let $\mathbf{B_{ij}}$ be the partial derivative of point $\hat{\mathbf{x}}_{ij}$ with respect to map-point $\mathbf{b}_i$ that is $\mathbf{B_{ij}} = \partial\hat{\mathbf{x}}_{ij}/\partial\mathbf{b}_i$. Following the structure of the parameter vector P the Jacobian matrix for a simple example with 3 cameras and 3 map-points would be as follows:

$$J = \frac{\partial \mathbf{X}}{\partial \mathbf{P}} = \begin{bmatrix} \mathbf{A}_{11} & 0 & 0 & \mathbf{B}_{11} & 0 & 0 \\ 0 & \mathbf{A}_{12} & 0 & \mathbf{B}_{12} & 0 & 0 \\ 0 & 0 & \mathbf{A}_{13} & \mathbf{B}_{13} & 0 & 0 \\ \mathbf{A}_{21} & 0 & 0 & 0 & \mathbf{B}_{21} & 0 \\ 0 & \mathbf{A}_{22} & 0 & 0 & \mathbf{B}_{23} & 0 \\ 0 & 0 & \mathbf{A}_{23} & 0 & \mathbf{B}_{23} & 0 \\ \mathbf{A}_{31} & 0 & 0 & 0 & 0 & \mathbf{B}_{31} \\ 0 & \mathbf{A}_{32} & 0 & 0 & 0 & \mathbf{B}_{32} \\ 0 & 0 & \mathbf{A}_{33} & 0 & 0 & \mathbf{B}_{33} \end{bmatrix} \tag{2.32}$$

The 0 values in the matrix address the case where a point is not visible in an image and thus the measurements are given 0 weight. Note how the ordering of the parameter vector P also nicely partitions the camera and point parameters. This sparse block structure can be exploited to reformulate the augmented normal equations (equation 2.31, the full derivation can be found in [67]. The sparse structure and ordering can be exploited to improve the performance of the bundle adjustment algorithm. Using the re-formulation described Lourakis et al. [67] the dependency between camera and point parameters is removed. This means the update $\delta_{\mathbf{P}}$ can be solved separately for $\delta_{\mathbf{a}}$ and $\delta_{\mathbf{b}}$. The re-formulated equations for $\delta_{\mathbf{a}}$ result in symmetric positive-definite matrix which can be solved efficiently using Cholesky factorisation [67]. Once the solution for $\delta_{\mathbf{a}}$ has been found the update for $\delta_{\mathbf{b}}$ can be solved by back substitution. Solving for $\delta_{\mathbf{a}}$ first is justified by the fact that a typical bundle adjustment problem contains far fewer cameras than map-points. This partitioning of the problem and the resulting reformulation leads to significant speed up in the computation time for each iteration of the LM algorithm.

2.4.8 Robust Bundle Adjustment

One potential drawback to using a least-squares scheme is the lack of robustness to outliers, using the sum of squared re-projection error directly in the presence of outliers will lead to very poor results as these outliers have a disproportional influence on the resulting least squares estimate. This is particularly the case with outliers caused by bad

correspondences as these often result in large residuals. In such case the assumption of normally distributed errors does not hold and a more robust cost function is required. A robust cost function can be implemented as a re-weighting of the error vector: $\epsilon' = w_i \epsilon_i$ where

$$C(\|\epsilon_i\|) = \|\epsilon_i'\| = w_i^2 \epsilon_i \tag{2.33}$$

The aim of the weighting factor w_i is to attenuate the cost of outliers and many suitable cost functions exist to achieve this. These include the following:

1. Square error cost function:

$$C(\epsilon) = \epsilon^2 \tag{2.34}$$

2. Cauchy cost function:

$$C(\epsilon) = b^2 log(1 + \epsilon^2/b^2) \tag{2.35}$$

3. Huber cost function:

$$C(\epsilon) = \begin{cases} \epsilon^2 & \text{where} |\epsilon| < b \\ 2b|\epsilon| - b^2 & \text{otherwise} \end{cases} \tag{2.36}$$

4. Tukey biweight cost function:

$$C(\epsilon) = \begin{cases} \epsilon\{1 - (\frac{\epsilon}{b})^2\}^2 & \text{if } |\epsilon| <= b \\ 0 & \text{if } |\epsilon| > b \end{cases} \tag{2.37}$$

A graph representing these cost functions is given in Figure 2.14. A more complete discussion of the various properties of the applicable bundle adjustment cost functions can be found in Appendix 6.8 of [40]. For the work in this thesis we make use of the Tukey and Huber cost functions as both perform well in practical experiments [49, 54].

2.5 State Estimation

The material in this section is based on the work of Thrun et al. in their book Probabilistic Robotics [111]. The problem of state estimation can generally be described as estimating some extrinsic properties of a system given (possibly noisy) measurements from multiple sources. In the first part of this chapter we looked at a some specific state estimation problems where vision was the only available sensor. However on typical robot systems this it not the case as they often feature multiple sensors. In most cases each sensor will have their own strengths and weaknesses, for example a gyro sensor is very good at measuring rotational velocity but is prone to drift over time. To address

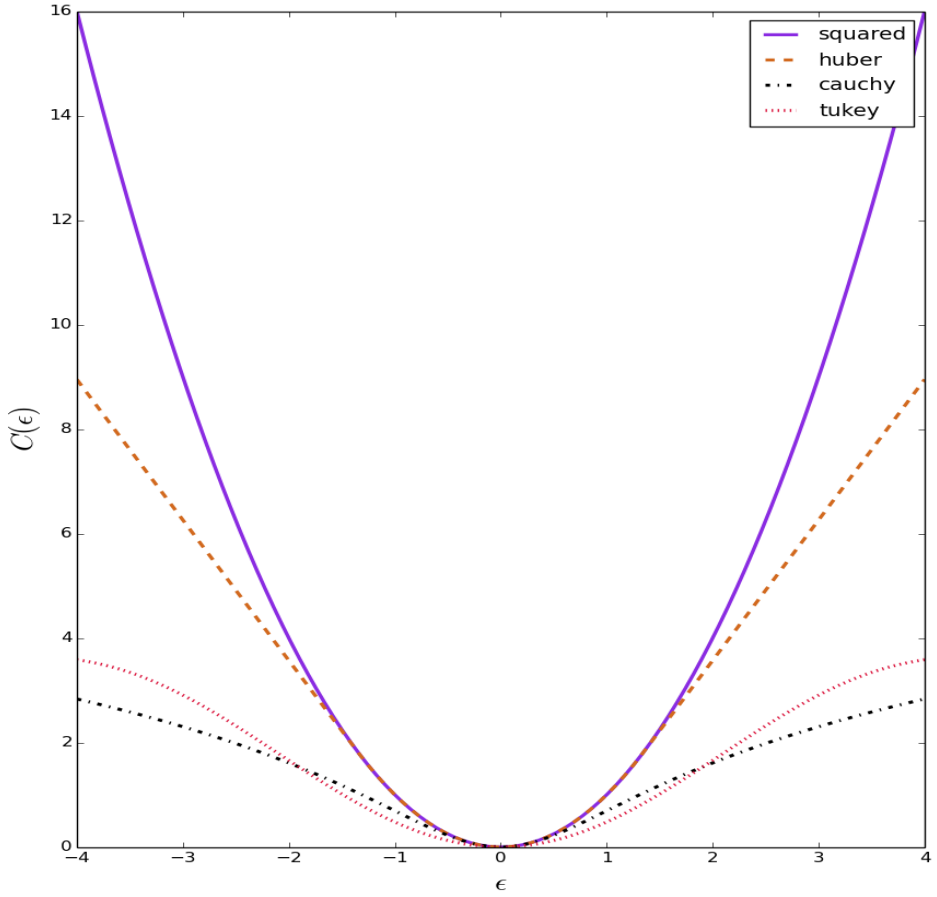


FIGURE 2.14: Graphs corresponding to the different weighted least squares cost functions.

this a common strategy is to combine sensors to mitigate these weaknesses for example adding an accelerometer (which measures acceleration) can be used to correct for gyro drift. Another example is monocular Visual SLAM, using the techniques discussed in this chapter one can estimate the position of a camera within it's environment but only up to some unknown scale factor. However combining the position estimates from a Visual SLAM system with sensor readings from an accelerometer and gyro (which provide readings in metric units) we can estimate the metric scale factor for the Visual SLAM system. This approach is commonly referred to as sensor fusion.

The Kalman filter[50] is a Gaussian, recursive state estimator derived from the more general case Baysian filter. The state, in a Kalman filter, is modelled using a multivariate normal distribution $p(x)$:

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} exp\{ -\frac{1}{2}(x-\mu)^T \sigma^-1(x-\mu)\} \tag{2.38}$$

where the state vector x is characterised by the mean μ and covariance Σ of a Gaussian probability density function. The Kalman Filter is described as a recursive state estimator for linear systems, meaning both the measurement and process models must be linear functions. The state vector defines the set of variables to be estimated for

example the position and velocity of a MAV:

$$\mathbf{x} = (x, y, z, \dot{x}, \dot{y}, \dot{z})^T$$

The Kalman filter process can be divided into two steps, the prediction step which a motion or control model is used to predict the state of the system on time step into the future and the correction/update step where sensor data is incorporated into the state to further refine the prediction.

The prediction step defines how the state evolves from one moment to the next as a result of the process model and the control model. The process model describes how the state changes from one moment to the next regardless of the control command. For example if the state of the system is affected by an external force such as in the case of a MAV operating in the wind, we can account for this in the process model. The control model describes how the state evolves as the direct result of a control command. The input command u_t is a vector of control commands executed at time t for example if we commanded our MAV to accelerate forwards with an acceleration of $1.0\,\mathrm{m/s^2}$:

$$\mathbf{u} = (0, 0, 0, 1.0, 0, 0)^T$$

we can then generate a prediction of the state in the next moment in time:

$$\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t \tag{2.39}$$

$$\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t \tag{2.40}$$

where A_t is a matrix describing the process model, B_t describes the control model and R_t models the uncertainty introduced by the prediction step, represented by a zero mean Gaussian.

The correction step, incorporates measurements or observations of the environment into the prediction. This is done in two parts, first we calculate the *Kalman gain*. Intuitively this can be described as the extent to which we trust our measurements over our prediction of the state. That is if the uncertainty of our measurements is lower than the uncertainty of our state predictions then the correction will be biased towards the measurements rather than the predicted state. The Kalman gain K_t is computed by:

$$K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1} \tag{2.41}$$

where: (i) C_t is the measurement model which relates the current state u_t to the measurement z_t and (ii) Q_t is the covariance of the zero-mean Gaussian measurement noise. In the next part we compute the innovation which is the difference between the actual measurement and the expected measurement calculated from the measurement model C_t and the predicted mean μ_t. We can then adjust the mean in proportion to the Kalman

gain and the innovation:

$$\mu_t = \bar{u}_t + K_t(z_t - C_t\ \bar{\mu}_t) \tag{2.42}$$

Finally the correction step completes with an update of the covariance, based on the incorporation of the new information contained in the measurement:

$$\Sigma_t = (I - K_t\ C_t)\bar{\Sigma}_t \tag{2.43}$$

The complete Kalman filter algorithm is as follows:

Algorithm 2 KALMAN_FILTER$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

Input: The previous mean μ_{t-1}, covariance Σ_{t-1} and current control u_t and current measurement z_t

Output: The new mean μ_t and covariance Σ_t

1: $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
2: $\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$
3: $K_t = \bar{\Sigma}_t\ C_t^T(C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$
4: $\mu_t = \bar{u}_t + K_t(z_t - C_t\ \bar{\mu}_t)$
5: $\Sigma_t = (I - K_t\ C_t)\bar{\Sigma}_t$
6: **return** μ_t, Σ_t

2.5.1 Extension to non-linear systems

A key insight into the workings of the Kalman filter is the fact that any linear transformation of a Gaussian results in another Gaussian [111]. Therefore, provided our system requires only linear transformations, we can ensure our state represented by a multivariate Gaussian distribution remains a Gaussian. However a large proportion of the state estimation problems in robotics involve non-linear functions. The Extended Kalman Filter (EKF) relaxes the linear transformation requirement, specifically we replace the process and control matrices A and B with a non-linear function g and the measurement model C is replaced with another non-linear function h. Additionally as there is no closed-form solution for non-linear functions it is no longer possible to calculate the exact mean and covariance of the state instead we calculate an approximation based on the first order linear approximation of the functions g and h using a Taylor series expansion.

The Taylor approximation of the function g is based on the value and slope of g', where the slope is the partial derivative of g with respect to the current control command u_t and the current mean μ_t:

$$g'(u_t, x_{t-1}) = \frac{\partial(u_t, \mu_{t-1})}{\partial \mu_{t-1}}$$

also known as the Jacobian of the function g. An important note as the Jacobian depends on u_t and μ_{t-1} and therefore must be recomputed each time. The state prediction step

(equation 2.39 then becomes:

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \tag{2.44}$$

and the covariance prediction step (equation 2.40) becomes:

$$\bar{\Sigma}_t = G_t \; \Sigma_{t-1} \; G_t^T + R_t \tag{2.45}$$

Where G_t is the Jacobian matrix consisting of all the partial derivatives of g with respect to u_t and μ_{t-1}. A similar linear approximation is used for the measurement function h. Here the linear approximation is based on the current state prediction $\bar{\mu}_t$. The prediction step is then altered to take this into account:

$$K_t = \bar{\Sigma}_t \; H_t^T (H_t \; \bar{\Sigma}_t \; H_t^T + Q_t)^{-1} \tag{2.46}$$

$$\mu_t = \bar{u}_t + K_t(z_t - h(\bar{\mu}_t)) \tag{2.47}$$

$$\Sigma_t = (I - K_t \; H_t)\bar{\Sigma}_t \tag{2.48}$$

where H is the Jacobian matrix consisting of all the partial derivatives of h with respect to $\bar{\mu}_t$. The full EKF algorithm is then given as follows: This allows us to apply the

Algorithm 3 ExtendedKalmanFilter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

Input: The previous mean μ_{t-1}, covariance Σ_{t-1} and current control u_t and current measurement z_t

Output: The new mean μ_t and covariance Σ_t

1: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
2: $\bar{\Sigma}_t = G_t \; \Sigma_{t-1} \; G_t^T + R_t$
3: $K_t = \bar{\Sigma}_t \; H_t^T (H_t \; \bar{\Sigma}_t \; H_t^T + Q_t)^{-1}$
4: $\mu_t = \bar{u}_t + K_t(z_t - h(\bar{\mu}_t))$
5: $\Sigma_t = (I - K_t \; H_t)\bar{\Sigma}_t$
6: **return** μ_t, Σ_t

Kalman filter algorithm to non-linear process and observation models at the expense of having to compute the Jacobians at each iteration. The drawback of the EKF approach is that the first order Taylor approximation is not always guaranteed to be the best approximation, particularly when the uncertainty in the current estimate is high, this leads to a poor linear approximation and the state estimate may become unstable. An alternative approach is the Unscented Kalman Filter (UKF) [117] which applies a deterministic sampling technique to pick a set of sigma points around the mean. These sigma points are propagated through the non-linear functions from which the mean and covariance of the estimate can then be recovered. This more accurately represents the true mean and covariance.

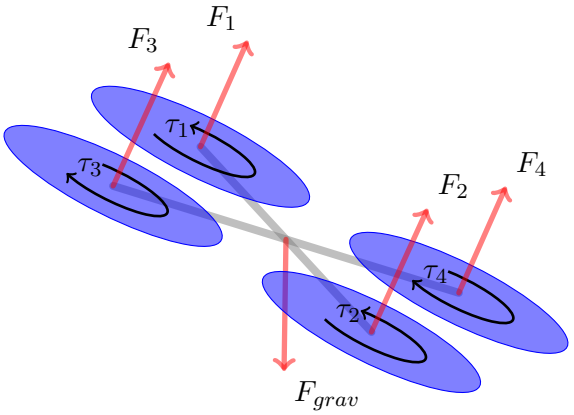


FIGURE 2.15: Quadcopter dynamics: each motor induces a torque T_i causing the propellers to rotate at a certain speed. Each rotating propeller accelerates the air and induces a perpendicular force F_i counteracting the force F_grav that pulls the quadcopter towards the earth.

2.6 MAV Control

In this section we will briefly describe the background relating to MAV control. Specifically we will review the dynamics of a typical multi-rotor MAV and go on to talk about a typical feedback control system. The material in this section is based on the excellent technical report by Randal Beard [6].

2.6.1 Quadcopter System Model

A quadrotor is a MAV equipped with four equally spaced rotors usually placed at the corners of an imaginary square body (this is known as the 'X' or cross configuration). The rotors are fixed pitch meaning thrust is altered by varying the speed of the motor rather than by altering pitch of the rotor (as is typical in single rotor craft). Figure 2.15 shows a simple model of a quadrotor aircraft. Each motor produces an upward force F_i from the thrust generated by the rotating propellers. The total force generated by the craft $\mathbf{F}$ is sum of all the individual forces generated by each motor:

$$\mathbf{F} = F_1 + F_2 + F_3 + F_4$$

Increasing the forces for different pairs of motors induces torques around the centre of mass of the craft for example the pitching torque τ_θ is given by:

$$\tau_\theta = l(F_3 - F_2 - F_1 + F_4)$$

where l is the distance from the centre of mass to the motor. Similarly the rolling torque τ_ϕ is given by:

$$\tau_\phi = l(F_2 - F_1 + F_3 - F_4)$$

As each propeller moves through the air the drag induces a yawing torque $(\tau_1, ...\tau_4)$ on the body of the craft in the opposite direction to the rotation of the propeller. If each propeller spins in the same direction this yawing torque would cause the body of the quadrotor to spin in the opposite direction. Thus the motors are configured to counteract this, motors $M1$ and $M3$ spin counter-clockwise and motors $M2$ and $M4$ spin clockwise. This ensures the yawing torque of each pair of motors in cancelled out by the other pair. The total yawing torque is given by:

$$\tau_\psi = \tau_2 + \tau_1 - \tau_3 + \tau_4$$

When $\tau_\psi = 0$ this means the yawing torques of both pairs of motors are in equilibrium and the quadrotor will maintain it's current heading. When the total force $\mathbf{F}$ is large enough to counteract gravity and $\tau_\theta = \tau_\phi = \tau_\psi = 0$ the quadrotor is in a stable hover. The forces and torques can be computed taking into account the various electromechanical and aerodynamic properties, however this is not necessary for control. Instead given that the lift and drag produced by the propellers is proportional to the angular velocity we can express the forces and torques as:

$$F_i = C_1\omega_i^2 \tag{2.49}$$

$$\tau_i = C_2 F_i \tag{2.50}$$

where ω_i is the angular velocity of motor i and C_1 and C_2 are constants that model the rotor characteristics which can be determined experimentally. This gives the complete torque model as:

$$\begin{bmatrix} \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} -C_M & C_M & -C_M & C_M \\ -l & -l & l & l \\ -l & l & l & -l \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{2.51}$$

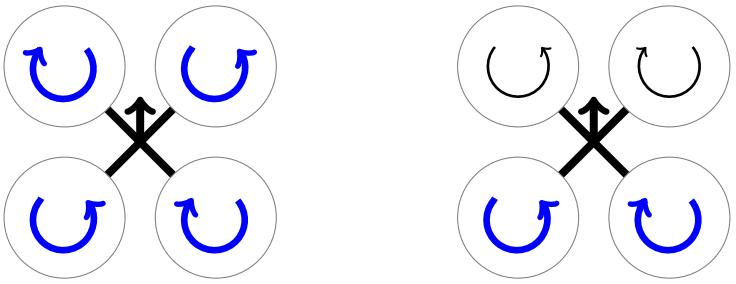
In addition to the forces created by the rotors there is also the gravitational force acting on the quadrotor. The gravitation force vector in the vehicle frame is given by

$${}^{\mathcal{V}}F_g = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

To apply this to the model we need the gravity vector in the body frame, this is given by:

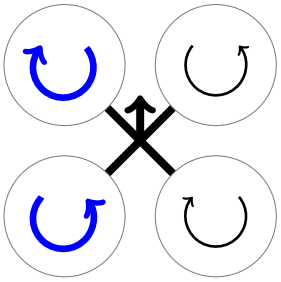
$${}^{\mathcal{B}}F_g = {}^{\mathcal{V}}F_g\mathbf{R} = \begin{bmatrix} -mg\ sin\theta \\ mg\ cos\theta\ sin\phi \\ mg\ cos\theta\ cos\phi \end{bmatrix}$$

where θ and ϕ are the pitch angle and roll angles from the vehicle frame to the body

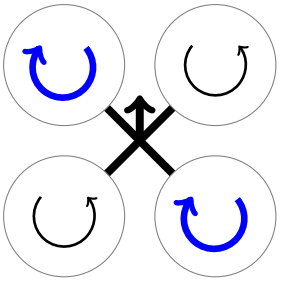


(A) Vertical Climb

(B) Pitch and Translate Forward



(C) Roll and Translate Right



(D) Yaw Left

FIGURE 2.16: Quadcopter control: this diagram shows how varying the speeds of each motor results in a corresponding movement, red arrows indicate increased speed. Note the coupled rotational and translations movements on the pitch and roll axes.

frame. We can then decompose the forces in the force elements along each axis (F_x, F_y, F_z) using the following transformation:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ \mathbf{F} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{2.52}$$

In order to determine the angular accelerations we need to know the moments of inertia, which represent the resistance to rotational acceleration of a body. We can calculate the moments of inertia of quadcopter assuming a spherical dense mass at the centre with mass m and radius r and model the motors as point masses at a distance of l from the centre. We assume the quadrotor is symmetric about all three axes and as such the inertial matrix is given as:

$$\mathcal{J} = \begin{bmatrix} j_x & 0 & 0 \\ 0 & j_y & 0 \\ 0 & 0 & j_z \end{bmatrix}$$

The inertia for the solid sphere and point masses is given as:

$$j_x = \frac{2mr^2}{5} + 2l^2m$$

$$j_y = \frac{2mr^2}{5} + 2l^2m$$

$$j_z = \frac{2mr^2}{5} + 4l^2m$$

Given the above we can approximate the angular acceleration as:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} j_x^{-1}\tau_\phi \\ j_y^{-1}\tau_\theta \\ j_z^{-1}\tau_\psi \end{bmatrix} \tag{2.53}$$

Decomposing equations into their separate components gives us the following dynamic model:

$$\begin{aligned} m\ddot{x} &= -\mathbf{F}(cos(\phi)sin(\theta)cos(\phi) + sin(\phi)sin(\psi)) \\ m\ddot{y} &= -\mathbf{F}(sin(\phi)sin(\theta)cos(\phi) - cos(\phi)sin(\psi)) \\ m\ddot{z} &= -\mathbf{F}(cos(\theta)cos(\psi)) + mg \\ \ddot{\psi} &= \frac{1}{J_x}\tilde{\tau}_\psi \\ \ddot{\theta} &= \frac{1}{J_y}\tilde{\tau}_\theta \\ \ddot{\phi} &= \frac{1}{J_z}\tilde{\tau}_\phi \end{aligned}$$

This simplified model is sufficient to model the motion of the quadrotor as it relates to the control problems we consider in this thesis. This model highlights some important considerations for the control problem. From the state model we can see that the quadrotor has six Degrees Of Freedom (DOF) linear movement in x, z, y and rotational movement in ψ, θ, ϕ but only four controllable actuators, motors $M1, M2, M3, M4$. This means the quadrotor is in under actuated system, meaning all six DOF cannot be controlled independently. In particular we can see that the linear and angular velocities about the x and y axis are coupled meaning the quadrotor cannot translate along the y-axis without a corresponding rotation about the x-axis.

2.6.2 Feedback Control

In the previous section we looked at the dynamic model of a typical MAV and how it can be controlled by varying the speeds of each rotor. It would be unrealistic to attempt to control a MAV in this fashion, instead in this section we will look at common control architecture for a MAV platform. We will start by introducing the general idea of feedback control, introduce a feedback control approach and discuss how this can be used to build an autonomous MAV controller.

In a general feedback control problem we have a estimate of the current state of the system (sometimes called the plant in control theory literature) x_t at time t and a desired state x_{des}. The estimate is based on a measurement z_t which is often subject to noise, given by δ_t. The goal is to produce a controller which takes as input the error e_t between the current state and the desired state and computes the necessary control

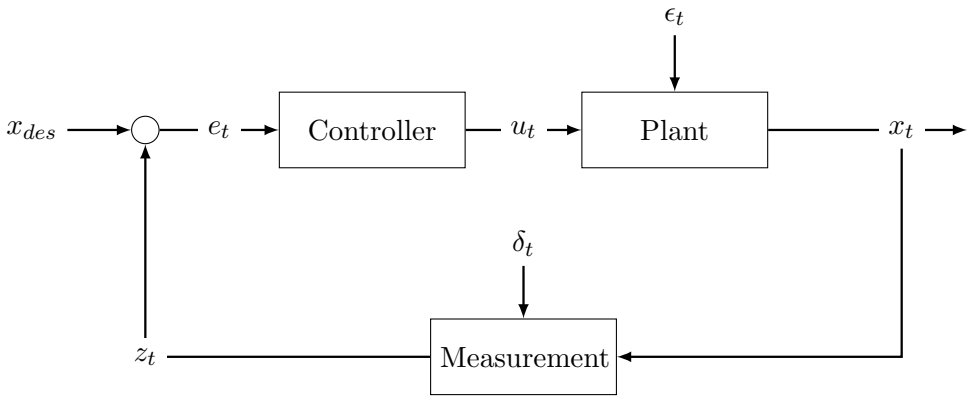


FIGURE 2.17: Block diagram of a typical feedback control problem

command u_t to make $x_t = x_{des}$. The performance of the plant may also be affected by some external disturbances ϵ_t for example wind in the case of MAV controllers.

One of the most widely used control mechanisms [129] is the Proportional Integral Derivative (PID) controller. The PID controller computes a control command as the sum of the three terms that make up it's name. The proportional error represents the simplest solution in a feedback control problem, i.e. compute a control value proportional to the error. The proportional term in a PID controller is given by :

$$K_P e(t)$$

The constant K_P, the proportional gain, can be used to adjust the proportional response. A high gain leads to a fast response but generally leads to instability caused by overshoots. Lower gains lead to a control output that can be too low to overcome system disturbances. The integral term computes the accumulated error which accounts for any system bias or external disturbances. The integral term is given by:

$$K_I \int_0^t e(\tau)$$

In the cases of continuous bias (e.g. gravity) we can add a constant bias term to account for this bias however there are often other sources of bias which are not fixed for example uneven weight distribution on a MAV, or environmental disturbances such as as wind. Such non fixed bias can prevent the system from reaching the desired state. However by calculating the accumulated error and adding this term to the output allows the controller to overcome even non fixed biases.

Combining the proportional and integral terms can sometimes be sufficient, in this case the controller is referred to as a PI controller. In other cases a PI controller can lead to a slow response. In such cases it is often useful to add the derivative term which takes into account the rate of change of the error. The derivative term is given by:

$$K_d \frac{de(t)}{dt}$$

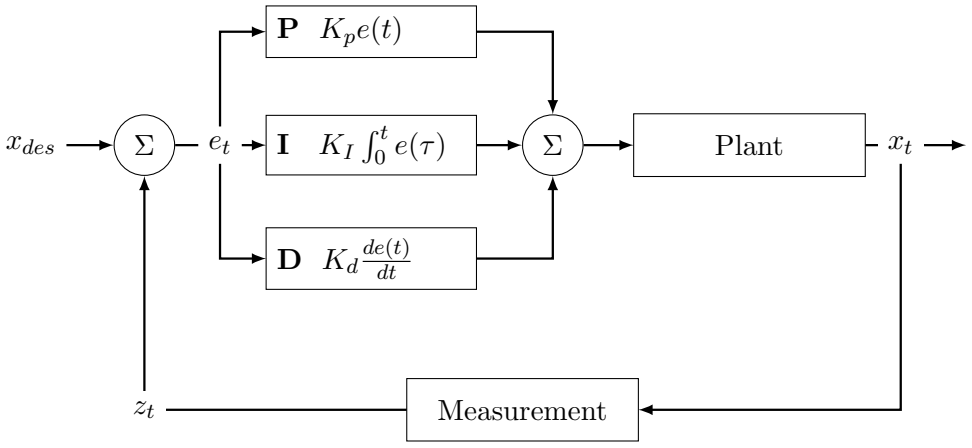


FIGURE 2.18: Block diagram of a Proportional Integral Derivative (PID) controller

The derivative term allows the controller to take into account how the error is changing over time, this leads to a faster response and increased stability. However more than any other term the derivative is affected by measurement noise as large variance in measurements lead to an incorrect or inconsistent estimates of the slope of the error. A common solution is to include a simple low-pass filter before the derivative term or omit it entirely if the variance of the measurement noise is too high. The full PID controller is given by the sum of the three terms:

$$u_t = K_P e(t) + K_I \int_0^t e(\tau) + K_d \frac{de(t)}{dt}$$

The performance of a PID controller is highly dependant on the tuning of the control gains $K_P,\ K_I,\ K_D$. There are often tuned experimentally however many heuristic and automatic methods also exist.

2.6.3 Quadrotor Angular Velocity Control using PID

This section describes how the PID control scheme can be used to control the attitude, or more specifically, the angular velocity of a quadcopter. This is the most basic type of control as the measured angular velocities for the MAVs gyro sensor are directly used to control the angular rate of the MAV about the three axes. To start the the inputs of the system are defined in terms of the controllable degrees of freedom of the quadcopter:

$$x_{des} = (\dot{\phi}, \dot{\theta}, \dot{\psi}, th)$$

where $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ are the desired angular rates in degrees per second and th is the desired throttle setting. The output of the controller should be the angular velocities of the four motors or:

$$u_t = (\omega_1, \omega_2, \omega_3, \omega_4)$$

The problem described above is described as Multiple Input Multiple Output (MIMO) control problem whereas the PID controller is a Single Input Single Output (SISO) mechanism. Meaning the solution requires a set of PID controllers, one for each controllable degree of freedom as well as a method of mapping the output of these controllers to the required output of the system.

The feedback comes from the on-board inertial sensors in the form of the angular velocities measured by the gyro:

$$x = (\dot{\phi}, \dot{\theta}, \dot{\psi})$$

We can define a set of PID controllers: PID_ϕ, PID_θ, PID_ψ for each axis, in this case we assume there is no feedback signal for throttle however a controller can be added if such feedback exists. For each PID controller the error is the difference between the desired angular velocity and the actual value. From the dynamic model of our quadrotor we can determine how much *influence* the output from the PID controllers should have on each motor. This is often refereed to as the motor mix table. An example of a quadcopter mix table is given in Table 2.1. This is an example for a completely symmetrical quadrotor, however a non-symmetrical quadrotor can use the same set of PID controllers by just adjusting the weights proportional to the physical characteristics of the quadrotor. This approach allows the same underlying set of PID controllers to be used for non-symmetrical quadrotor platforms, and for platforms with increasing numbers of motors.

TABLE 2.1: Motor mixing table for quadrotor, based on the motor order from Figure 2.15

Motor	**Throttle**	**Roll**	**Pitch**	**Yaw**
M1	1.0	1.0	-1.0	-1.0
M2	1.0	-1.0	-1.0	1.0
M3	1.0	1.0	1.0	1.0
M4	1.0	-1.0	1.0	-1.0

The angular velocity for each motor is set based on this mixing table for example $\omega_1 = th + PID_\phi - PID_\theta - PID_\psi$. The angular rate controller can then form the basis for a more complex control scheme with additional controllers providing the input to the angular rate controller.

2.6.4 Cascaded PID Control

This sub-section considers the problem of position control, the same strategy as presented above can be employed where the input to the system is the desired position and yaw angle $x_{des} = (x, y, z, \psi)$, and the feedback is in the form of the current position and yaw $x = (x, y, z, \psi)$ (provided by a localisation system e.g. Visual SLAM, GPS). We can then define another set of 4 PID controllers to compute the motor speed commands based on the error between x_{des} and x. This approach would be sufficient for a very calm indoor environment with little or no disturbances as however well the gains are tuned

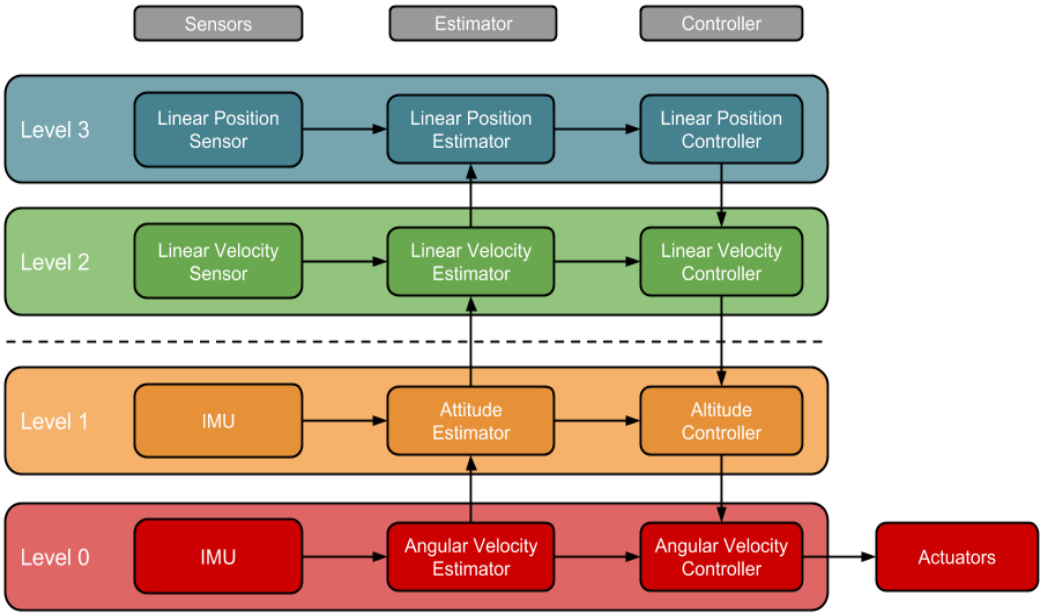


FIGURE 2.19: The general high-level system architecture for an autonomous MAV.

the controllers will not be able to take into account the rapid dynamics of a typical MAV. Indeed this is especially true for localisation solutions with slower update rates e.g. GPS which has a typical update rate of 5 Hz. To improve the performance of the controller and improve it's rejection of disturbances a typical solution is the cascade control approach. A typical cascaded MAV control architecture is show in Figure 2.19. In the case of an quadrotor MAV we can define multiple levels of PID controllers to exploit the available sensor data and control the dynamics of the MAV to a much more granular level. At the lowest level we control the angular velocity of the MAV around each rotational axis, the angular estimator uses data from the IMU to estimate the angular velocity of the MAV and angular velocity controller is a set of PID controllers for each rotation axis.

The set-points for each controller come from the level above, for this to work well the controllers cannot operate at the same update rate. The lower level controllers are typically responsible for controlling the more dynamic aspects of the quadrotor such as angular velocity, which can change extremely rapidly and as such must operate at a much faster rate (typically between 200 to 1000 Hz depending upon the size of the quadrotor and capabilities of the sensors). The controller above will typically run much slower, for example the attitude controller may run between 50 and 100 Hz. This is another important aspect of the cascaded control approach as now each higher level controller can consider the lower level controller as a quasi-static system i.e. one that responds almost instantly to the desired commands sent. Due to the disparity in update rates this is effectively true. This drastically simplifies the complexity of the controllers at each level while still accounting for the fast dynamics of the system leading to improved control performance as well as improved disturbance rejection. The drawback to this approach

is the requirement to tune the gains for all the controllers at each level. However the relative simplicity of each layer means that often a full PID controller is not required, for example PI control is often sufficient for position control.

2.7 Conclusion

In this chapter we have introduced the mathematical background relevant to the work in this thesis. Specifically we introduced the notation and representations for rigid body transformations. We introduced the pinhole camera and FOV lens distortion models and discussed briefly how these are used to model observations made using a 2D camera. We then went on to discuss relevant topics in computer vision including feature extraction, descriptors and feature tracking. The computer vision component of the chapter concluded with a look at the important topic of Structure from Motion in particular the PnP, triangulation and bundle adjustment problems and their solutions. We discussed how these solutions could be used for both incremental motion estimation (localisation) and recovering the 3D structure of an environment (mapping). The chapter concluded with a look at two related topics, State Estimation and Control. We introduced two important algorithms in these domains, the Kalman Filter and PID controller, and discussed how they could be used in the context of autonomous navigation of MAVs.

Chapter 3

Autonomous Navigation for Micro Aerial Vehicles

One of the most difficult challenges in autonomous navigation is the localisation problem, localisation solutions for MAVs can be divided into two categories: (i) those that provide absolute positioning directly such as GPS, or a motion capture system and (ii) those that provide positioning via Simultaneous Localisation and Mapping (SLAM). In this chapter we will explore the related work in both areas; however given the focus of this thesis is visual navigation a large portion of this chapter is dedicated to exploring the related work in that area.

3.1 Absolute Positioning Approaches

Absolute position systems can be divided into two categories, radio based (this includes system such as GPS) and vision based (this includes motion capture systems). This section explores the related work of these systems and their application to MAV navigation.

3.1.1 Radio Based Navigation

The most common radio based navigation solution for autonomous vehicle is the Global Position System (GPS). The GPS system consists of a network of satellites in Medium Eath Orbit (MEO) which is an altitude of approximately 20,200 kilometres. Each satellite is equipped with an atomic clock which is synchronised with the clocks on-board other satellites and the ground. The satellites continuously broadcast a time coded signal which includes the current position of the satellite. A receiver on the ground can use the time coded signal to calculate the distance between satellite and receiver. Typical GPS receivers do not include very precise clocks therefore it is also necessary for the receiver to compute it clock drift relative to the satellites. This means the system must compute four unknowns (3D position and clock dirft) which requires a signal from four or more satellites.

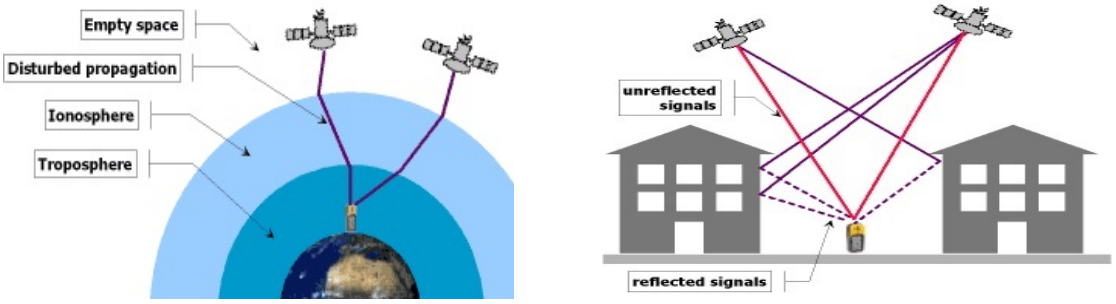


FIGURE 3.1: The two main sources of GPS interference, atmospheric (left) and multi-path (right) [19].

There has been much success both in research and industrial work with MAVs using GPS-based autonomous navigation. Several successful industrial applications such as aerial surveillance and mapping, aerial photography and even autonomous delivery rely on GPS for autonomous navigation. GPS navigation has been used for autonomous ground robots [53, 89] and for MAVs as part of the STARMAC project [44, 47] and even a 27 gram MAV platform [94]. There are two major drawbacks to GPS-based navigation, namely the precision/reliability and the overall coverage of GPS navigation. The precision of GPS-based system is dependant on the sophistication of the GPS receivers used, most off-the-shelf UAS systems make use of GPS receivers with a best-case (i.e. clear line of sight to a number of GPS satellites) precision of 2.5 metres. GPS is primarily affected by two sources of interference see Figure 3.1. Atmospheric conditions both in the ionosphere and the troposphere can perturb the very weak signal sent by the orbiting satellites. However for MAVs multi-path interference has a much larger effect, that is where some or all of the signals do not travel directly to the receiver but instead are reflected off surrounding environmental features such as mountains and tall buildings. This can reduce the accuracy of GPS to up to 26 metres.

Differential GPS systems can improve this accuracy to 0.1 centimetres. Differential GPS is simply put another GPS receiver placed at a known location, this receiver will calculate it's position based on satellite signals and compare this to it's known position. The error between these two data points corresponds to error induced by local atmospheric conditions. This correction can then be broadcast to neighbouring (mobile) GPS receivers allowing them to correct for atmospheric errors. Even with the atmospheric error correction provided by Differential GPS, the sensitivity to multi-path interference and lack of coverage indoors makes GPS based navigation only applicable for large open area navigation.

Other radio based navigation solutions are also possible; in recent years with the growth in coverage of wireless local area networks (WLAN), radio based localisation using WLAN has become a popular research topic [8, 48, 88, 91]. In contrast to GPS which is purpose built for navigation WLAN is not. This means that is typical WLAN does not broadcast precisely timestamped signals that include the transmitters position, as GPS does. This means most WLAN approaches make use of the signal strength rather

than time of flight to determine the distance between the base station and receiver. The relationship between signal strength and distance can be modelled relatively easily when receiver and base station have line of sight between one another. However without line of sight modelling this relationship becomes difficult as the propagation of reflected wireless signals is dependant on many factors some of which include the properties of the occluding material and the frequency and signal strength of the base station.

Another radio based localisation technology is Ultra-Wide-Band (UWB) radio [1, 34, 61]; UWB systems are dedicated navigation systems, similar to GPS in that they broadcast time stamped messages and make use time of flight for distance measurements. A UWB localisation system typically consists of a number of base stations at fixed locations which broadcast time coded signals to moving receivers. UWB radio has several advantages over WLAN based systems, for example signals are transmitted over a larger frequency bandwidth (up to 500 Mhz) which allows the signals to penetrate barriers more easily (as the signal occupies a larger frequency spectrum it is more likely that some part of the signal will penetrate a barrier). While many other positioning techniques are possible one of the most accurate for UWB are the time-based approaches. In cases where the base station are synchronised a time of arrival scheme similar to GPS can be used. The use of a time of flight-based distance measurement approach produce more accurate and reliable distance measurements than signal strength as used with WLAN positions. The typical range of UWB systems is 300 metres in outdoor environment and 100 metres in indoor environments and their accuracy in with the range of 0.1 to 0.15 metres. UWB localisation has been applied to both ground-based robots [57] and aerial vehicles [7, 70].

3.1.2 Motion Capture Based Navigation

Another approach to autonomous navigation indoors is the use of high speed motion capture systems such as Vicon and Optitrack (see Figure 3.2). These systems use a set of high speed infra-red cameras coupled with infra-red emitters to precisely track reflective markers which can be attached to any object. Motion capture systems provide sub-millimetre precision at very high update rate 100-200 Hertz (Hz) which makes them very useful for the precise control of autonomous MAVs. In addition given that multiple sets of markers can be tracked these systems enable the autonomous indoor navigation for multiple MAVs. This facilitates research into a number of interesting applications such as : (i) aggressive manoeuvres [76], (ii) formation flight [59], (iii) object transportation [59] and (iv) bridge construction [3].

There are several drawbacks to using motion capture systems, primarily that entry level systems cost well over £30,000. Additionally these systems have limited scalability and coverage, meaning they can only track a fixed number of MAVs (4-6 depending upon the configuration) and cover a limited area (for accurate localisation the MAV has to be within view of at least 3 cameras). This limits the autonomy of a MAV using such a system for localisation.

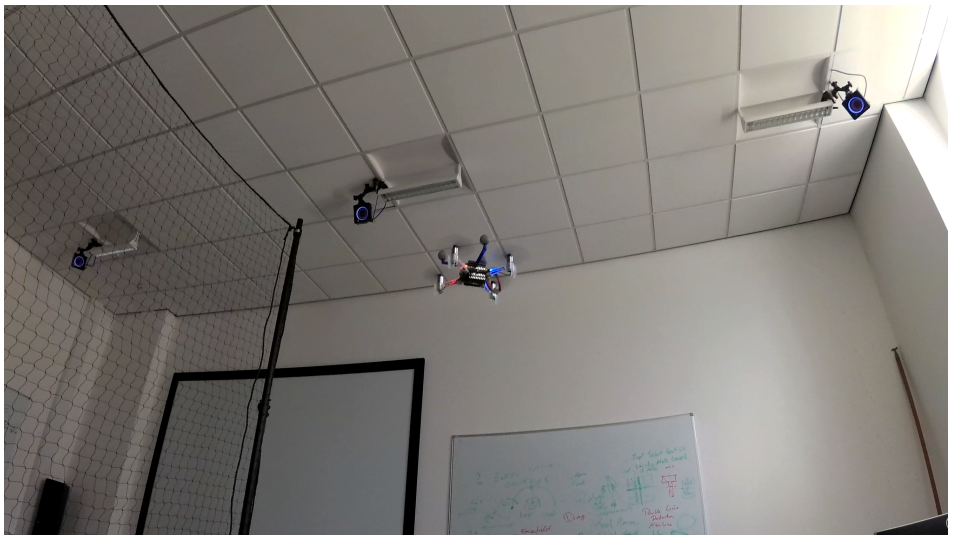


FIGURE 3.2: An example of using motion capture to control a palm sized Crazyflie Nano MAV in the University of Liverpool's smARTLab.

3.2 Simultaneous Localisation and Mapping

All the navigation approaches discussed in the previous section provide absolute position information with respect to some fixed reference frame. For example motion capture systems provide positioning based on the coordinate system defined by the cameras, UWB radios provide positioning based on the coordinate system defined by their based station. Another approach is to create a fixed coordinate system on-the-fly using observations from sensors while simultaneously localising within that coordinate system. This is referred to as Simultaneous Localisation And Mapping (SLAM) introduced in Chapter 1. There are many possible SLAM solutions with applications to MAVs these can most readily be categorised by the type of sensors used; this section will explore the related work for laser range-finder based approaches as well as both stereo and monocular vision based approaches.

3.2.1 Laser Based Navigation

In this Section we will discuss those solution which make use of laser range-finders. Laser range finders provide direct measurements of the distance between the sensor and environment. This is done using an laser emitter and receiver, a pulse is sent out and reflected back to the receiver, distance can then be calculated based on the time of flight. The use of laser range-finder readings to solve the SLAM problem has been extensively studied with respect to ground-based robots [20, 112]. Unfortunately these approaches do not directly translate to MAVs equipped with laser range-finders. Adapting ground -based SLAM techniques to computationally constrained MAVs present numerous challenges, including the 3D motion of the MAVs, lack of wheel odometry and the limited computational resources available. Recent work has addressed this issue using altitude data from the MAV to build 3D maps from 2D laser scans [30] or by mounting a 2D laser

scanning on a rotating platform to produce 3D scans [84]. Other work has focussed on integrating laser based navigation into a complete autonomous control framework [4, 36] including SLAM, sensor fusion and high-level control. Current laser range-finder technology has limited applicability on-board MAVs as while they provide highly accurate readings these units are often heavy and consume enough power to make them impractical to use on-board MAVs.

3.2.2 Stereo Camera Based Navigation

A stereo camera consists of two separate cameras mounted side-by-side at a fixed, known distance. Stereo cameras replicate human binocular vision and allow the recovery of depth data from an observed scene. As shown in Section 2.4.3 the epipolar constraint makes matching points between images a 1D search problem. However the epipolar line is not guaranteed to be in the same place in each image as this depends on the transform between the two cameras. However if the transform (also called the baseline) between two cameras is fixed the images can be warped such that the images are projected onto the same plane. This process is know as rectification. As the rows of each image are then aligned this simplifies the correspondence search even further as well as simplifying point triangulation.

The main advantage of stereo cameras in terms of visual navigation is their ability to recover scene depth from a single pair of images. This makes it possible to recover both the 3D structure of a scene as well as the camera position in metric units. In the monocular system used in this thesis such metric scene and position recovery is only possible via the use of additional sensors or via recognition of objects of known size in the environment. Additionally the immediate depth information provided by stereo cameras can be used for reactive collision avoidance [90, 113]. In terms of localisation and SLAM, several stereo vision approaches have been applied to MAVs Heng et al. [41] demonstrate a complete stereo vision based navigation solution using an off-board visual SLAM solution. Nikolic et al. [85] developed a standalone stereo vision system featuring stereo vision synchronised with an Inertial Measurement Unit (IMU). A visual-interial SLAM system based on this sensor was presented by Leutenegger et al. [64]. One of the drawbacks of passive stereo vision is the need for highly textured scenes to facilitate point matching and triangulation. An alternative is the use of active stereo (or structured light) for example the Microsoft Kinect sensor. In such systems the second camera in a typical stereo system is replaced by a patterned-light projector. The first camera then observes the scene and can compare the projected pattern to what is observed in the captured image and compute the depth of each point. This allows active stereo systems to operate in low texture environments. These systems have also been applied to MAV navigation [46, 52, 63]. Active stereo systems typically make use of infra-red pattern projectors in conjunction with an infra-red camera as this simplifies detection of the projected pattern in the captured image. This limits the use of such sensors to indoor environments where there are typically fewer sources of infra-red light.

An alternative is full stereo are hybrid stereo/monocular approaches which addresses the computational requirements of typical stereo approaches. Shen et al. [102] present such as system where a stereo camera equipped MAV uses a monocular visual-inertial navigation approach running at high speed for motion estimation. The second camera in the stereo pair operates at a much lower rate and is used primarily for scale estimation and drift compensation.

The major drawback to stereo vision systems for navigation is the reliance on the fixed baseline between the two cameras. As depth is estimated from point disparity between the two images the range in which depth can be calculated is fixed by the baseline between the cameras. Points at longer range will exhibit little or no disparity between the two cameras meaning depth values cannot be computed.

While these developments are exciting they only serve to address the computational limitations of stereo vision.

3.2.3 Single Camera Based Navigation

Using a single image sensor is an attractive alternative to stereo or laser based system, due to the significantly (in the comparison to laser range-finders) smaller Size Weight and Power (SWaP) requirements. However, in contrast to all the previous sensors mentioned, a monocular camera has no method of reliably obtaining depth directly from a single image. However, as discussed in Chapter 2 depth can be obtained by matching and triangulating common features from multiple views (i.e. moving the camera). As monocular vision is the focus of the work presented in this thesis the following sections will explore previous monocular navigation approaches in more detail. Previous work on monocular vision-based navigation for MAVs can be divided into three categories:

1. Visual Markers
2. Visual Odometry
3. Visual SLAM

In the following sub-sections we consider each of these categories in turn and discuss related work in each category. Note many of the techniques discussed in this section may also be applied to stereo cameras, with the added benefit of metric scaling. However as the focus of this thesis is monocular systems we will omit application to stereo vision from the discussion.

Visual Markers

We discussed visual markers as an application for the PnP problem presented in Chapter 2. Visual markers simplify many of the challenges of monocular visual navigation, in particular depth estimation. This makes marker based visual SLAM an attractive solution to the problem of vision-based navigation for MAVs. Markers also address one

of the classic problems in SLAM, that of data association; that is the matching of interest point observations to existing points in the map. Visual markers can be generated to encode information similar a bar code. A typical approach is to generate markers representing a range of numbers which solves the data association problem (assuming markers are correctly identified). With ease of detectability and the ability to recover depth from a single image this makes artificial markers very attractive for visual navigation, indeed Sanchez-Lopez et al. present a framework for localising MAVs using visual artificial markers [98]. In Sanchez-Lopez et al.'s work all processing is done off-board, Meier et al. [75] presented a marker-based localisation approach running on board their Pixhawk platform. Artificial marker based localisation systems have some limitations, namely localisation is only possible when markers are in view of the camera meaning either markers must be placed at regular intervals around the workspace or the MAV must rely on other sensors to navigate between markers. This impacts the robustness of such a system and limits the operation of the MAV to pre-prepared environments. While this can be a valid assumption in certain scenarios, for examples robot patrolling, it does limit the adaptability of the MAVs to new environments.

Visual Odometry

An alternative to artificial markers is the use of naturally occurring geometric features as discussed in Section 2.3. While it is more computationally expensive to detect and match such features, the use of natural features means the system will be able to operate in unprepared environments. One of the most common approaches that makes use of natural features is Visual Odometry. In Visual Odometry the aim is to track the motion of the camera by continuously computing the relative motion between frames using feature correspondences (see Figure 3.3. A typical Visual Odometry algorithm shown in Algorithm 4.

Algorithm 4 VisualOdometry(I_{k-1}, C_k)

Input: The previous camera image I_{k-1} and the previous camera position C_k.
Output: The current image I_k and the updated camera pose C_k.

1: Capture new image I_k
2: Extract and match features between I_{k-1} and I_k
3: Compute essential matrix from correspondences between I_{k-1}, I_k
4: Decompose $\mathbf{E}_k$ into $\mathbf{R}_k$ and $\mathbf{t}_k$ and compute relative transform T_k
5: Add relative transform to current pose $C_k = C_{k-1}T_k$
6: **return** I_k, C_k

A typical approach computes feature correspondences using a feature tracking algorithm such as the Kanade-Lucas-Tomasi (KLT) tracker [69] and computes the essential matrix from correspondences using any of the n-point algorithms, for example Nisters 5-point algorithm [86]. It is also common to use a RANSAC in conjunction with an n-point algorithm to ensure robustness to outliers. Visual Odometry has been shown to work well for ground based vehicles and robots where the constrained motion of the vehicles

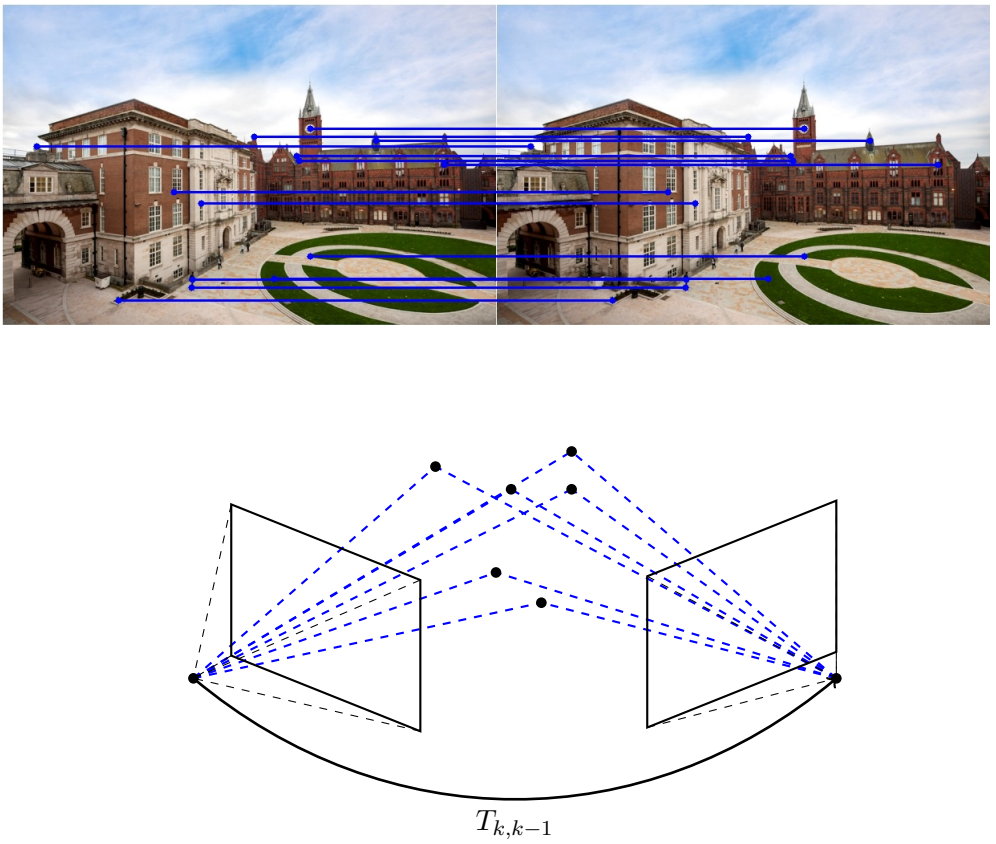


FIGURE 3.3: The visual odometry problem, computing the motion of a camera from incremental computations of the relative transformation between images.

serves to simplify the problem. For example Davide Scaramuzza [99] showed how the non-holonomic constraints for road vehicles can be exploited to constrain the problem such that only a single point is required to compute the essential matrix. For the unconstrained motion of a MAV however such an approach is not possible. However, similar assumptions may be made to simplify the problem, for example a quadcopter equipped with a downward-facing camera. In such a configuration if we assume the MAV is travelling over a planar scene (namely the ground) then we can directly use the optical flow to compute the translational velocity of the MAV. One problem with this approach is that the flow at different heights will be scaled, meaning it would seem as if the higher the MAV flies the slower it moves see Figure 3.4 (right). Pradalier et al. addressed this issue with their CoaX flying platform[92]. Their MAV is equipped with a sonar sensor to measure the MAVs height above the ground. They combined the height measurements from the sonar with the known intrinsics of the sensor to compute the metric velocity from the optical flow. The metric optical flow is computed by:

$$v = h \times \frac{\Delta}{f} \tag{3.1}$$

Where v is the MAV velocity in metres per second, based on the optical flow velocity in pixels Δ, the relative height h and the focal length of the camera f. This solves the issue

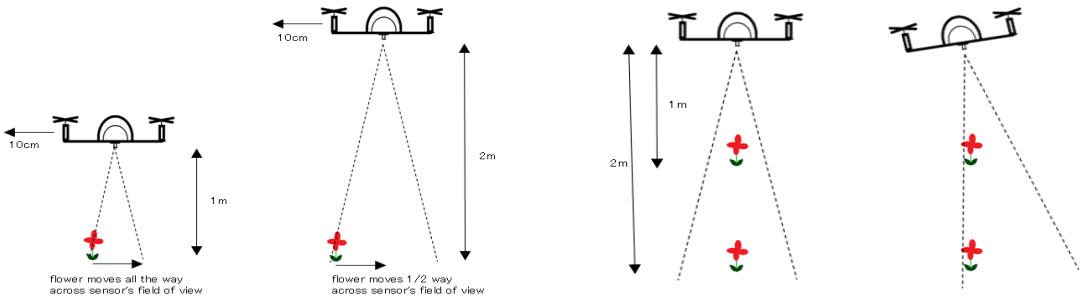


FIGURE 3.4: Challenges of visual odometry on MAVs (right) the effect of height on measured velocity and (left) the effect of rotation on measured velocity.[17]

of scaling the optical flow measurements. Another issue is the rotational movement of the MAV while translating, the calculated optical flow includes the movement induced by both the translation and rotation of the craft see Figure 3.4 (left). Additionally if the craft changes height between frames this would also affect the optical flow calculation. These effects can be compensated for if both the relative rotation and relative height is known or measured by other sensors such as sonar and IMU. Again assuming the scene we are observing is planar then the transformation between the two images I_1 and I_2 can be described by a homography matrix H of the form:

$$\mathbf{H} = \mathbf{R} + \frac{1}{d}\mathbf{t}N^T \tag{3.2}$$

Where R is the rotation matrix between the two frames, d is the relative difference in height above the plane t is the relative transformation and N is the normal vector of the ground plane. The "knowns" are R (from IMU), d (from sonar) and N; and the only unknown is the relative transformation. We can then use the known values and homography to translate image I_2 to have the same rotation and height as image I_1 and calculate the translation from the optical flow as usual. This approach has been used by Honegger et al. to developed a standalone Visual Odometry sensor for UAVs [45]. The sensor includes a downward facing sonar sensor for height compensation and metric velocity estimation as well as an on-board gyroscope to measure the sensors orientation. The addition of an IMU can also be used to simplify the problem in other ways, computing the essential matrix from point correspondences can require up to 8 point correspondences. This is because the unknowns are the translation and rotation between the two images. However the combination of camera and IMU means the rotation for both frames is known leaving on the relative translation unknown. Fraundorfer et al. showed that in this case computing the essential matrix requires only 3 point correspondences [32]. The drawback to this approach however is the requirement for synchronisation between camera and IMU which is not always a straightforward task.

All the approaches discussed so far can be classified as two-frame visual odometry approaches, that is we only compute the relative pose between two images to obtain the cumulative trajectory of the vehicle. The main drawback to this approach is small

errors in the relative pose estimates accumulate over time and lead to a large error in the overall trajectory [100]. One method to address this problem is to estimate the camera transform from more than two images. The most common approach to achieving this is reconstructing the 3D structure of the local scene by feature point triangulation and minimising the re-projection error over the n closest frames (using a non-linear least squares approach). This approach exploits the additional information gained by partially reconstructing the local scene structure to improve the accuracy of relative motion estimation. This in turn reduces the overall drift errors accumulated over the course of a long trajectory. Algorithm 5 provides a general algorithm for the sliding window Visual Odometry. This approach is very closely related to Visual SLAM

Algorithm 5 KeyframeVisualOdometry(C_{k-1}, M)

Input: The current camera pose C_{k-1} and the map M.
Output: The update camera pose C_k and updated map M.
if $M = \emptyset$ **then**
  Initialise map M
end if
Capture new image I_k
Extract features from I_k
Predict new pose C_k using motion model and previous pose C_{k-1}
Calculate visible map features using estimated pose
Re-project map features into I_k and search for matches
Compute pose that minimises re-projection error for matched map points
Add any new features to M and optimise M
return C_k, M

and often multi-frame Visual Odometry and Visual SLAM approaches share much of the same components. However the main aim of Visual Odometry is to compute a consistent trajectory of the camera in real time as opposed to Visual SLAM which aims to jointly reconstruct the camera trajectory and scene structure in a globally consistent manner.

Visual Simultaneous Localisation and Mapping (V-SLAM)

Visual SLAM can be thought of as an extension to Visual Odometry, as instead of only estimating the incremental motion of the camera and reconstructing the scene locally, we now add the additional step of constructing a complete map of the environment. Visual SLAM approaches can be categorised as being either filter-based or keyframe-based. Early work on Visual SLAM focused on so called filter-based methods[13, 14, 16, 21]. In this early work camera pose and feature locations are estimated jointly within the state of a Bayesian filter such as the Extended Kalman Filter (EKF). As camera pose is part of the EKF state, in order to achieve reliable tracking the complete state of the filter must be updated with each new image. However, given that the state contains both the camera pose and feature positions this update is costly and scales quadratically with the number of feature points. Not only does this limit the maximum size of the

map to the order of hundreds of features, it is also computationally wasteful, little new information (in terms of the feature points) is gained by processing every frame. In contrast keyframe-based approaches separate the tasks of camera pose tracking and map building. This allows them to use each frame for the computationally less expensive process of real-time tracking and then build a map from only those images which provide new information, these are referred to as the "keyframes".

This removes the real-time constraints on the map building components allowing the use of more costly but more precise techniques for map optimisation. The extent to which the map is optimised is what differentiates keyframe based Visual Odometry from Visual SLAM. In Visual Odometry approaches we are only concerned with camera trajectory and as such map optimisation approaches tend to be limited to a small local area (namely last n keyframes. In a Visual SLAM system we are also concerned with constructing a global consistent map. As such the process of map optimisation can be more involved. In the next section we describe in detail a popular keyframe based Visual SLAM approach and go on to discuss global map optimisation in detail.

The PTAM Algorithm

One of the most important works in the field of Visual SLAM is the PTAM system developed by Klien and Murray [54] (see Figure 3.5). As the original PTAM algorithm forms a basis for much of the work in this thesis the following section presents a summary of the algorithm as well as a discussion of its key features. Klien and Murray were the first to demonstrate the benefits of splitting the two tasks of camera pose estimation and mapping into separate threads. The tracking thread handles camera pose estimation using features extracted from the camera images. The extracted features are compared to the recorded features from the map and used to estimate the full 6 Degrees of Freedom (DoF) camera pose. The PTAM tracking algorithm is as follows:

1. A simple decaying velocity motion model is used to estimate the current camera pose based on the previous iteration.

2. The stored map points are projected into the current camera frame based on the new estimate of the camera pose.

3. A coarse scale search is conducted for a small number (50) of features (the data association step) and the camera pose is refined based on the matched features.

4. A larger number of features (1000) is projected and searched for in the image.

5. A final pose estimate is computed based on all the matched features found.

New keyframes are selected based on a frame number and distance heuristic. A keyframe consists of a four level image pyramid and the camera pose at which the keyframe was captured. The mapping thread is responsible for processing new keyframes.

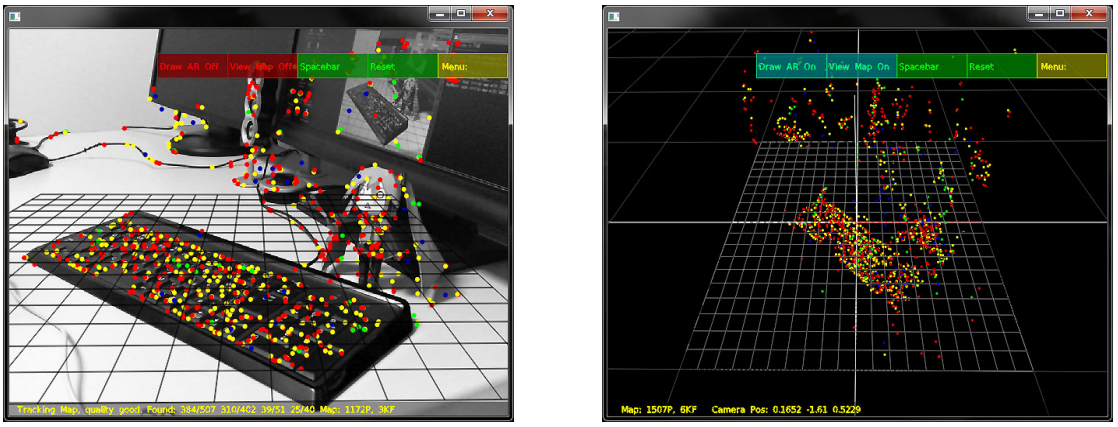


FIGURE 3.5: The original PTAM showing the real-time tracking of features in a typical office desk scene (left) and the corresponding map points (right).

Once a new keyframe is generated the set of extracted feature points are refined to include the most salient points based on their Shi-Tomasi score. Additionally feature points close to existing map points are removed to avoid adjacent points causing bad map point to feature point correspondences. The remaining feature points are searched for in the closest keyframe to perform a triangulation to calculate depth information. If successful a new 3D map point is added to the map. After adding a new keyframe a local batch optimisation is performed on the four closest keyframes in order to refine both the keyframes and map-points. The optimisation approach Bundle Adjustment (see Section 2.4.5), which is commonly applied to offline map generation in the Computer Vision, is used to optimise the map. This is a key aspect of Klien and Murray's approach, by decoupling the tracking and mapping processes into separate threads this removes the real-time constraint on the mapping procedure imposed by the need to process every incoming frame. This facilitates the use of a slower but much more precise map optimisation approach i.e. bundle adjustment.

Another notable feature of PTAM is the lack of explicit modelling of uncertainty characteristic of the EKF-based approach and much of the previous work on SLAM in robotics. This is compensated for by using larger numbers of features as well as the local and global bundle adjustment. Strasdat et al. [106] conducted an evaluation the keyframe-based approach versus filtering and concluded that the use of larger numbers of features and separate bundle adjustment-based map optimisation as used in the keyframe-based approach is both faster and more reliable than filtering based approaches. Additionally the use of large numbers of feature points for tracking makes this approach robust to partial occlusions.

Building on Keyframe-based Visual SLAM

One of the limitations of the original PTAM was the lack of large loop detection and closure. During mapping loop detection refers to the problem of recognising when the robot/MAV has returned to a previously mapped area. This may appear simple however,

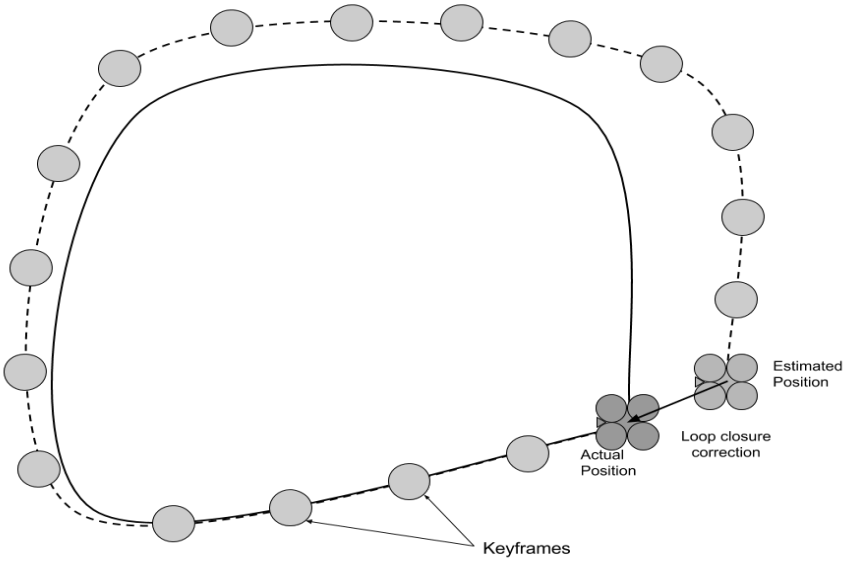


FIGURE 3.6: A simple example of a loop closure situation. In the figure the robots trajectory is show as a solid line and it's own trajectory estimate is given as a dashed line.

due to the accumulated drift as a result of small errors in incremental motion estimation, the robot may believe this has not occurred. An example of this is shown in figure 3.6. In this example a camera equipped MAV follows the trajectory shown building a map of visual features, in this example the stored camera poses, which we refer to as keyframes are indicated as circles. As can be seen from the example as a result of the accumulated errors in the MAVs position estimate both it's current position estimate and the keyframe positions exhibit drift with respect to the ground truth (solid line). To correct this the system must be able to compute the transformation (the loop closure correction) required to realign the map. This correction can then be back propagated throughout the map to correct all the keyframe positions. This is referred to a pose graph optimisation in the literature.

Lacking a fixed reference for scale, in monocular SLAM, drift occurs in both position and scale. To address this Strasdat et al. [106]developed a 7 Degree of Freedom (7DOF) loop closure approach in which both position and scale are corrected. This goes a long way to improving the long range capabilities of a monocular SLAM system. As such this 7DOF loop closure approach has been widely used in modern monocular SLAM systems [23, 28, 79].

Loop closure detection (also called visual place recognition) is a well studied problem in computer vision, the most widely used approaches in the visual SLAM community are image-to-image [121] matching approaches using where image features are used to determine the similarity between images. One of the most widely used image-to-image place recognition approaches is FAB-MAP developed by Cummins et al. [15]. They make use of the the so-called bag-of-words approach in which instead of directly matching image features a linguistic approach is used. The bag-of-words approach summarises

the content of an image in terms of a visual vocabulary. This visual vocabulary is in effect a prioritisation of the descriptor space and is constructed offline from a large set of training images. The more general the training set the easier the vocabulary is to use for different environments and however too general a training set can affect the recall rate. The novelty of Cummins et al.'s approach is the ability to compute the probabilities of two images being of the same place based on the co-visibility of features. A drawback to the FAB-MAP approach is the use of SIFT features which are computationally costly to compute. Klien and Murray make use of an efficient direct image matching approach in PTAM [55] this will be described in more detail in Section 5.2. This approach has been shown to work well where the density of keyframes is high and relative viewpoint between the current frame and the closest keyframe is similar in both our experiments and those of Riazuelo et al. [95]. Recently Mur-Artal presented ORB-SLAM their keyframe-based SLAM approach [79]. They address the problem of place recognition using ORB features, like SIFT/SURF, ORB features are invariant to rotation and scale making them useful for place recognition but are not as computationally demanding and thus can also be used for real-time tracking without a GPU.

Another limiting factor is the cubic complexity of Bundle Adjustment, while many efficient optimisation frameworks go some way towards addressing this it remains a problem particularly for large scale (in terms of the number of keyframes) applications. A second influential paper by Strasdat et al. [106] addressed this problem with a double-window optimisation approach reminiscent of the sliding window approach used in keyframe visual odometry. Here the inner window optimises keyframes and map-points over a sliding window in a similar fashion to PTAMs local bundle adjustment except the inner window consist of the n most recent keyframes as well as any keyframes which observe the same map-points as those in the window. The outer window performs pose graph optimisation on a the set of keyframes where the relative constraints between the keyframes are defined by the co-visibility of map-points. This approach was applied by Mur-Artal for their ORB-SLAM system; they demonstrate good performance even over long trajectories (the longest reported was 2.2 kilo-meters).

Direct methods

An alternative to feature based approaches are the so called direct methods which make use of measurable image quantities (e.g. intensities) for each pixel in the image rather than extracting a sparse set of features. That advantage of direct methods it that the whole image can be used for tracking rather than a sparse set of features. On the mapping side this also allows the construction of more complete environment maps rather the sparse feature maps. A notable example is Direct Tracking and Mapping (DTAM)[82] developed by Newcombe et al. where the mapping thread computes a dense depth-map for each keyframe using the minimisation of a global, spatially regularised energy function. Tracking is done using whole image alignment using the depth-map. This approach

is computationally very intensive and requires large scale GPU parallelisation to run in real-time. To address this issue many semi-dense approaches have been developed.

There have been many recent advances made in parallel with the work in this thesis with a view to running direct methods without the use of a GPU. Engel et al. developed Large Scale Semi-Direct (LSD) SLAM[23]. In LSD SLAM, instead of computing a depth map for the entire image they select image regions with large gradients which provide the most accurate depth information. This hybrid approach is akin to using both point and line features for tracking. This allowed their system to operate in real-time on a CPU and still build semi-dense maps of the environment. The main limitation of the LSD-SLAM approach is the limited map optimisation, as the system does not compute descriptors for image regions it is not possible to re-project map features and optimise the map via re-projection minimisation. This means they must settle for pose graph optimisation and loop closure. The authors of ORB-SLAM presented a comparison between ORB-SLAM (a feature-based Visual SLAM approach), PTAM and LSD-SLAM in which the localisation accuracy of LSD-SLAM was shown to be poorer than both ORB-SLAM as well as PTAM. This is largely down to the lack of structural refinement (i.e. Bundle Adjustment) present in both PTAM and ORB-SLAM.

Foster el al. developed a semi-direct approach (Semi Direct Visual Odometry (SVO)) that can be seen as a bridge between feature-based and direct methods [29]. They use direct image based alignment but instead of operating on a dense or semi-dense depth map they use sparse 3D map points similar to PTAM and other feature based methods. Tracking is done using a combination of minimisation of the photometric and re-projection error. The inclusion of the direct-based tracking makes the tracking very robust even with very fast camera movements. However the inclusion of feature points means they get all the benefits of a traditional bundle adjustment based mapping thread for map optimisation. One important note however is, similar to the work of Wiess et al., in order for SVO to operate in real time on-board and MAV the maximum number of keyframes is restricted.

Visual SLAM for MAVs

The state of the art for Visual SLAM on MAVs can be divided into two categories, off-board where the complete Visual SLAM system runs on a suitably powerful ground station computer and on-board where the full system runs on-board the MAV. Engel et al. demonstrated the efficiency and robustness of the keyframe-based approach to address the problem of visual navigation using MAVs [24]. Their approach builds on the original PTAM and integrates it into a complete state estimation and control framework. They demonstrated the benefits of the monocular visual navigation approach as they were able to develop a system to autonomously control a very light weight (200 g), low cost (£300), off the shelf MAV, namely the Parrot AR. Drone. Running a full monocular SLAM system on-board a MAV is a challenging task. One of the main limitations is bundle adjustment, the complexity for straightforward bundle adjustment is $O((m+n)^3)$

with m keyframes and n features and even sparse bundle adjustment has a complexity of $O(m^3 + mn)$ [54]. Thus the computational complexity scales cubically with the number of retained keyframes. There are many approaches which address this issue, in PTAM the mapping thread regularly performs a local bundle adjustment on a selected keyframe and it's 4 closest (euclidean distance) neighbours. And only performs a global bundle adjustment when the tracker is operating in previously mapped areas.

However, this is still not enough to run PTAM in real-time on-board a MAV. Wiess et al. [118] analysed the effect of the number of keyframes retained in the map and it's effect on drift. They exploited the fact that keyframes further away from the current position have negligible effect on the optimisation step. This allowed them to develop a system with constant complexity by only retaining a fixed number of keyframes. When a new keyframe is added the furthest keyframe from the current position and all it's map points are removed. This allowed Wiess et al.'s approach to run at 20 Hz on-board their MAV equipped with a low cost Atom 1.6 GHz computer. This is the same approach employed by Foster et al. [29]. for the map optimisation back-end of their appearance-based Visual Odometry system Semi-Dense Visual Odometry (SVO).

An alternative is to run part of the system on board the MAV and offload the more computationally intensive components to a more powerful ground-station computer. Such an approach was presented by Andreas Wendel [119], here a single MAV uses the on-board computer for frame-to-frame tracking and off-loads the map creation and optimisation steps to a ground station computer. Wendel leveraged the capabilities of the ground-station computer to also perform dense reconstruction of the environment. Wendel's work was the inspiration for the distributed multi-MAV framework which will be presented in Chapter 5 of this thesis.

Multi Robot Visual SLAM

The multi-robot SLAM problem has been previously explored for ground-based robots with range sensors (such as laser range-finders, stereo vision)[10, 31]. There is much less work on the use of monocular vision as the only extrospective sensor, or involving agents capable of omni-directional (6DOF) motion such as flying robots or hand-held devices (e.g. mobile phones).

The Collaborative Structure from Motion (CSfM) developed by Foster et al. [28] is a multi-MAV visual navigation approach aimed at addressing the issue of map fusion. Each MAV runs a keyframe limited Visual Odometry system on board. When a new keyframe is added by the Visual Odometry algorithm it is sent via wireless link to the centralised map server running on the ground station. The map retains all keyframes and uses these to construct a global map. Both pose graph optimisation and local Bundle Adjustment (BA) are used to correct drift in the maps created. Each MAV in the system has a separate map until an overlap is detected (using appearance based methods considered in Section 3.2.3). When an overlap is detected it is verified using the Perspective-Three-Point algorithm (P3P) (see Section 2.4.1) which also computes the

relative transformation between the two maps from point correspondences. The relative scale between the two maps is computed by comparing the distances between two sets of corresponding points within both maps. Finally the two maps are merged into one and both MAVs now have a common global coordinate system. Foster et al. report localisation Route Mean Squared Errors (RMSE) of 0.04 to 0.06 metres. Foster et al.'s approach is very efficient in terms of bandwidth it makes use of Binary Robust Invariant Scalable Keypoint (BRISK) features [62]; thus each keyframe is only 125 Kilobytes. Foster et al. report that real time performance is still possible with up to 3 MAVs. An interesting note about CSfM is the fact that the map produced by the system is not transmitted back to the MAVs, instead only the corrected global pose of the MAV is transmitted back. This makes the system more bandwidth efficient and it does not require transmission of the entire map to the MAVs. Additionally, the on-board Visual Odometry algorithm can operate in constant time complexity and the size of its map is always fixed [118]. The drawback to this approach is the lack of robustness; if communication with the ground station is lost the MAV does not have the global map in memory and will no longer be receiving drift corrected pose updates from the ground station. This will mean the drift in the MAVs pose estimates will eventually accumulate to the point where it becomes unusable.

C2TAM developed by Riazuelo et al. [95] shares many similarities with both Foster et al.'s work as well as the work presented in this thesis. They also present a multi-robot visual navigation approach, similar to the work presented in this Chapter in that they build on Klien and Murray's Parallel Tracking and Mapping (PTAM). Similar to Foster et al.'s work they also focus on multiple maps, overlap detection and map merging. In contrast to Foster et al. they make use of Klien and Murray's blurred image matching approach (see Section 5.2) for map overlap detection. To resolve the scale ambiguity problem between maps they rely on depth data from RGBD cameras (such as the Microsoft Kinect) to provide absolute scale information. This limits the applicability of their approach to robots with both the payload and processing capabilities to handle an RGBD camera. This also impacts the bandwidth requirements of their system as they must transmit both RGB and Depth data for each keyframe. They report a bandwidth requirement of 1 Megabyte per second (for RGB data only, not including depth data). Riazuelo et al. do not report any localisation performance data; instead a comparison of distances between real world objects such as tables and chairs to those estimated by their system is used to verify mapping performance. The average error they report is 0.0041 metres; however, given the improved accuracy in depth measurement provided by the RGBD cameras used in their work, this result is not surprising.

The system developed by Cherbrolu et al. [12] builds on the more recent Large Scale Semi-Direct SLAM (LSD-SLAM) system discussed in Section 3.2.3. They target ground-based robots with more processing power available than MAVs and as such run the complete LSD-SLAM system on-board the robots. On of the drawbacks to the LSD-SLAM approach is the lack of map and keyframe optimisation via bundle adjustment.

Cherbrolu et al. leverage the additional processing power available on the server and run feature-based bundle adjustment for map optimisation. They use a similar approach to Foster et al. and Riazuelo et al. adopting appearance-based map overlap detection using FAB-MAP. They make use of a three step process to align the two maps. First the relative transformation between the maps is computed using a RANSAC optimised version Horns method. Next they exploit the semi-dense depth data from LSD-SLAM to refine the transformation by finding a transformation that minimises the photometric error between the two frames (LSD-SLAM uses a similar optimisation during pose-graph optimisation). Finally an Iterative Closest Point (ICP) based approach is used to compute the final transformation. Cherbrolu et al.'s work is in the preliminary stages therefore not much in the way of experimental evaluation has been conducted. However given that their work is based on LSD-SLAM we expect localisation performance at least on par with and potentially better (due to the increased correction provided by the Bundle Adjustment) than LSD-SLAM.

Schmidt [101] built on MonoSLAM[16] (an Extended Kalman Filter (EKF) based Visual SLAM system), to allow multiple ground-based robots to map their environment. Similar to Foster et al. and Chebrolu et al. they focus on map overlap detection and map merging. They made use of marker-based robot-to-robot observations to localise a robot within the global coordinate frame. Schmidt conducted experiments with two ground-based robots. They note the limitations of the EKF-based approach and in particular the quadratic complexity of the Kalman filter hindering real-time performance.

Danping et al. present a centralised system for multi-camera navigation in dynamic environments [128]. Danping et al.'s system CoSLAM is able to handle collaborative multi-camera tracking and mapping. Interestingly their system exploits situations where multiple cameras viewpoints overlap to improve map point triangulation. Additionally this camera grouping is not fixed meaning the cameras a free to move around the environment independently of one another, with the caveat that all cameras must be synchronised. CoSLAM also makes use of a re-projection error based classification to differentiate static map points from dynamic ones meaning the approach is able to handle both static and dynamic environments. Danping et al. report experiments with up to 12 cameras however with an average frame rate of 1 frame per second. Their system can be viewed as more as an offline structure-from-motion system than a SLAM system for on-line robotic navigation. Indeed all experiments reported in their work were conducted offline using recorded data.

This section describes the trajectory controller developed for use with the CCTAM and DCTAM frameworks. The controller uses the classic cascaded PID control scheme introduced in Section 2.6.4. As the work presented in this thesis involves multiple MAVs operating within the same environment it was also necessary to include some facility for MAV-to-MAV collision avoidance. To achieve this the controller makes use of a Velocity Obstacle (VO) based approach together with a local communication model. This allows each MAV to compute and follow collision free paths towards their given goals.

3.3 Summary

In this chapter, we have motivated the use of monocular vision-based navigation for Micro-aerial vehicles. Although monocular vision requires an additional level of processing the very low SWaP (Size Weight and Power) footprint makes it the ideal sensor for payload limited MAVs. We discussed in detail the various monocular vision based navigation approaches from Visual Odometry to Visual SLAM and looked at one Visual SLAM system in particular namely Klien and Murray's PTAM. We also discussed the state of the art for on-board monocular Visual SLAM for MAVs and the complexity of the bundle adjustment problem being a limiting factor in the deployment of Visual SLAM on MAVs. In the next chapter we will start by exploring a centralised, off-board approach to multi-MAV Visual SLAM using a low cost MAV platform. We will demonstrate how this simple approach allows cooperative multi-robot tasks to be performed by teams of MAVs.

Chapter 4

Multi-robot Coordination Case Studies

4.1 Introduction

In this chapter we focus on the two multi-agent coordination problems introduced in Chapter 1 namely collision avoidance and exploration. The aim of these two coordination problems is to provide case studies for the evaluation of the multi-robot visual navigation frameworks presented in this thesis. These two tasks were selected as they not only provide case studies for evaluation but also useful functionality for multi-robot experimentation. A fast and reliable robot to robot collision avoidance solution is essential when working with multiple robots in constrained environments. Additionally given that part of the visual navigation work involves mapping previously unknown environments, an automated approach to exploration is also a helpful tool to have.

4.2 Aerial Collision Avoidance Case Study

The collision avoidance problem addressed by this case study is one of aerial collision avoidance between MAVs. The problem of static obstacle avoidance is not considered. Here the goal is for a team of cooperating aerial vehicles to work in and occupy them same space without colliding with one another. Many typical collision avoidance approaches follow a sense-and-avoid approach, that is making some observation of another aircraft using either radar or visual sensors and taking the necessary avoiding action. An alternate approach is a communication-based approach, that is where each vehicle shares it's own position and velocity with all the other vehicles in the area. The sharing of both position and velocity allows each vehicle to determine if it's current heading will result in a collision at some point in the future and take the necessary corrective action.

As discussed in Section 1.3 such position sharing is only possible if the robots share some common coordinate system. Any discrepancy between the reported position and the actual position of a robot may result in a collision. It is for this reason that such an approach was chosen as a case study to verify the performance of a multi-robot

visual navigation system. Such a system allows each MAV to localise within a common coordinate system and allows them to share their positions for the purposes of collision avoidance. Any inconsistencies or drift between the estimated positions of the MAVs and their actual positions may result in collisions.

In order to achieve this a MAV trajectory controller with collision avoidance capabilities was developed. This controller makes use of a Velocity Obstacle (VO) based approach together with a position communication model described above. The remainder of this section is dedicated to an introduction to the VO approach together with a description of the trajectory controller itself.

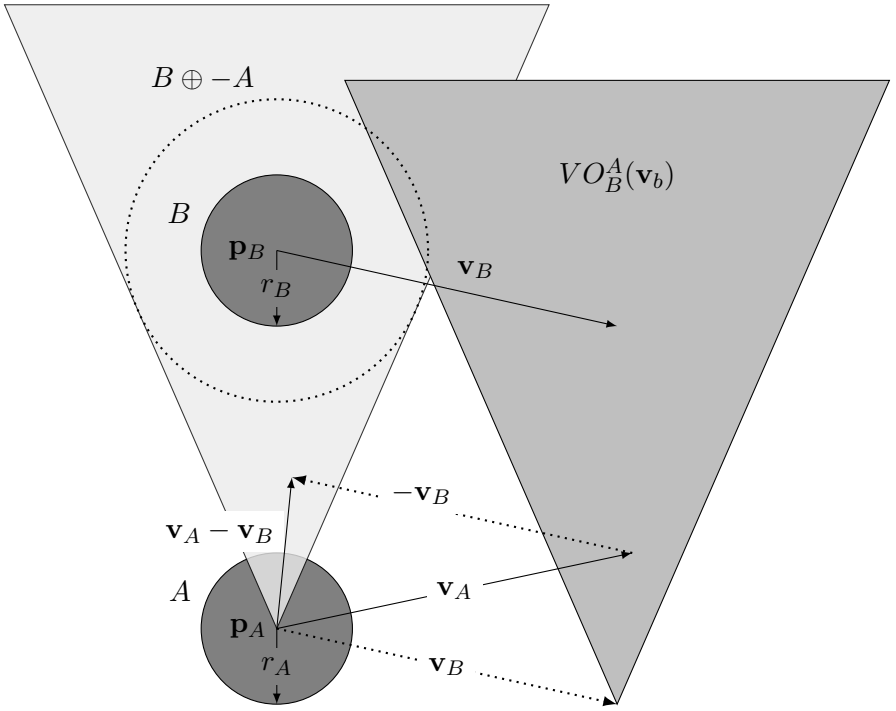


FIGURE 4.1: An example of Velocity Obstacles, here the VO of robot B with respect to robot A in absolute velocity space is illustrated by the dark gray cone.

4.2.1 Velocity Obstacles

The velocity obstacle concept was first introduced in the context of robot motion control by Fiorini et al [26]. The velocity obstacle (VO) is a representation of the set of all unsafe velocities i.e. velocities that will eventually result in a collision. The concept will be introduced using circular shaped robots (to approximate the real ones) operating in a 2D environment. However the concepts principle map readily to 3D environments which is discussed in more detail in Section 4.2.4. Consider Figure 4.1 which features two robots A and B, each represented by a circle of radius r_A and r_B respectively. The position of each robot is given by position vectors $\mathbf{p}_A$ and $\mathbf{p}_B$ and their respective velocities by $\mathbf{v}_A$ and $\mathbf{v}_B$. Let $A \oplus B$ be the Minkowski sum for the two vectors A and B that is:

$$A \oplus B = \{a + b \mid a \in A, b \in B\}$$

and let $-A$ denote A reflected about it's reference point, that is

$$-A = \{-a | a \in A\}$$

Finally let $\lambda(\mathbf{p}, \mathbf{v})$ denote the line with origin $\mathbf{p}$ and direction pointing toward $\mathbf{v}$. We can say that if the relative velocity robot A with respect to robot B given as $\mathbf{v}_A\mathbf{v}_B$ falls within the Minkowski sum of $B \oplus -A$ centred on $\mathbf{p}_B$ then robot A and B will collide at some point in the future. This is represented by the light gray cone in Figure 4.1. However the use of relative velocities makes this velocity obstacle difficult to combine with velocity obstacles generated by other robots. An alternate formulation uses the absolute velocity of robot A, this requires the translation of velocity obstacle to take into account the velocity of robot B, that is the velocity obstacle is shifted by $\mathbf{v}_B$. This is represented by the gray cone in Figure 4.1. Now if the *absolute* velocity of robot A, that is $\mathbf{v}_A$ falls within the translated velocity obstacle the robots are guaranteed to collide at some point in the future. More formally the Velocity Obstacle of B with respect to A is given as:

$$VO_B^A = \{\ \mathbf{v}_A \mid \lambda(\mathbf{p}_A, \mathbf{v}_A - \mathbf{v}_B)\ \cap B \oplus -A\ \neq\ \emptyset\} \tag{4.1}$$

Given that we are operating in the absolute velocity space of robot A given n obstacles we can take the union of all the translated velocity obstacles:

$$VO_{all} = \bigcup_{i=1}^{n} VO_{B_i}$$

If robot A selects a velocity outside VO_{all} then it is guaranteed to be collision free with respect to the n obstacles. Given there may be a large set of obstacles that may or may not be close to robot A at any given moment in time it may be prudent to prioritise obstacles. Fiorini et al. introduced the notion of a time horizon T_h where obstacles are only considered if a collision occurs at some $t < T_h$, often referred to as an imminent collision. Here a simple linear approximation of the obstacles trajectory is used to predict future collision. This results in a modification to the Velocity Obstacle formula:

$$VO_h = \{\ \mathbf{v}_A \mid \mathbf{v}_A \in VO, \| \mathbf{v}_{A,B} \| \leq \frac{d_n}{T_h}\ \} \tag{4.2}$$

where d_n is the minimum relative distance between an obstacle and a robot. This modifies the collision guarantee; now if robot A selects a velocity outside VO_H it is guaranteed to be collision free within time horizon T_h.

A drawback to the Velocity Obstacle approach is that under certain situations oscillations may occur. Consider the example given in Figure 4.2 (left) that features two robots A and B with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$ respectively directing each agent to a goal location. However the robots are on a direct collision course and thus each robot chooses the collision free velocity ($\mathbf{v}_A^{free}$ and $\mathbf{v}_B^{free}$) closest to their target velocities. Figure 4.2

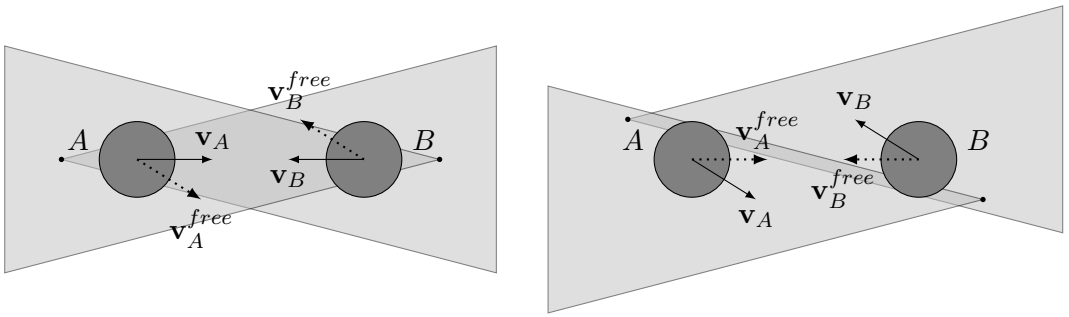


FIGURE 4.2: Illustration of the oscillations that can occur when using the Velocity Obstacle approach [115].

(right) shows the next moment in time and the updated velocity obstacles for both robots, now their original desired velocity is no longer in collision and the robots select this as their new velocity. This leads to a cycle of oscillations which, while still avoiding collisions, does result in sub-optimal trajectories (caused by the constant oscillation).

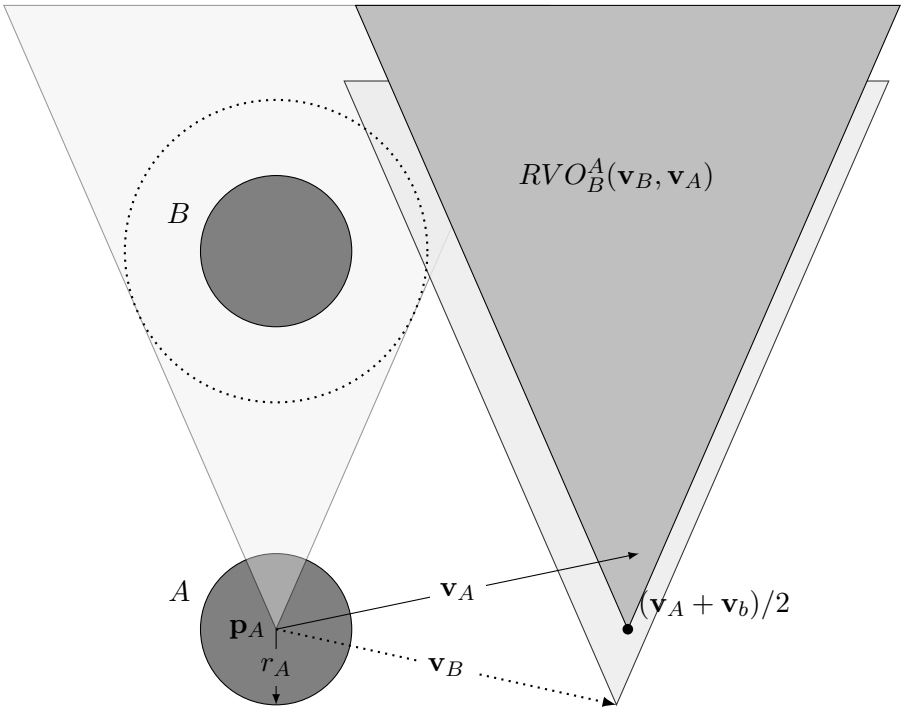


FIGURE 4.3: Illustration of the Reciprocal Velocity Obstacle (RVO) for the example introduced in Figure 4.1.

4.2.2 Reciprocal Velocity Obstacles (RVO)

To address the oscillation issue Van den Berg et al. [115] introduce the Reciprocal Velocity Obstacle (RVO) approach. Van den Berg et al. reasoned that in a multi-agent environment the other agents are not just dynamic obstacles but reasoning agents who themselves take steps to avoid obstacles. If each agent takes *half* the responsibility for avoiding a collision with an obstacle (in this case the obstacle is assumed to be another

robot) then it's motion is still guaranteed to be collision free under the assumptions that all other agents reciprocate by taking the complementary avoiding action. In the case of a static obstacle the robot must still take full responsibility for avoiding collisions (using the normal VO approach). More formally the RVO for obstacle robot B with respect to robot A can be defined as:

$$RVO_B^A(\mathbf{v}_B, \mathbf{v}_A) = \{\mathbf{v}' \mid 2\mathbf{v}'_A - \mathbf{v}_A \in VO_B^A(\mathbf{v}_B)\} \tag{4.3}$$

Here the reciprocal velocity obstacle of robot B with respect to robot A contains the set of velocities for robot A that are the average of it's current velocity and some velocity that lies inside the velocity obstacle $VO_B^A(\mathbf{v}_B)$. An alternative interpretation is the geometric one, in which the apex of velocity obstacle $VO_B^A(\mathbf{v}_B)$ is translated to the point $\frac{\mathbf{v}_A+\mathbf{v}_B}{2}$ as shown in Figure 4.3. The RVO approach removes the oscillation problem experienced using the VO approach. Consider Figure 4.4 which shows the same scenario as in Figure 4.2 except that the robots now use the RVO approach. In the first moment in time each robot selects a velocity outside the RVO induced by the other robot. However because the apex of the RVO is based on the average of the two velocities is remains in the same position as the previous time step. This means that the previously selected velocity remains collision free, resulting in no oscillations.

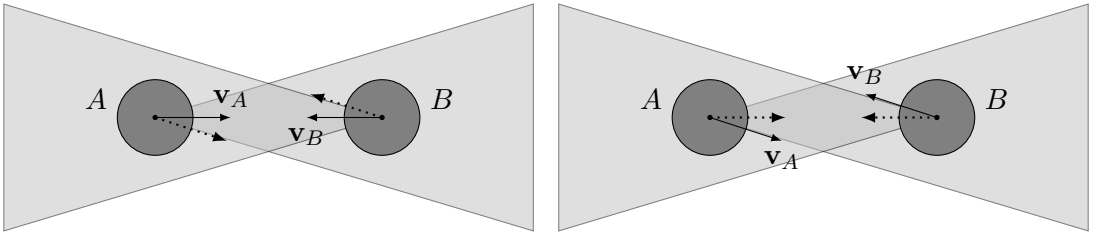


FIGURE 4.4: Illustration of how the RVO approach helps avoid oscillations that occur in situations similar to Figure 4.2.

In order to reach their goals each agent would ideally proceed directly to their goal locations. We describe the velocity that takes a robot directly to their goal location as it's *preferred* velocity. This can lead to situations where a robot chooses a collision free velocity that results in what are referred to as reciprocal dances. This is the robotic equivalent of the social situation most people encounter in daily life when two pedestrians walking towards each other on the street choose to pass each other on different sides. That is person A chooses to pass on the left side and person B chooses to pass on the right. Realising this people often alter their choice and choose the other side to pass, however the other person may do the same resulting in a "dance". This is represented in Figure 4.5 (left). The influence of a third agent may also lead to similar behaviour, this is illustrated by Figure 4.5 (right).

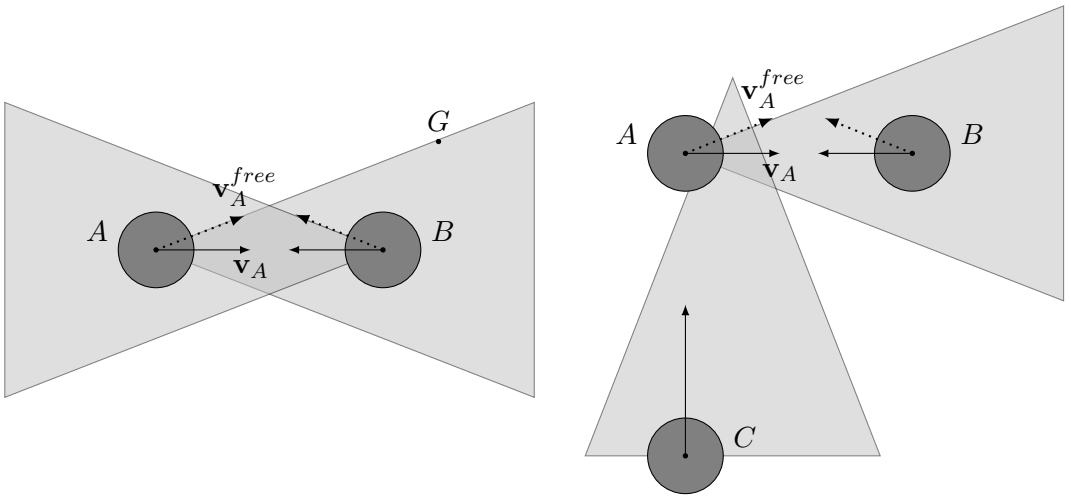


FIGURE 4.5: Two examples of situations where robot A is unable to select a velocity outside the RVO. In the first example (left) Robot A's goal location is given by G, the vector of the goal location means A is unable to select a velocity outside the RVO induced by robot B. In the second example (right) the presence of a third robot C also restricts the choice of safe velocities for robot A. [105].

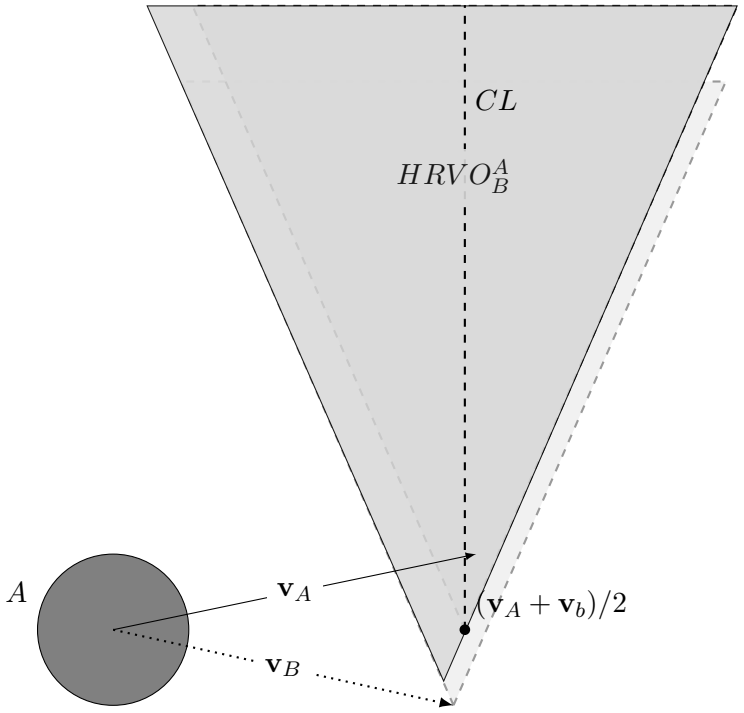


FIGURE 4.6: Illustration of the Hybrid Reciprocal Velocity Obstacle (HRVO) for the example introduced in Figure 4.1.

4.2.3 Hybrid Reciprocal Velocity Obstacles (HRVO)

A solution to the problem of reciprocal dances was introduced by Snape et al. [105] in the form of the Hybrid Reciprocal Velocity Obstacle (HRVO) shown in Figure 4.6. The aim is to bias the velocity obstacle such that each robot will always pass each other on the same side. However simply shifting the RVO is not enough as it only serves to shrink the Velocity Obstacle on one side. Instead the RVO is extended on one side using

the regular VO. Which side is extended depends on where the robot's velocity falls in relation to the RVO centreline (denoted by CL in Figure 4.6). For example if robot A's velocity falls on the right of the centreline the RVO is expanded on the left side, and vice versa. The combination of the RVO and VO results in a Hybrid Reciprocal Velocity Obstacle (HRVO). This means if a robot chooses to pass on the Reciprocal side (the right) it need only take half the responsibility for avoiding the collision. If the robot chooses the left, (the VO side) it must take full responsibility for avoiding the collision. This eliminates the occurrences of reciprocal dances described in the previous section.

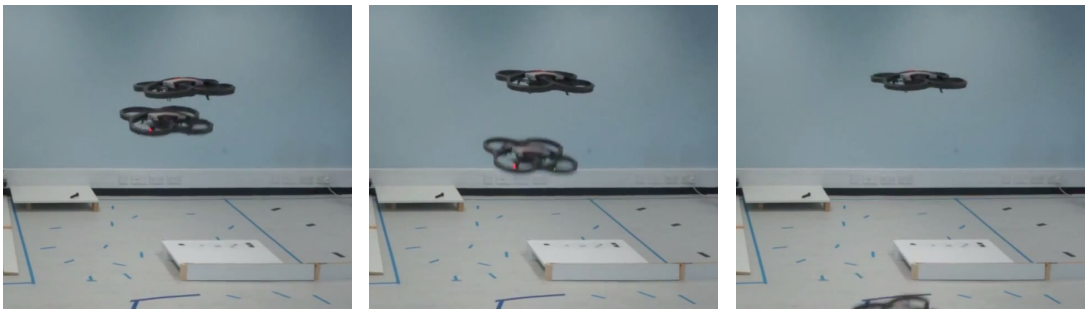


FIGURE 4.7: An example of a crash resulting when a MAV attempts to avoid collision by passing over the other MAV. In the first image (left) the MAVs begin to pass over one another, in the second image (middle) the MAV is pushed down by the propeller wash of the MAV above, in the final image (right) the MAV cannot maintain stable flight and hits the ground.

4.2.4 3D Velocity Obstacles

There are a number of formulations of 3D VO's, examples include that of Snape et al. [104] and [2]. In these formulations the velocity obstacle is modelled using a 3D cone rather than two lines. The main advantage of the 3D approach is a significant increase in the available velocity space, meaning that the search for collision free velocity is made easier. The main drawback to the 3D approach is the overflight problem. Given that robots can now move in 3D to avoid obstacles then passing above another robot is a valid option to avoid a collision. However for a MAV this is not a valid option due to the turbulence induced by the MAVs propellers; this is shown in Figure 4.7. This results in two separate issues, (1) the MAV below receiving a downward push caused by the fast moving air generated by the propellers of the MAV above, and (2) The turbulence (often referred to as propeller wash) generated by the MAV above causing the propellers of the MAV below to generate less thrust. The inconsistent nature of this air disturbance makes it difficult for the flight controller to maintain stabilisation and can often results in a rapid loss of stabilisation leading to a crash. A more effective solution is to consider the collision avoidance problem purely in 2D space. The benefit of this approach is both a reduction in computation time as well as elimination of the fly over problem. The drawback to this approach is a reduction in the available velocity space which MAVs can use to avoid collisions. To provide flexibility both a 2D and 3D velocity obstacles are implemented within the controller described in this chapter. The 2D approach is

used for indoor flights where limited height makes use of the 3D approach problematic, and the 3D approach is available for less constrained environments.

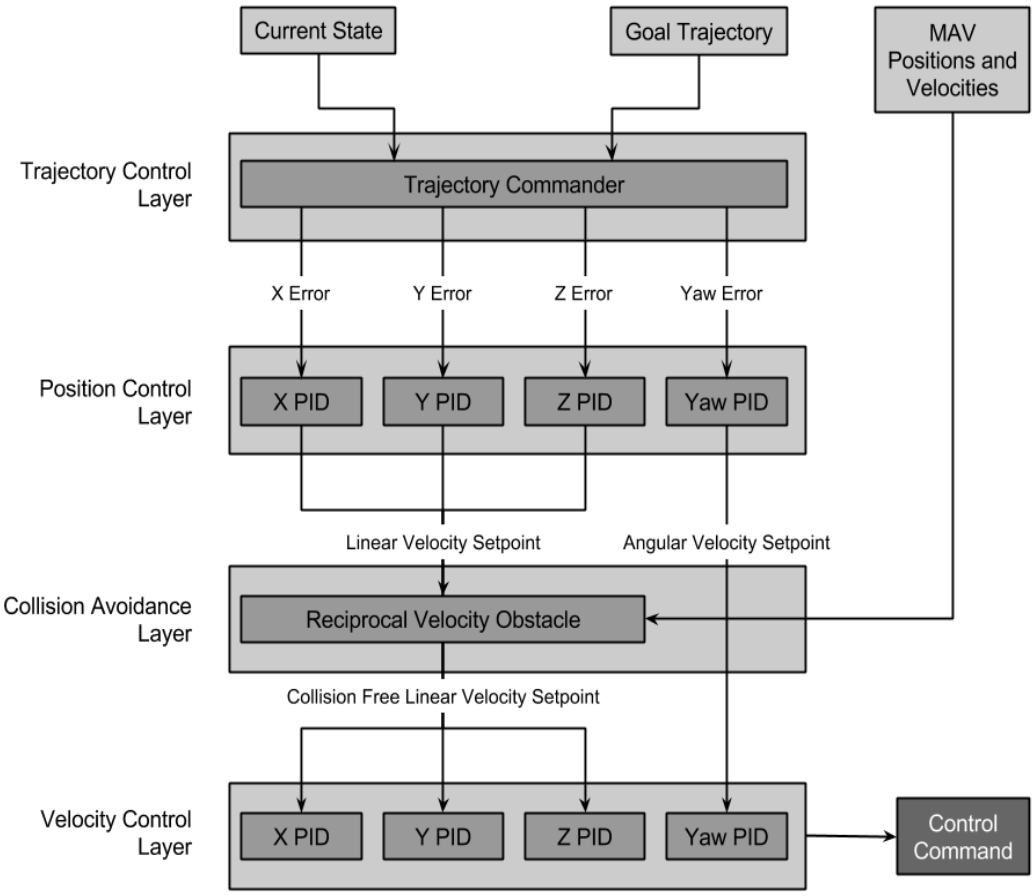


FIGURE 4.8: Controller architecture.

4.2.5 The General Approach

An overview of the controller is given in Figure 4.8. The inputs to the system are:

1. The current state of the MAV $S_t = (x, y, z, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\psi})$, this takes the form of the current position and velocity for each of the four controllable degrees of freedom. Roll and Pitch are coupled to the linear movement in the x and y axes as discussed in Section 2.6.1 and therefore not directly controllable.

2. The current goal trajectory $G = (g_1, \ldots, g_n)$ which is a sequence goal states where $g_i = (x_g, y_g, z_g, \psi_g)$.

3. The current state of the other MAVs $AS = (as_1, \ldots, as_m)$ where each state takes the form: $as_i = (id, x, y, z, \dot{x}, \dot{y}, \dot{z}, r)$ where id is the unique agent identifier and r is the radius of the circle/sphere used to represent the MAV.

At each moment in time the ComputeControl procedure is used to calculate the next control command to send to the MAV based on the current state estimate, the current goal and the current state of any other MAVs. This procedure is described in Algorithm 6.

Algorithm 6 ComputeControl(S_t, g_t, AS_t, T_h, r)

Input: The current MAV state S_t, current goal trajectory point g_t, current agent states AS_t, the time horizon T_h and the MAV radius r.
Output: The control command to send to the MAV sp_{acc}.

$e_{\text{pos}} \leftarrow g_t - S_t$
$\text{sp}_{\text{vel}} \leftarrow \text{CombinedPID}(e_{\text{pos}})$
$\text{sp}_{\text{safevel}} \leftarrow \text{ComputeSafeVel}(\text{sp}_{\text{vel}}, S_t, AS_t, T_h, r)$
$e_{\text{vel}} \leftarrow S_t - \text{sp}_{\text{safevel}}$
$\text{sp}_{\text{acc}} \leftarrow \text{CombinedPID}(e_{\text{vel}})$
return sp_{acc}

Algorithm 7 $\text{PID}(e_t, K_p, K_i, K_d, \max_I, \text{sf})$

Input: The current error e_t the proportional K_p integral K_i and derivative K_d gains, maximum integral $\max_I$, derivative filter smoothing factor sf
Output: The output set point O

$P \leftarrow K_p e_t$
$I \leftarrow K_i(\sum_{i=0}^{t} e_i \Delta t)$
if $I > \max_I$ **then**
    $I \leftarrow \max_I$
end if
$D \leftarrow \frac{e_t - e_{t-1}}{\Delta t}$
$FD \leftarrow (1 - \text{sf})FD_{t-1} + \text{sf}D$
$sp \leftarrow P + I + FD$
return sp

The PID controller used is a modified version of the classical PID controller described in Section 2.6.2, see Algorithm 7. To improve performance in practical applications two modifications are made to the way in which the integral and derivative terms are calculated. These are both widely used in control theory literature. A common issue when the integral term in a classical PID controller is referred to as integral wind-up. The integral term is based on the accumulated error and is therefore susceptible to run-away situations. For example a MAV attempting to reach a goal height may be temporarily stopped by an obstacle. In such a case the error between the goal position and the current position remains high despite the efforts of the PID controller. This causes the integral term to accumulate a large amount of error over a short time. If the obstacle is suddenly removed the accumulated error in the integral term will cause the MAV to shoot up at maximum velocity well past its original goal. A common solution to this problem is to restrict the maximum accumulated error, the ensures the integral

term can still account for disturbances, but prevents run-away situations caused by unobservable control failures.

The derivative term can also be the source of problems if noise in the feedback signal is sufficiently high. The derivative term accounts for the rate of change of the error. However noise in the feedback signal results in oscillations in the derivative, effectively amplifying the noise. In the context of MAVs this has the effect of inducing oscillations as the derivative term is continuously reacting to changes that do not occur. Typical sources of noise on board a MAV are the motors and propellers which create high frequency noise. Therefore a common practise is to include a low pass filter on the derivative term to filter out this high frequency noise.

A PID controller is used for each controllable degree of freedom for a MAV as described in Section 2.6.3. These controllers are arranged in a cascaded PID scheme described in Section 2.6.4 where the Combined PID controller (see Algorithm 8) is used for both position and velocity control in the cascaded structure shown in Figure 4.8.

Algorithm 8 CombinedPID(e^x, e^y, e^z, e^ψ)

Input: The current roll, pitch, yaw and thrust error $(e^x, e^y, e^z, e^{\psi)}$
Output: The roll, pitch, yaw, thrust set points $(\text{sp}^x, \text{sp}^y, \text{sp}^z, \text{sp}^\psi)$
1: $\text{sp}^x \leftarrow \text{PID}^x(e^x, K_p^x, K_i^x, K_d^x, \max_I^x, \text{sf})$
2: $\text{sp}^y = \text{PID}^y(e^y, K_p^y, K_i^y, K_d^y, \max_I^y, \text{sf})$
3: $\text{sp}^z = \text{PID}^z(e^z, K_p^z, K_i^z, K_d^z, \max_I^z, \text{sf})$
4: $\text{sp}^\psi = \text{PID}^\psi(e^\psi, K_p^\psi, K_i^\psi, K_d^\psi, \max_I^\psi, \text{sf})$
5: **return** $(\text{sp}^x, \text{sp}^y, \text{sp}^z, \text{sp}^\psi)$

Algorithm 9 ComputeSafeVel$(\text{sp}_{\text{vel}}, S_t, AS_t, T_h, r)$

Input: The current velocity set point sp_{vel}, the current MAV state S_t, the current state of all other agents AS_t, the time horizon T_h and the MAV radius r.
Output: A collision free velocity set point $\mathbf{v}_{safe}$.
1: $\text{HRVO}_{all} \leftarrow \emptyset$
2: **for all** $as \in AS_t$ **do**
3: $\quad as_{\text{pos}} \leftarrow as_{\text{pos}} + (as_{\text{vel}}\Delta t)$.
4: $\quad \text{HRVO}_{all} \leftarrow \text{HRVO}_{all} \cup \text{HRVO}_{as}$
5: **end for**
6: $\mathbf{v}_{safe} \leftarrow \underset{v \notin \text{HRVO}_{all}}{\arg\min} \parallel S_t - \text{sp}_{\text{vel}} \parallel_2$
7: **return** $\mathbf{v}_{safe}$

After calculating the desired velocity set point, using the position PID controllers this is passed to the Compute Safe Velocity procedure in order to compute the closest collision free velocity to the desired velocity set point. This procedure is described in Algorithm 9. The first step of this procedure is to compute the current position of the other MAVs. This is done by taking the last position and velocity message (AS) received from each agent and updating the position to the current moment in time using a linear velocity model. This helps account for any delay in communication. Then an HRVO

is computed for each agent and combined into a single HRVO for all the other MAVs. From this the closest collision free velocity can be computed. We use the HRVO library developed by Snape et al. [105] to compute the 2D and 3D velocity obstacles.

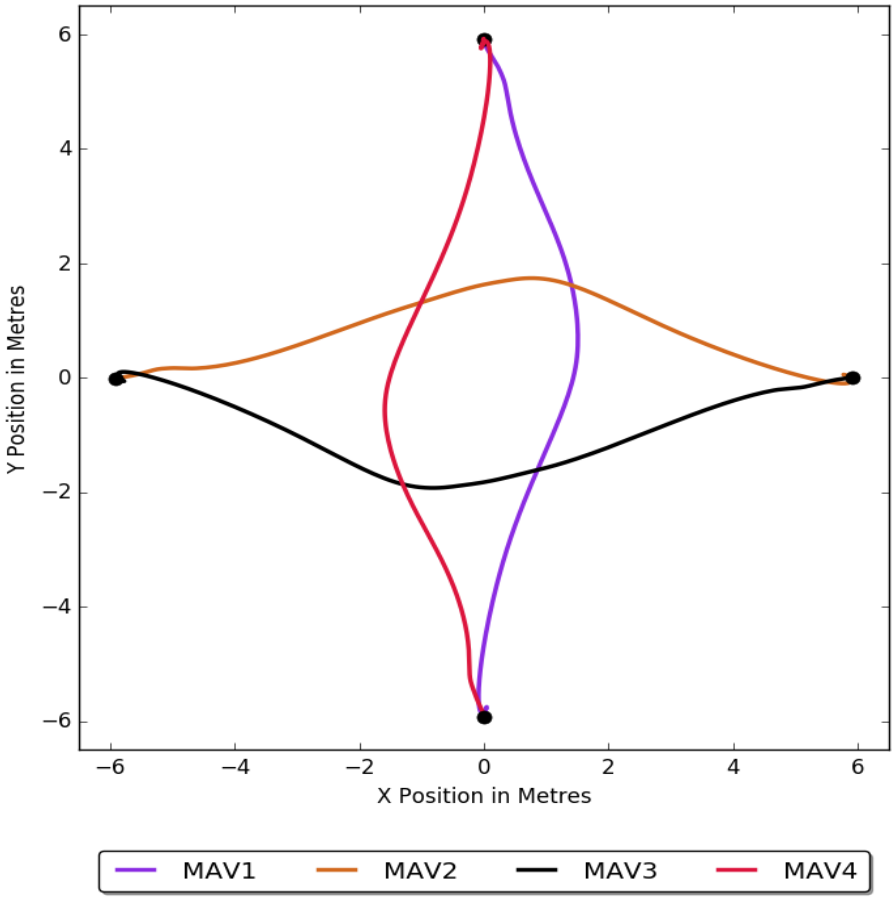


FIGURE 4.9: MAV trajectory plot for a collision avoidance experiment with 4 MAVs, the start location for each MAV is marked with a dot.

4.2.6 Evaluation

In order to verify the performance of the collision avoidance controller introduced a number of simulated experiments were conducted. The simulated environment constructed for these experiments is based on the the Gazebo multi-robot simulator. All simulated experiments were run on a desktop computer with a 3.4 Ghz Intel i7 processor and 16 GB of RAM.

In these experiments simulated ground truth position data was used as input for the controller. As this information is perfect without noise or drift these experiments will serve as a baseline performance measure for later comparison with position data from visual SLAM (presented in Chapters 5 and 6). The experiments were conducted as follows, the MAVs started in a circular configuration and each MAV was commanded to fly to the opposite side of the circle (via the centre). Each MAV starts at the same time and proceeds at the same speed meaning without any collision avoidance all the MAVs

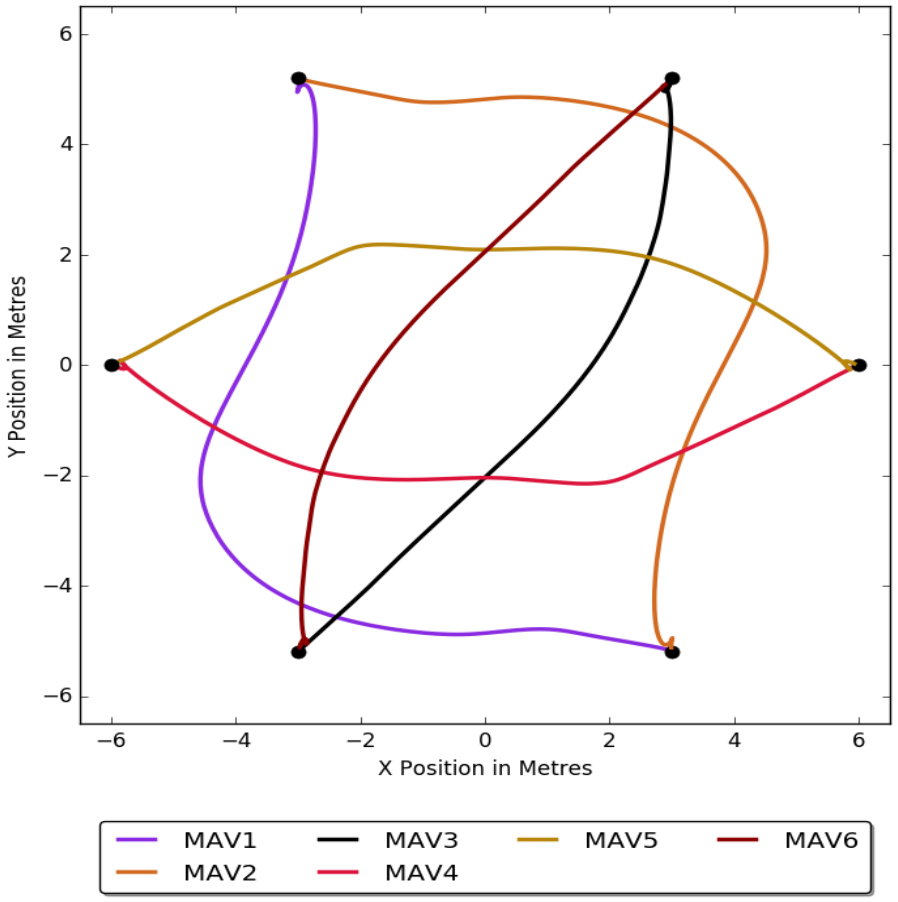


FIGURE 4.10: MAV trajectory plot for a collision avoidance experiment with 6 MAVs, the start location for each MAV is marked with a dot.

TABLE 4.1: Collision Avoidance Experiment Summary

Team Size	**Collisions Avoided**
2	100%
3	100%
4	100%
5	100%
6	100%
7	100%
8	100%

would collide with one another. The experiments were repeated 100 times for each team size and a summary of the results are presented in Table 4.1. In addition 2-dimensional (top down) plots of two representative examples are presented in Figures 4.9 and 4.10.

From the results it can be seen over the 100 trials at each team size no collisions occurred using simulated ground truth data for control. These experiments will be repeated in Chapters 5 and 6 using the position estimates provided by the multi-robot visual SLAM systems presented in this thesis to determine if these system provide position estimates that are accurate and consistent enough to maintain the 100% collisions avoided benchmark.

4.3 Cooperative Exploration Case Study

In this Section a market based cooperative exploration application using visual SLAM is presented. The aim of this application is to demonstrate a simple approach for multi MAV environment exploration using the minimal information extracted from a sparse visual feature map. This work is intended as a case study for the evaluation of the visual SLAM frameworks developed in this thesis rather than a novel approach to MAV environment exploration. The remainder of this Section is structured as follows. An introduction to the exploration problem and a discussion of the related work is presented first. The two main components of the application are then described namely, the interest point extraction and action-based interest point assignment. The section is concluded with a discussion of possible improvements to the system for practical deployments.

The most influential work in the field of autonomous exploration has been the work of Brian Yamauchi [125]. In this work Yamauchi introduced the notion of frontier points, which are points on the border between known (mapped) space and unknown (unmapped) space. Yamauchi's work was based on the occupancy grid used for ground robot navigation. Here each cell stores the probability of the area encompassed by the cell being occupied. Typically an occupancy grid starts with an initial distribution, usually a uniform distribution of a fixed value (for example 0.5). As observations of the environment are collected the occupancy probability of the observed cells is ether increased (if a cell is observed to be occupied) or decreased. Yamaichi used this information to classify grid cells into one of three categories:

1. **open**: occupancy probability $<$ initial probability
2. **unknown**: occupancy probability $=$ prior probability
3. **occupied**: occupancy probability $>$ prior probability

A frontier cell is defined as any open cell adjacent to an unknown cell. These cells represent the border between known and unknown space. Adjacent frontier cells are then grouped into a regions and any region above a certain threshold is considered a frontier point worth exploring.

A related research area is that of active SLAM, the goal of active SLAM algorithms is to map and environment while jointly reducing the uncertainty within the map. The challenge here is exploring a frontier points gains information, but may also increase the overall uncertainty of the map. Loop closures provide a method to reduce uncertainty, that is by revisiting a previously mapped area so that any drift in the robots trajectory can be corrected and the correction can be propagated throughout the map thus reducing overall uncertainty. Many active SLAM approaches will maintain a balance between exploring frontier points and identifying and visiting loop closure locations. This ensures that as the map continues to grow (and with it the overall uncertainty) the robot will regularly revisit previously mapped areas to perform loop closures and reduce uncertainty.

The aim of this application was to develop a way to deploy a team of MAVs to autonomously explore an environment. More formally given a map m, consisting of k feature-points, the object is to extract a set of *interest points* $i \subseteq k$ so that by assigning a MAV to explore these interest points extends the existing map m. This presents two challenges:

1. How to extract the set of interest points,
2. How to assign these tasks to the team of MAVs in an efficient manner

To address these two challenges we have developed an application that uses a frontier-based approach for interest point extraction and a Sequential Single Item (SSI) auction approach for task assignment.

4.3.1 Interest Point Extraction

In order to build a reliable map each MAV must be certain of its position when adding new features, this means that each MAV must keep a portion of the existing feature map visible at all times in order to maintain good visual tracking. Another factor to consider is the ability of the system to close large loops; in the case of the visual SLAM approaches presented in this thesis only small loops closures are possible. Therefore the interest points must be sufficiently close to previously mapped areas while still being sufficiently far away so new features can be discovered. Therefore the aim of the interest point extraction method was to allow the MAVs to map the environment while continuously maintaining a overlapping view of previously mapped areas. This approach is therefore less efficient in terms of distance travelled for the MAVs but it consistently produces reliable maps; which is demonstrated in the experimental evaluation in Sections 5.8.7 and 6.8.7.

The notion of keyframes in visual SLAM was introduced in the previous chapter, a keyframe is at the most basic level a snapshot of an environment from which geometric features are extracted for the purposes of localisation and mapping. Each keyframe represents a viewpoint of the environment from which map features are extracted. Under the assumption that we are mapping from a top down perspective (using a downward facing camera) we can represent this viewpoint as a rectangle on a plane. This rectangle represents the slice of the world captured by the keyframe. We can reconstruct the actual size of this viewpoint on the ground plane using the known intrinsic properties of the camera (image width and height as well as focal length) together with the average scene depth (this is the average depth of all map points visible in the keyframe). This is effectively the far plane of the viewing frustum in computer graphics terminology [108]. The frustum is the pyramidal region which encapsulates the region of space of the world that currently appears on screen (or in some viewpoint). Given a set of frustums created from the set of keyframes in a map it is possible to determine how much of the plane representing the world has been mapped. Figure 4.11 shows an example of a keyframe

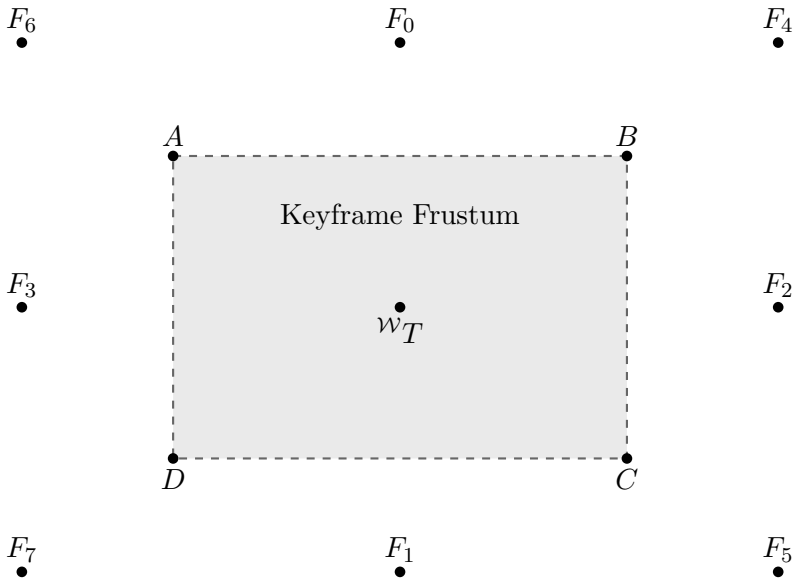


FIGURE 4.11: Illustration of the frustum based and candidate keyframe poses.

frustum with the keyframe pose ${}^{\mathcal{W}}T$ at the centre point. In order to create a new keyframe we must be reliably localised therefore the view of that keyframe must overlap with an existing frustum. Thus knowledge of the existing frustums can be used to predict the location of a new keyframe. This is done by creating a frustum that partially overlaps with an existing one. The amount of overlap can be configured and helps determine the keyframe density of the resulting map. This new frustum can be back projected, by the scene depth of the previous one, to determine the predicted position of this new keyframe. For each new keyframe added to the map a set of new keyframe candidate points are generated. In Figure 4.11 the eight candidate points $(F_0, \ldots, F_7)$ are shown. After the set of candidate points have been generated for each new keyframe the points are filtered to ensure they do not overlap with existing keyframe frustums and are not too close together. This is now the final set of interest points for the MAVs to visit. Figure 4.12 show the extracted interest points for an example map.

This approach is computationally very simple and thus can be run on maps with several hundred keyframes. Additionally, as can be seen in Figure 4.12, this approach will also identify gaps or holes in the keyframe coverage.

4.3.2 Auction Mechanism

There are many approaches to assigning tasks to robots in a multi-robot environment in both a centralised and distributed fashion. Auction mechanisms offer an attractive solution due to their flexibility and ability to distribute solution computation to multiple agents. Auction mechanisms for multi-robot coordination are a well studied area and many practical experiments have shown the Sequential Single Item auction (SSI) to be an effective solution to multi-agent task allocation problems [56, 127].

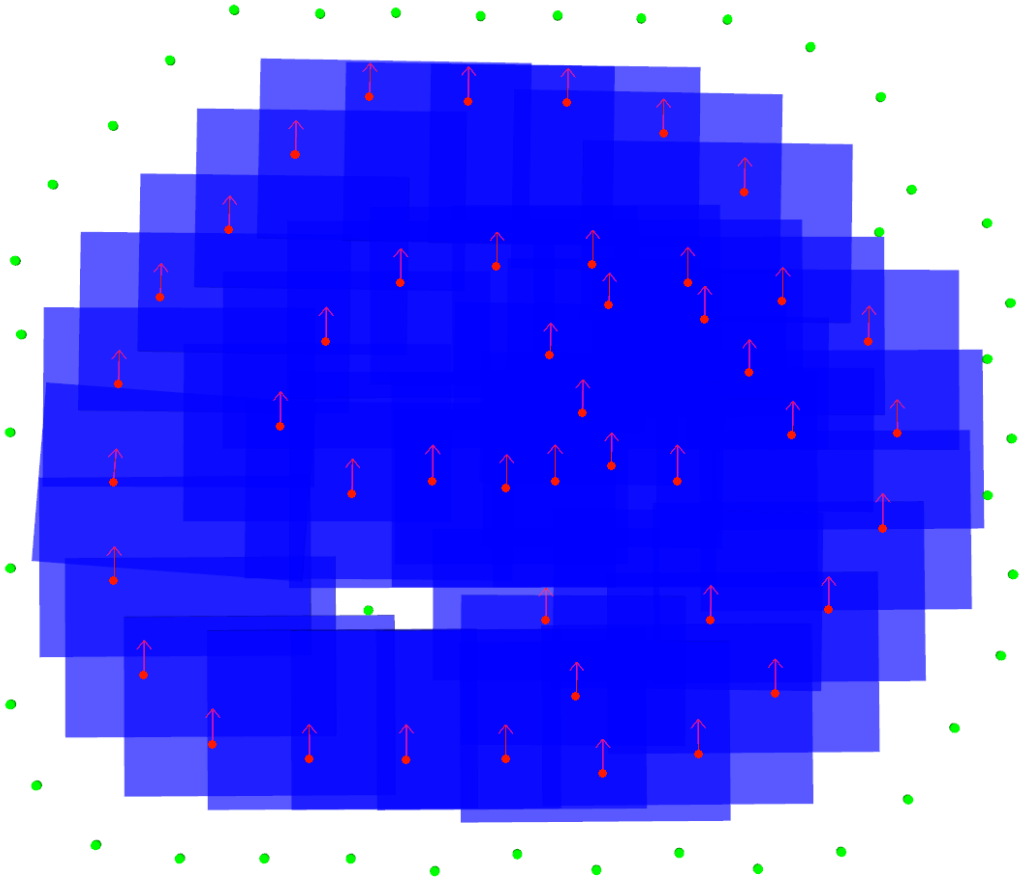


FIGURE 4.12: An example of frontier point extraction; the keyframe positions are shown as red arrows, the keyframe frustums are shown a blue rectangles and the frontier points as green dots.

An SSI auction proceeds as follows. Given a set of targets $T = t_1, \ldots, t_n$ and a set of robots $R = r_1, \ldots, r_n$. At each round a target t_i is selected and advertised to the set of robots R. Each robot r_i bids on the marginal path cost for the advertised target, the bid is therefore the cost of adding the current target t_i to robot r_i's already allocated targets. The winner is the robot with the smallest marginal cost bid, this procedure then repeats until all targets are assigned. Lagoudakis et al. [60] showed that for the resulting allocation from a SSI auction the sum of the travel times for all robots will in general be a factor of 1.5 times the minimal, but at the worst case will only be 2.0 times the minimal.

Another insight into the SSI mechanism is it's flexibility in terms of architecture. A completely centralised implementation is possible as the only prerequisite for computing a bid is knowing the current positions of all agents and targets. With respect to the work presented in this thesis this information is already being shared for collision avoidance purposes. Conversely a completely distributed implementation is also possible as winner determination involves selecting the agent with the lowest bid. Provided each agent broadcasts their bids the auction mechanism can be implemented in a completely

distributed fashion. This fits well with the architectures of both the centralised and partially distributed frameworks which will be presented presented in this thesis.

The centralised auction protocol for multi MAV exploration proceeds as follows:

- Once an initial map has been created the first set of interest points is extracted using the keyframe frustum method described in the previous section.
- The set of bids for each MAV and each point are then calculated based on the interest point locations and the MAV current positions. A winner is determined for each MAV and paths are computed for each MAV to visit their assigned points. These paths are passed on to the trajectory controller of each MAV for execution.
- During the execution phase each MAV visits the interest points which adds new keyframes to the Map.
- These new keyframes are used to compute a new set of interest points and another round of the auction commences.
- This process continues until no more keyframes are added to the map.

4.3.3 Evaluation

As this case study requires the data from a visual SLAM system no evaluation using simulated ground truth data is possible. As such the evaluation of this approach will be differed until these systems are introduced (in Chapters 5 and 6).

4.4 Conclusion

This Chapter has presented the two multi-robot coordination case studies which will be used in the evaluation of the multi-robot navigation frameworks presented in this thesis. Specifically an approach for trajectory control for MAVs and agent-to-agent collision avoidance based on the Hybrid Reciprocal Velocity Obstacle Approach. An experimental evaluation of this controller with varying numbers of MAVs using simulated ground truth showed a 100% collision avoidance rate. This result provides a baseline for later comparison with the visual SLAM systems presented in this thesis.

The Chapter has also presented an auction based approach to multi MAV environment exploration. An interest point extraction method was also presented based on keyframe viewpoints as well as a SSI auction based task allocation approach. The use of the keyframe view point approach for feature extraction is computationally cheap and also identifies gaps in the feature map. The SSI auction approach provides a flexible approach to multi-robot task assignment with favourable worst-case guarantees. The applications presented in this Chapter serve as both example applications for the visual navigation frameworks in this thesis, but also as useful tools for evaluating those frameworks. The results for both are presented in Chapters 5 and 6.

Chapter 5

A centralised approach to multi-robot Visual SLAM

5.1 Introduction

The problem of multi-robot SLAM is typically addressed using a map merging approach. That is each robot proceeds as normal using a single robot SLAM approach until a convergence event occurs. These convergence events typically take the form of the robots meeting each other (robot rendezvous) or the maps produced by a pair of robots are detected to overlap. When these convergence events occur the maps of the two robots can be joined together and the robots can proceed to navigate in a common coordinate system. While this approach is the most flexible it does have some drawbacks. The rendezvous approach requires the robots be able to sense one another, which dependent on their sensory capabilities can be challenging or unreliable. Another drawback is that direct cooperative behaviours are either limited or not possible until such a convergence event occurs. The direct cooperative behaviours we refer to are those described in Section 1.3.

Another solution to the problem is to have the robots all start from known locations, that is the robots all start localised within the same coordinate system. This approach is less desirable as ensuring the robots all start from known locations can be cumbersome to achieve in practical situations. However for camera equipped MAVs, appearance based re-localisation techniques (see Section 3.2.3) offer a solution to this problem. As appearance-based localisation makes direct use of local image features which are available to any camera equipped MAV, this allows a MAV to localise itself within the visual map of another MAV. The aim of the work presented in this Chapter is to explore the feasibility of using appearance-based localisation within a Visual SLAM framework to support large scale multi-MAV visual navigation. To achieve this we developed a proof-of-concept visual navigation system which makes use of an off-the-shelf commercial MAV platform the AR.Drone (see Figure 5.1). This work is founded on Klien and Murray's

original PTAM library [54] as well as the excellent state estimation framework for the A.R. Drone developed by Engel et al. [24]

FIGURE 5.1: The Parrot AR.Drone, a commercially available MAV platform used for the work presented in this Chapter.

The remainder of this Chapter is organised as follows in Section 5.2 we discuss the multi-MAV visual navigation approach in general terms, Section 5.3 presents an overview of the proposed approach. Sections 5.4 to 5.6 provide a detailed description of each component. The implementation of the proposed approach is discussed in Section 5.7, followed by presentation of the experimental evaluation in Section 5.8. The Chapter is rounded off with a short summary and conclusions in Section 5.9

5.2 Multi-camera Visual SLAM via Re-Localisation

In Section 3.2.3 the concept of place recognition for loop closure detection and re-localisation was introduced. In PTAM Klien and Murray make use of a direct image matching approach [55]. As the keyframes retain the original image as well as all the down-sampled images in the pyramid the entire image is available to be used as a single image descriptor. This approach relies on a large population of keyframes providing a dense sampling of the different viewpoints within the mapped area. They take the lowest pyramid level of 80×60 pixels and down-sample further to 40×30 pixels. A Gaussian blur with a default $\sigma = 2.5$ pixels is applied and the mean image intensity is subtracted to make the descriptor more robust to changes in illumination. This is done for each of the k keyframes.

Once re-localisation is initiated the current image I_1 is similarly processed and then matched against all keyframes in the map taking the one with the smallest sum of squared differences as the closest keyframe. The current camera position is set to that of the closest keyframe. To estimate the camera rotation between the keyframe image

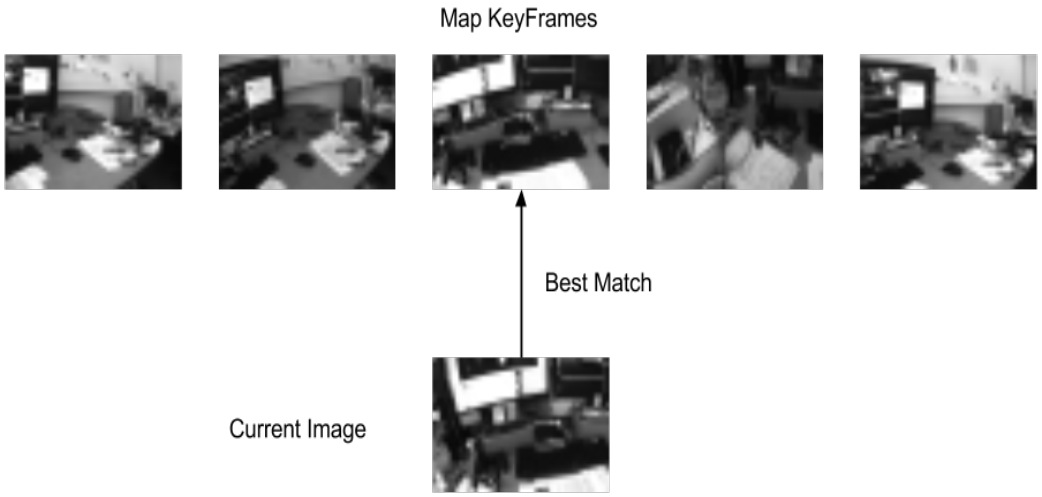


FIGURE 5.2: An example of the blurred image re-localisation technique used in PTAM.

and the current image a whole image alignment approach is used. The blurred, down-sampled and normalised keyframe descriptors are aligned using a direct second-order minimisation. This approach has the the advantage of computational efficiency and demonstrates good performance provided the density of keyframes remains high and the difference in viewpoint between keyframes and re-localisation frame is low. It does rely on retaining the original keyframe images which given PTAM does this for map-point matching is not a significant encumbrance. An example of this approach is shown in Figure 5.2

Visual re-localisation not only improves tracking robustness and facilitates map re-use but it can also facilitate multi-camera Visual SLAM. Given that it is possible to re-localise a camera within an existing map it is trivial to re-localise an arbitrary cameras within that map provided the map data is shared. The process is as follows a single MAV creates an initial map m which is shared between all MAVs. The other MAVs, under the assumption they are observing the same scene, use place recognition to re-localise themselves within the existing map m. All MAVs are now localised within the shared coordinate system defined by m and can proceed to localise and map further. As far as the author is aware this multi-camera re-localisation approach has not been previously employed for multiple disconnected cameras on the scale presented in this work. Castle et al. [11] make a modified version of PTAM to localise a single human equipped with multiple non-overlapping cameras for an augmented reality application. The idea of localisation via multiple cameras on a single MAV was also applied by Harmat et al. [38] for their MCPTAM system. Harmat et al. demonstrate the benefits of a multi-camera platform for performing tasks such a take-off and landing manoeuvres with full visual feedback. This work demonstrated improvements in localisation accuracy (fusing multiple visual pose estimates from each camera) and robustness (tracking failure of a single camera does not affect the platform as a whole). The drawback of the multiple cameras attached to a single platform approach is the increased platform

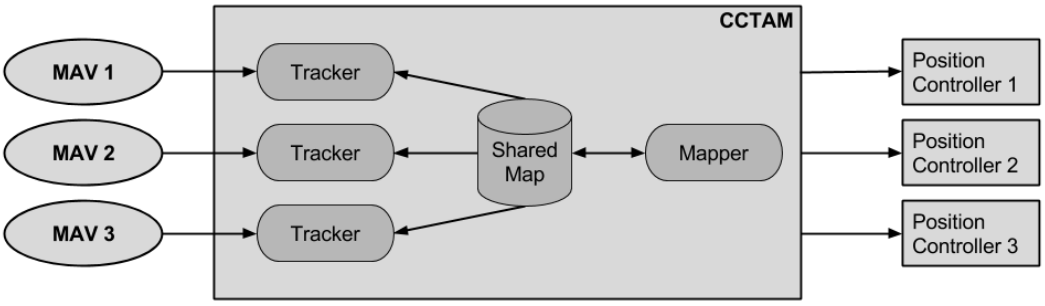


FIGURE 5.3: A high level overview of the CCTAM Framework. The main components are the MAVS which provided sensor data images to CCTAM which computes state estimates for the MAVs based on this data. This is passed to the position controller which compute suitable control commands to send to the MAVs.

size and increased computational requirements for multiple non-overlapping cameras. In this author's opinion this somewhat negates the main advantage of monocular cameras, namely the low weight and low power consumption. This Chapter focuses on multiple disconnected cameras and explores the idea of using this approach to enable large teams of MAVs to perform cooperative tasks such as collision avoidance and exploration autonomously.

5.3 Framework Overview

The centralised visual navigation framework presented in this Chapter is a complete solution for the autonomous control of a small team (up to six) of camera-equipped MAVs. The main components are: (i) a multi-robot visual slam system referred to as the Centralised Collaborative Tracking and Mapping (CCTAM) system, (ii) an Extended Kalman Filter (EKF) based state estimation system and (iii) a Proportional Integral Derivative (PID) based position controller. Whilst this framework targets a specific platform (the AR.Drone) the system is generalisable to a platform with a similar sensor suite, namely a camera and Inertial Measurement Unit (IMU) (this will be shown in Chapter 5).

Figure 5.3 provides a high level overview of the framework presented. Each MAV provides CCTAM with both sensor data for the EKF as well as camera images for Visual SLAM. Each tracker runs as a separate thread within the CCTAM process. The multi-threaded approach allows each tracker to potentially run concurrently at full frame rate, 30 Hz (depending on camera frame rate and team size see Section 5.8.5). The camera images and sensor data are processed by CCTAM which provides a global state estimate (fused with inertial sensor data) for each MAV to their respective position controllers. The position controllers use the state estimates and user given goals to compute the necessary control commands to send to each MAV. The following Sections will discuss the individual components of the framework in more detail.

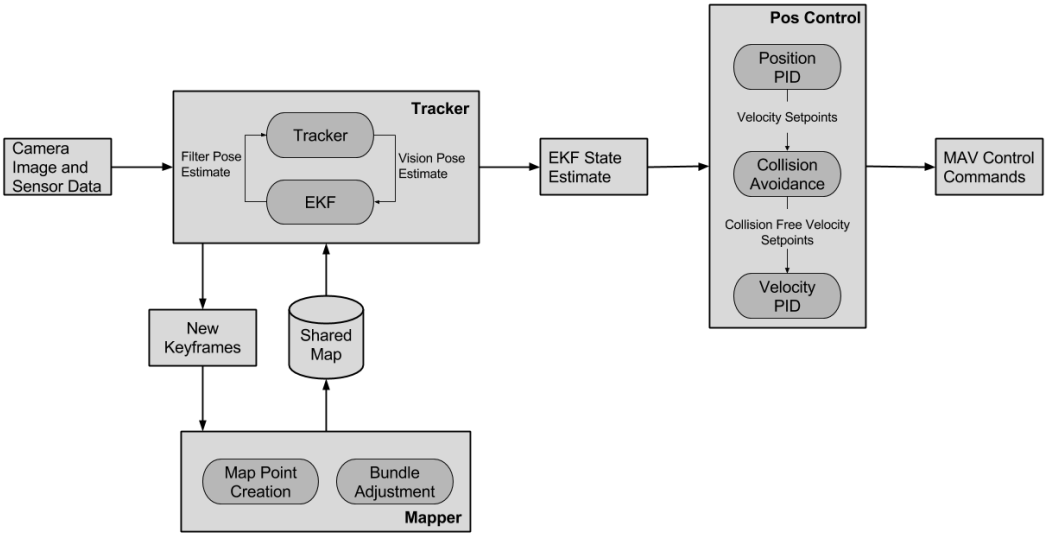


FIGURE 5.4: The processes for a single MAV in the CCTAM system.

5.4 CCTAM

CCTAM is based on PTAM as introduced in Section 3.2.3; the next Section provides more details of the original library as well as a discussion of the modifications made to the library for CCTAM. The main components of the system are the Trackers, the Mapper and the Map itself; each Tracker is responsible for frame-to-frame position tracking for each MAV whereas the Mapper is responsible for map building and optimisation. An overview of the system is given in Figure 5.4, for clarity this diagram details the components of the system for a single MAV, for multiple MAVs these components are duplicated (except the shared map and mapper).

5.4.1 The Map

The map ($\mathcal{M}$) in CCTAM consists of a set of keyframes ($\mathcal{K}$) and a sparse set of map-points ($\mathcal{P}$).

$$\mathcal{M} = (\mathcal{P}, \mathcal{K})$$

Full details of the map structure not relevant to the work presented in this Chapter, therefore we defer a full description to the next Chapter (see Section 6.3.1).

5.4.2 Tracking

As in the original PTAM [54], the tracker is responsible for real-time camera pose estimation and selecting the keyframes to be used for map construction. Figure 5.5 describes the tracking procedure in detail. To start the tracker requires an estimate of the current camera pose, in the original PTAM this estimate is generated by applying a decaying velocity motion model to the previous camera pose estimate. In CCTAM a more accurate estimate of the current pose provided via forward prediction using the EKF and

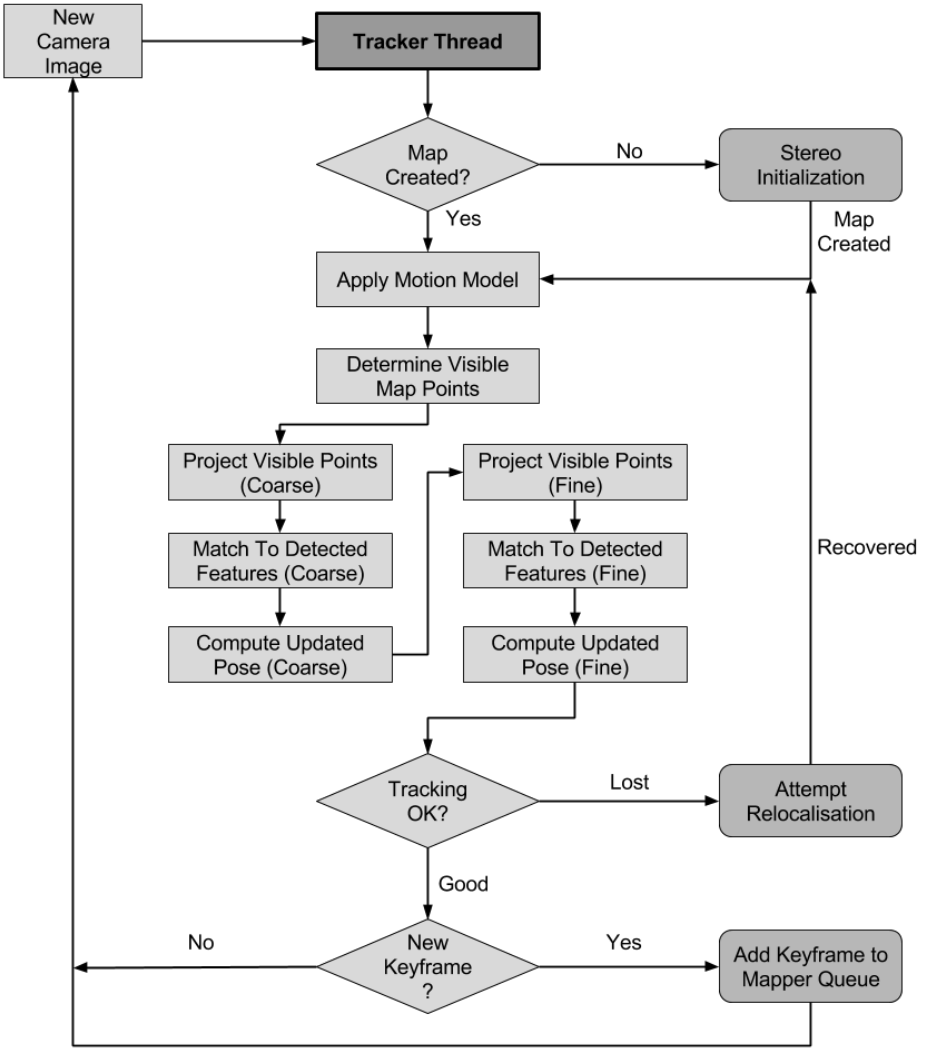


FIGURE 5.5: The CCTAM tracking process

MAV motion model (described in Section 5.5.3). From this pose estimate the tracker then determines which map-points should be visible in the current camera image. This procedure has the largest effect on the tracking runtime as it scales linearly with the size of the map. However as will be demonstrated in practical experiments tracking time remains near constant for map sizes of ≈ 300 keyframes and ≈ 20000 map-points. This is a sufficient size with which to map an area of 20×20 metres (also demonstrated in the experimental evaluation).

The tracker will then create an image pyramid from the current frame and extract visual features at each level of the pyramid. CCTAM makes use of the AGAST [73] detector to extract corner features for each level. As detailed in Section 2.3.1 it is a more general detector than the original FAST and as such does not require re-training to maximise performance. Additionally the dual decision tree approach makes the detector more robust to self-similar structures. Self-similar structures such as repeating patterns

in indoor environments or grass and tarmac in outdoor environments present a problem for Visual SLAM as they may lead to bad correspondences. The AGAST detector with its two tree approach provides some level of robustness to such structures.

The tracker uses a two-stage tracking process; first a small set of map-points (50-100) which should appear in the coarsest levels of the image pyramid are searched for. This is done using the estimated camera pose and 3D position of the map-point to re-project the point into the current image using the projection function described in Section 2.2. If an AGAST corner in found within small radius of re-projected image coordinates it is a possible match for the map-point. To verify this an 8×8 patch around the detected feature point is compared to the corresponding patch in the source keyframe for that particular map-point. However because the viewpoint may have changed from the original keyframe an affine warp is applied to the source patch [54]. The affine warp matrix A is given by:

$$A = \begin{bmatrix} \frac{\partial u_t}{\partial u_s} & \frac{\partial u_t}{\partial v_s} \\ \frac{\partial v_t}{\partial u_s} & \frac{\partial v_t}{\partial v_s} \end{bmatrix} \tag{5.1}$$

where (u_s, v_s) are the pixel coordinates of the source pixel and (u_t, v_t) are the pixel coordinates of the target pixel. This is computed by projecting unit pixel displacements from the plane of the source patch to the current target frame. Determining in which pyramid level a map-point should be searched for is done by taking the determinant of the matrix A; the determinant corresponds to the area of the patch in square pixels that a source pixel occupies in the original image resolution. Therefore if 4 pyramid levels are used the correct pyramid level to find the patch is given by $det(A)/4$. The warped patch is then compared to the target patch using the Zero-Mean Sum of Squared Differences (ZSSD) to provide some robustness to lighting changes. This procedure is repeated for all AGAST features within a small region of the predicted image coordinates and the feature with the lowest ZSSD that is beneath a predefined threshold is taken as a match. Each match represents an observation of a map-point for which we have a estimate of the 3D position. This gives a set of 3D world point to 2D image points this is exactly the perspective-n point problem described in Section 2.4.1. To obtain the most accurate solution the linear solution is not employed; instead the problem is solved by using non-linear least squares, minimising the sum of the re-projection error as discussed in 2.4.4. To improve robustness to outlier the standard re-projection error is replaced by one of the robust cost functions described in Section 2.4.8. In this work we used the Tukey cost function however in our tests similar performance is obtained with the Huber and Cauchy cost functions. Once a pose update has been successfully computed on the small set of coarse features a fine-grained search is carried out on a larger set of points (1000-5000) from all pyramid levels. After the final pose update is complete the tracking quality is assessed to determine if re-localisation is necessary. The tracking quality heuristic is based on the fraction of successful observations of map-points which should be visible in the current frame. Tracking quality is also used to determine if a

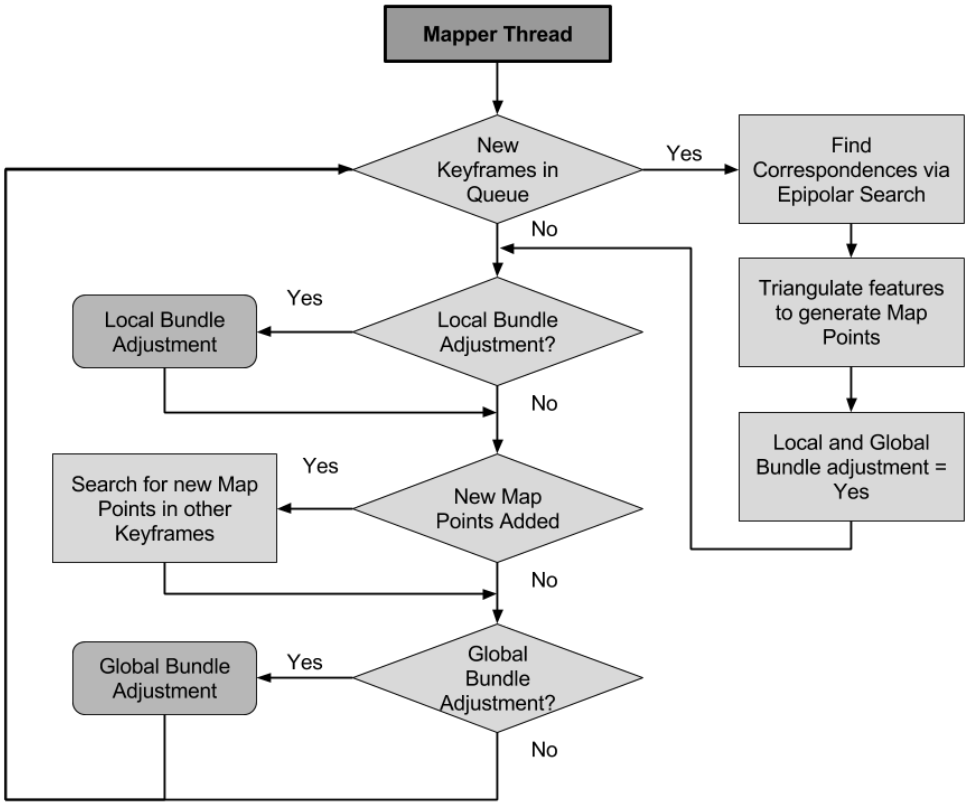


FIGURE 5.6: The CCTAM mapping process

new keyframe should be added to the map. If the tracking quality is high enough and the distance to the nearest keyframe is sufficient a new keyframe will be added to the map.

5.4.3 Mapping

The mapping thread is responsible for building the initial map using a stereo initialisation process and then further extending the map using the keyframes provided by the trackers. Figure 5.6 describes the mapping procedure in detail. When the mapper receives a new keyframe it will find the closest keyframe (by euclidean distance) and search for candidate map-points at each pyramid level by epipolar search (see Section 2.4.3). Candidate points that are too close to existing map points are then removed to avoid duplication. Finally the 3D position of the points are triangulated and the new map-points are added to the map. The Mapping thread will then run a local bundle adjustment on the new keyframe and it's four nearest (spatially) neighbours. If there are no new keyframes to process the Mapping thread will also run a complete global bundle adjustment optimising all keyframe poses and map-points.

The stereo initialisation procedure in the original PTAM, assumes user input to capture the two initial frames. The user presses a button to start the procedure and

the first frame is captured. Corner features are extracted and tracked in subsequent frames using a simple frame-to-frame tracking approach. The user then presses the button again once a sufficient baseline has been reached and a second frame is captured. The two frames and all feature correspondences are then passed on to the Mapping thread to bootstrap the map. The initialisation scene is assumed to be planar and the correspondences are known therefore it is sufficient to use compute a homography to describe the relative rotation between the two frames. The baseline between the two frames is assumed to be fixed (e.g. to 0.5 metres) this assumption is used to scale the initial map into metric space. This means the further the camera is from the assumed 0.5 metre baseline the poorer the metric scaling of the map. After this has been computed the first two keyframes are created; map-points can then extracted from these two frames to initialise the map. Finally if sufficient map-points are found a plane is fitted to the newly generated map-points using a RANSAC approach, this plane serves as the origin for the map coordinate frame with the z axis aligned orthogonal to the plane.

This initialisation procedure is difficult to perform automatically on-board a MAV however the additional sensors on board a MAV can be exploited in order to automate this procedure. The AR.Drone in particular features both an Inertial Measurement Unit (IMU) as well as an on-board metric optical flow approach similar to that described in Section 3.2.3. Combining these two sources in an Extended Kalman Filter (EKF) provides a noisy but complete state estimate. This allows us to fully automate the stereo initialisation procedure using the EKF and position controller to obtain two images spaced at a sufficient distance apart, as well as computing the actual metric baseline (instead of assuming a fixed one). The IMU measurements can also be used to align the map origin to the gravity vector without the need to find a plane using the RANSAC approach.

5.5 State Estimation

The state estimator in CCTAM is an Extended Kalman Filter (EKF) which fuses position estimates from the visual SLAM system with sensor data from the MAVs. The state of the EKF is given as:

$$\mathbf{x}_t = (x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, \phi, \theta, \psi, \dot{\psi}) \tag{5.2}$$

Where x_t, y_t, z_t represents the position of the quadcopter, $\dot{x}_t, \dot{y}_t, \dot{z}_t$ the linear velocity, ϕ, θ, ψ the attitude and $\dot{\psi}$ the angular velocity about the z axis. Our framework makes use of the sensor and motion model developed for the AR.Drone by Engel at al. [24]. The AR.Drone includes many sensors which provide partial observations of the state of the system, in the following Section the sensor observation models for each sensor are introduced.

5.5.1 Sensor Observation Model

The AR.Drone provides velocity estimates ($\dot{x}_t,\ \dot{y}_t$) computed from on-board optical flow using the small downward facing camera (scaled to metric units using on-board sonar sensor). These estimates are in the quadcopter body coordinate frame and must first be rotated to the global coordinate frame before integration into the state estimate. The attitude of the quadcopter ($\phi_t,\ \theta_t,\ \psi_t$) is computed on-board by fusion of the accelerometer and gyros. This provides a reliable estimate of roll and pitch angles but given there is no gravity vector to compensate for yaw drift, yaw estimates are less accurate. The AR.Drone does include a three axis magnetometer which can be used to estimate attitude based on the magnetic field of the earth. However the magnetic field of the earth is very weak in comparison to the many common sources of electromagnetic fields within a typical indoor environment; such as electrical cabling, metal support structures, electronic devices and so on. This makes the sensor very unreliable for indoor environments and the decision was made to ignore the magnetometer data. Instead yaw drift is compensated for by differentiating the yaw measurements and treating the yaw estimates as yaw velocities. Visual SLAM provides a reliable enough yaw estimate to account for any residual drift. Finally the vertical velocity ($\dot{z}_t$) is taken by differentiating the relative height measurements from the AR.Drones on-board sonar sensor. This is mainly due to the inaccuracy of the sonar sensor compounded by the numerous sources of acoustic interference found on-board a typical MAV such as propeller noise, frame vibration and air turbulence. The full sensor observation function $h_o(\mathbf{x}_t)$ is given as follows:

$$h_o(\mathbf{x}_t) = \begin{bmatrix} \dot{x}_t cos\psi_t - \dot{y}_t sin\psi_t \\ \dot{x}_t sin\psi_t + \dot{y}_t cos\psi_t \\ \dot{z}_t \\ \phi_t \\ \theta_t \\ \dot{\psi}_t \end{bmatrix} \tag{5.3}$$

And the measurement vector $z_{o,t}$ is given as

$$z_{o,t} = \left(v_{x,t}, v_{y,t}, \frac{h_t - h_{t-1}}{\delta_{t-1}}, \phi_t, \theta_t, \frac{\psi_t - \psi_{t-1}}{\delta_{t-1}} \right)^T \tag{5.4}$$

5.5.2 Vision Observation Model

During normal operation CCTAM provides the estimated camera pose with respect to the global frame. Therefore it is necessary to transform this estimate to the inertial frame of the quadcopter. The vision observation function is given as:

$$h_v(\mathbf{x}_t) = (x_t, y_t, z_t, \phi_t, \theta_t, \psi_t)^T \tag{5.5}$$

This is used as a direct observation of the global pose for each quadcopter and without differentiation (a common approach for single robot estimators with multiple pose sources), this is to ensure each MAVs pose is consistent with the common global coordinate frame. The measurement vector is given as:

$$\mathbf{z}_{v,t} = ({}^{\mathcal{IC}}\mathbf{H}{}^{\mathcal{WC}}\mathbf{H}_t) \tag{5.6}$$

Where ${}^{\mathcal{WC}}\mathbf{H}$ is the current camera pose in the world coordinate frame and ${}^{\mathcal{IC}}\mathbf{H}$ is the fixed transform from the camera to the inertial frame of the MAV.

5.5.3 State Transition Function

The work presented in this thesis directly uses the state transition function for the AR.Drone developed by Engel et al. [24] which approximates the acceleration and angular velocities based on the current state x_t and the current control command u_t. This linear prediction model is key to the performance of the EKF as data from the AR.Drone can be subject to significant delays (100-200 milliseconds). In particular usage is made of the linear prediction model of the influence of control commands u_t on the state of the AR.Drone, given by:

$$\ddot{x}(x_t) = c_1\mathbf{R}(\phi_t, \theta_t, \psi_t)_{1,3} - c_2\dot{x}_t \tag{5.7}$$

$$\ddot{y}(x_t) = c_1\mathbf{R}(\phi_t, \theta_t, \psi_t)_{2,3} - c_2\dot{y}_t \tag{5.8}$$

Where $\mathbf{R}(\cdot)_{i,j}$ denotes the ith and jth entries of the rotation matrix defined by ϕ, θ and ψ. The attitude and vertical velocity are approximated from the current state x_t and the current control u_t.

$$\dot{\phi}(x_t, u_t) = c_3\bar{\phi}_t - c_4\phi_t \tag{5.9}$$

$$\dot{\theta}(x_t, u_t) = c_3\bar{\theta}_t - c_4\theta_t \tag{5.10}$$

$$\dot{\psi}(x_t, u_t) = c_5\bar{\dot{\psi}}_t - c_6\dot{\psi}_t \tag{5.11}$$

$$\ddot{z}(x_t, u_t) = c_7\bar{\dot{z}}_t - c_8\dot{z}_t \tag{5.12}$$

Where $(c_1, \ldots, c_8)$ are model parameters which are tuned experimentally. The full state transition function is given as:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \\ \dot{x}_{t+1} \\ \dot{y}_{t+1} \\ \dot{z}_{t+1} \\ \phi_{t+1} \\ \theta_{t+1} \\ \psi_{t+1} \\ \dot{\psi}_{t+1} \end{bmatrix} \leftarrow \begin{bmatrix} x_t \\ y_t \\ z_t \\ \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \\ \phi_t \\ \theta_t \\ \psi_t \\ \dot{\psi}_t \end{bmatrix} + \delta_t \begin{bmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \\ \ddot{x}(\mathbf{x}_t) \\ \ddot{y}(\mathbf{x}_t) \\ \ddot{z}(\mathbf{x}_t, \mathbf{u}_t) \\ \phi_t \\ \theta_t \\ \psi_t \\ \dot{\psi}_t \end{bmatrix} \tag{5.13}$$

Sensor and image data from the AR.Drone is all received with varying delays, to combat this all sensor data is stored in a cache with the exception of image data which is immediately processed (due to the relatively high computation time of the visual SLAM). At each iteration the filter update procedure states with the EKF state after the last successful vision observation (that is assumed to be the most accurate). At this point a forward prediction of the state of the system is computing integrating all previous control commands and sensor data. The state of the system is predicted to a point slightly ahead of the current moment in time to account for the delay in computing and transmitting the control commands. This approach helps compensate for both the delays as a result of the wireless link to the AR.Drone as well as the lack on synchronisation between camera and the inertial sensors.

5.6 Trajectory Control

For trajectory control we employ the collision avoidance trajectory controller described in Chapter 4. The system takes as input a sequence of goal positions of the form $x_{des} = (x, y, z, \theta)$ the position control layer computes the desired velocity for each controllable axis from the position error. A velocity obstacle based collision avoidance scheme is used. It takes the desired velocity set point and computes the closest collision free velocity. These are then fed into the velocity PID controllers which compute the desired acceleration for each degree of freedom based on the velocity error.

The AR.Drone control commands take the form of tilt angles for the roll and pitch axes. The desired acceleration vector output from the PID controllers is converted to an angle command using the following formula:

$$\phi_{cmd} = \frac{1}{g}(\ddot{x}\ sin\ \psi_t - \ddot{y}\ cos\ \psi_t) \tag{5.14}$$

$$\theta_{cmd} = \frac{1}{g}(\ddot{x}\ cos\ \psi_t - \ddot{y}\ sin\ \psi_t) \tag{5.15}$$

where ϕ_{cmd} and θ_{cmd} are the roll and pitch commands respectively, ψ_t is the current yaw angle, $\ddot{x}$ and $\ddot{y}$ are the linear accelerations for the x and y axes respectively and g represents the earth gravitational constant. The AR.Drone controls for the yaw and vertical axis take the form of velocity commands so we can directly pass the velocity set-points from the position PID controllers.

MAV Radius Scaling

In order to account for any uncertainty in the position estimates provided by CCTAM a two factor scaling approach to account for both localisation uncertainty and excess communication delays. The localisation source of the trajectory controller is the EKF described in the previous section which fuses pose estimates from CCTAM with inertial sensor data. The EKF not only provides pose information but also an estimate of the certainty of each estimate via the covariance matrix. The radius of each MAV r_{mav} is used to determine the size of each velocity obstacle, therefore the radius can be used to account for the position uncertainty (increasing the assumed radius of the MAV when uncertainty increases). To achieve this the size of the one sigma uncertainty ellipse of the $x-y$ position given by the EKF is determined. The MAV radius is then scaled such that a robot of radius r_{mav} at the extreme edges of the uncertainty ellipse is encompassed by the new radius r_{unc}.

5.7 Implementation

The original PTAM library was designed to run two concurrent threads; the Mapper and the Tracker and as such the data structures were designed to support this mode of operation. In particular the tracker maintains a data structure for each map-point which handles point re-projections. In order to maintain data consistency mutual exclusion is enforced for the Trackers at the map-point level. Each Tracker must gain exclusive access to all map-points it requires. This is done after it calculates which map-points should be visible in the current frame, this ensures that Trackers only obtain a lock on map-points required for tracking. The Mapper thread also modifies the map, specifically the keyframe and map-point poses and positions when performing local and global bundle adjustment however, these operations can take several seconds particularly the global bundle adjustment. Therefore to ensure real-time operation can be maintained the Mapper will copy the map data structure and perform bundle adjustment on the copied map. Once the bundle adjustment is complete the Mapper then takes exclusive access of the map structure to update the keyframe poses and map-point positions (a very fast operation).

Our framework has been implemented in C++ and integrated into the Robot Operating System (ROS) [93], each main component within our system has been implemented as a separate ROS node.

5.8 Evaluation

In this Section the experiments conducted on the CCTAM framework are presented. This Section is organised as follows, it begins with a description of the experimental setting including the simulation and hardware platforms. Section 5.8.3 describes the localisation experiments performed on the CCTAM framework, Section 5.8.5 goes on to expand on the experiments performed to determine the scalability of the framework. And in Sections 5.8.6 and 5.8.7 CCTAM is used to perform two multi-agent coordination tasks, collision avoidance and exploration and the results from these two sets of experiments is presented.

5.8.1 Simulation Environment

The simulated environment constructed for these experiments is based on the the Gazebo multi-robot simulator. Gazebo provides capabilities to model complex 3D environments, reliably model multiple flying vehicles and generate realistic sensor data including camera images. Gazebo also integrates easily with the Robot Operating System (ROS) meaning CCTAM can be run on both simulated and real robots without altering the framework.

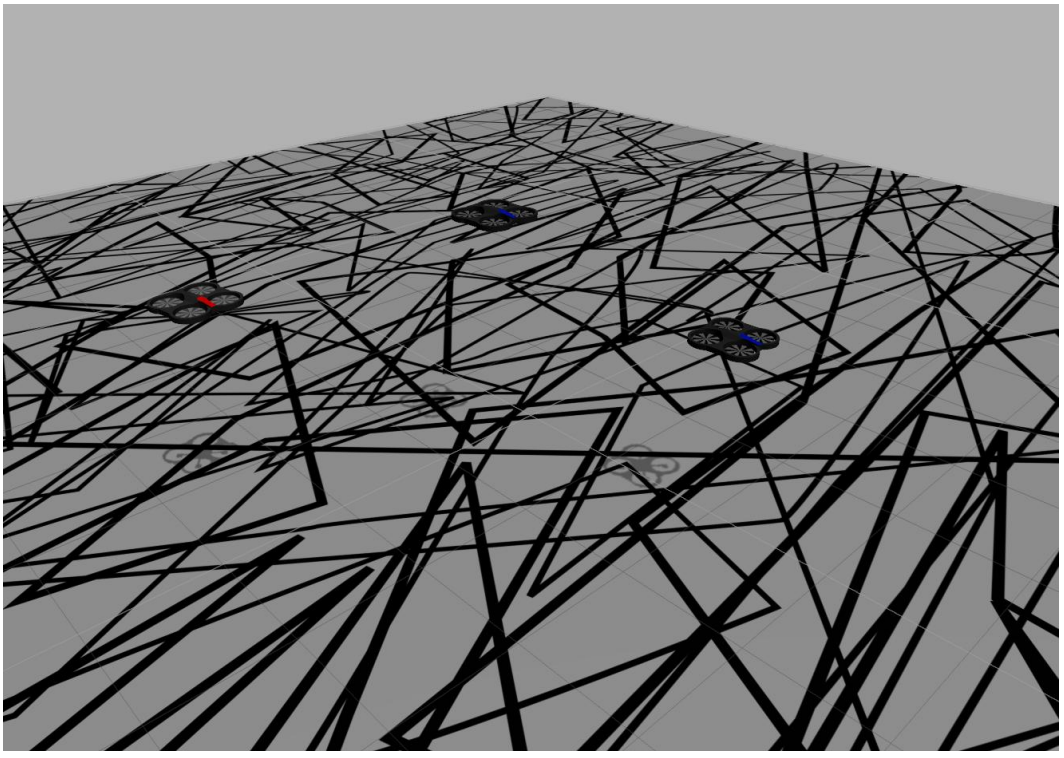


FIGURE 5.7: The simple simulation world

Meyer et al. [77] introduced a number of UAS-specific sensor plug-ins for Gazebo such as barometers, GPS receivers and sonar rangers. The work in this thesis is focussed more on accurate sensor modelling rather than flight dynamics and as such our simulator uses the simplified flight dynamics model introduced in Section 2.6.1. The sensor plug-ins

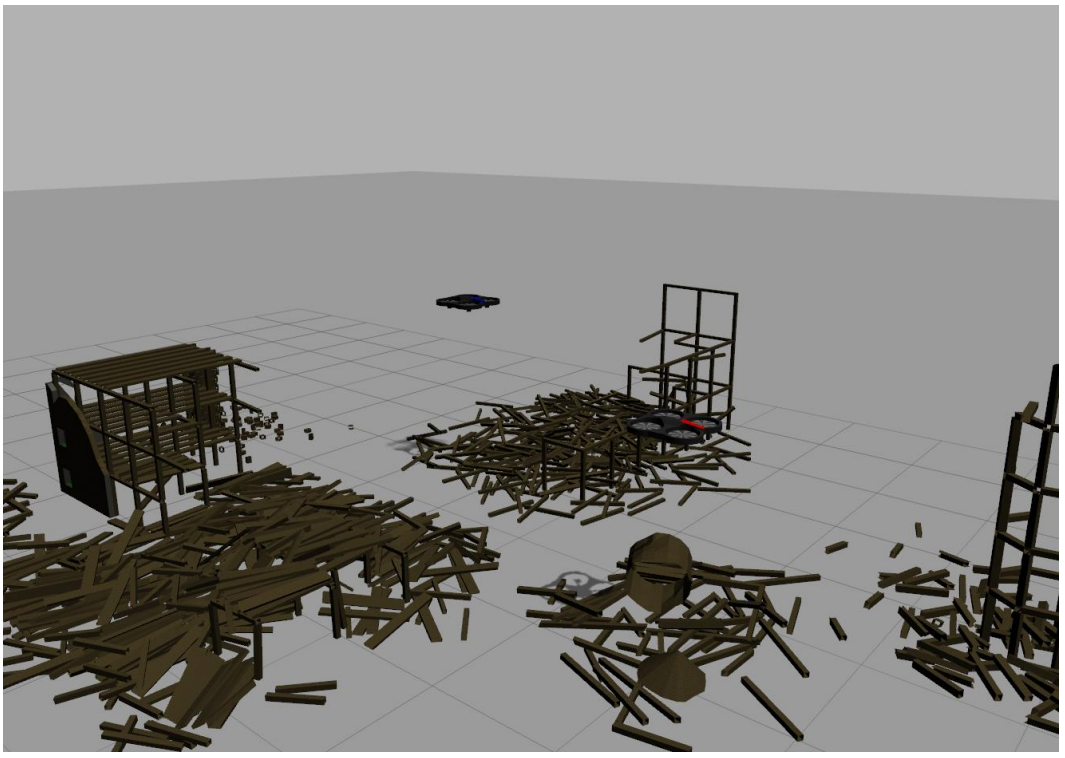


FIGURE 5.8: The simulated disaster world

developed by Meyer et al. were used to replicate the sensor suite on the AR.Drone. The standard camera plug-in within the Gazebo simulator is used; this plug-in featured the ability to add sensor noise (modelled as a zero-mean Gaussian), however it did not support lens distortion modelling. This was not critical as real hardware experiments are performed and these serve the validate the approach with real world lens distortion.

In addition to the MAV two simulated worlds were developed to conduct experiments; the first a very simple world consisting of a flat 20×20 metre plane, the plane is textured to ensure enough visual features are available for localisation. The second environment consists of a 20×20 metre plane on top of which are placed models of damaged buildings to simulate a disaster site. Figures 5.7 and 5.8 show the two simulated worlds. The aim of these simulated worlds is not to create fully realistic environment but instead facilitate large scale, real time experimentation with MAVs with an accurate sensor model. The simple environment models make scaling the simulation to larger numbers of MAVs feasible as the more complex the environment the more processing time is required to generate images for the simulated cameras. Validation of the system in a realistic environment is covered by the hardware experiments performed. The simulated experiments conducted presented in this Section did not involve a realistic communication model i.e. no delay was added to inter-component communication to simulate a wireless network, this is investigated in detail in Chapter 5. All simulated experiments were run on a desktop computer with a 3.4 Ghz Intel i7 processor and 16 GB of RAM (this also served as ground-station computer in the later hardware experiments).

In later Sections the speed of the MAVs is discussed; for clarity it should be mentioned that in general the maximum speed of the MAVs for simulated experiments was set at 5 metres per second. For real world experiments where limited space was available a maximum of 2 metres per second was used.

5.8.2 CCTAM Hardware Platform

For these experiments two Parrot AR.Drone 2.0 MAVs are used; a very lightweight (400 grams), low cost MAV (£250) platform. It is constructed primarily out of Expanded Polypropylene (EPP) foam with a carbon fibre reinforced central cross providing the frame with rigidity and mountings for the motors and electronics. The drone features two cameras, the front facing camera is capable of streaming images at 640×360 and is fitted with a 92° wide angle lens. The downward facing camera is primarily used for the on-board optical flow and captures images at 160×120 which is up-scaled to 640×360 for streaming. The on-board sensors include an Inertial Measurement Unit (IMU) attitude as well as a sonar sensor (for low altitude) and barometric pressure sensor for altitude sensing. The on-board processor is an 1 Ghz ARM Cortex A8 processor running a custom version of the light footprint Busybox Linux distribution. Finally the drone features an 802.11 N wireless network interface running on the 2.4 Ghz band for control and video streaming.

The work in this thesis assumes the MAVs use a downward facing camera as it provides the best viewpoint for localisation and sensing as well as allowing the simplification of some planning tasks to 2D (see Chapter 6). Therefore for these experiments the AR.Drone was modified to have the front facing camera point downwards. The resolution of the downward facing camera is too low and the field of view to narrow for reliable localisation, particularly with the image artefacts introduced when the image is up-scaled to 640×360. This is a very straightforward modification due to the modular structure of the AR.Drone. The drone communicates via 802.11n WiFi which is used to stream control and sensor data as well as a single compressed camera feed. The high bandwidth of the information stream leads to communication latencies, especially when multiple drones are deployed at the same time. Additionally the choice of the 2.4 Ghz frequency band can lead to more issues as 2.4 Ghz is one of the most commonly used public frequencies for not only wireless computer networks but also other devices such as security cameras, cordless telephones, blue-tooth devices and even microwave ovens. More details on the communication limitations of the AR.Drone will be presented in Section 5.8.6.

5.8.3 Localisation Performance

An important factor in the performance of the CCTAM system is the localisation accuracy. This determines the precision which with the MAVs can navigate within the environment. To verify the performance of the CCTAM framework in terms of localisation performance and consistency. For the first experiment a CCTAM map was constructed

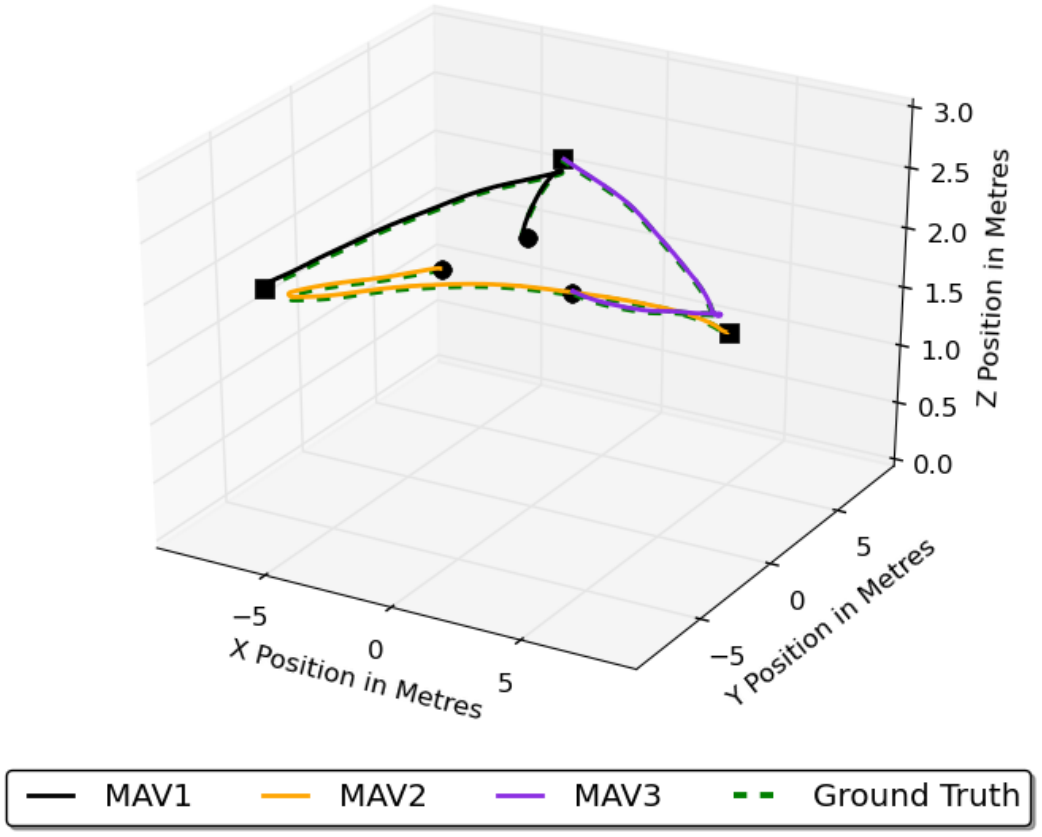


FIGURE 5.9: A representative example of the CTAM localisation experiments. The dashed green line represents the ground truth trajectory and the solid coloured lines represent the trajectory for each MAV.

for the simulated environment manually. The CCTAM map origin is aligned with the simulated ground truth origin which enables us to directly compare the recorded trajectories. A team of MAVs are commanded to fly to a set of predefined way-points, following a 20 metre long path, using the full CCTAM framework for localisation and control. For each execution the estimated trajectory for each MAV is compared to that of the simulated ground truth data. The Root Mean Squared Error (RMSE) metric was chosen to compare trajectories as it captures the position error over the length of the trajectory. The RMSE for a trajectory is given as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}$$

where n is the number of points in each trajectory, x_i is the ground truth position at point i and $\hat{x}_i$ is the estimated position at point i. GPS based navigation, with a typical error of 2.5 metres, is sufficient for large, open airspace with little or no obstacles but insufficient to operate close to buildings and other obstacles. Other localisation

TABLE 5.1: Localisation Performance Experiment Summary

Team Size	**Mean Trajectory Length**	**Mean RMSE**	**STD Deviation**
1	30.0	0.096	0.094
2	28.0	0.104	0.063
3	25.322	0.100	0.079
4	22.142	0.124	0.122
Mean	**26.366**	**0.106**	**0.09**

approaches such as laser based approaches typically have an RMSE of ≈ 0.06 metres. For this work an RMSE of ≈ 0.1 metres is targeted; this represents similar localisation performance to many single robot Visual SLAM systems [24, 28]. Additionally an overall error of 0.1 metres along a trajectory is sufficient to achieve the tasks described in this thesis (collision avoidance and exploration) as well as other tasks such as target tracking. Figure 5.9 shows an example of the results obtained; in this experiments the average RMSE for the 3 MAVs in this experiment was 0.09 metres.

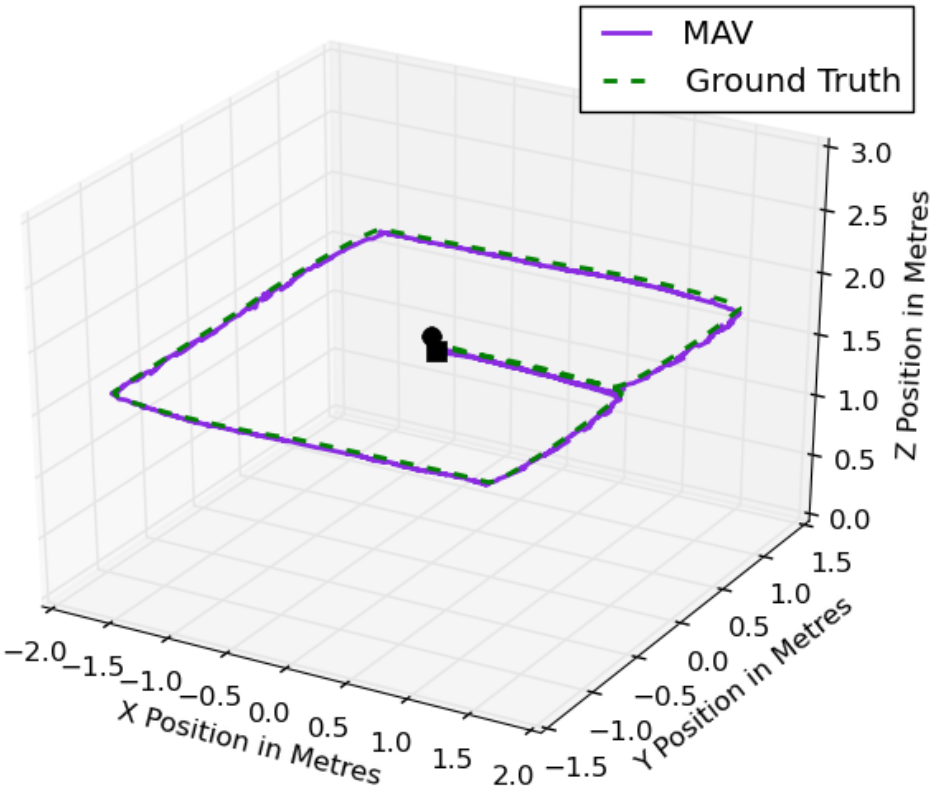


FIGURE 5.10: The result of the hardware localisation experiment, the ground truth trajectory is shown as a dashed green line and the esitmated trajectory as a solid purple line. The RMSE for this experiment was 0.10 metres.

For CCTAM these localisation experiments are conducted for team sizes from 1 to 4 MAVs and for each team size the experiment was repeated 50 times. Table 5.1 shows a summary of the results. The results show CCTAM system is able to maintain a typical

RMSE of less than 0.1 metres. Experiments are also conducted on real hardware, where an Optitrack motion capture system was used to provide ground truth data. Here due to limitations in the size of our motion capture lab only a single MAV is used however this is sufficient given the main aim of this experiments is verify localisation performance under realistic conditions (i.e. real world images and lens distortion). These experiments required the MAV to follow a square shaped path; the experiment was repeated 20 times and representative example the results are shown in Figure 5.10. The mean RMSE for these experiments was 0.10 metres.

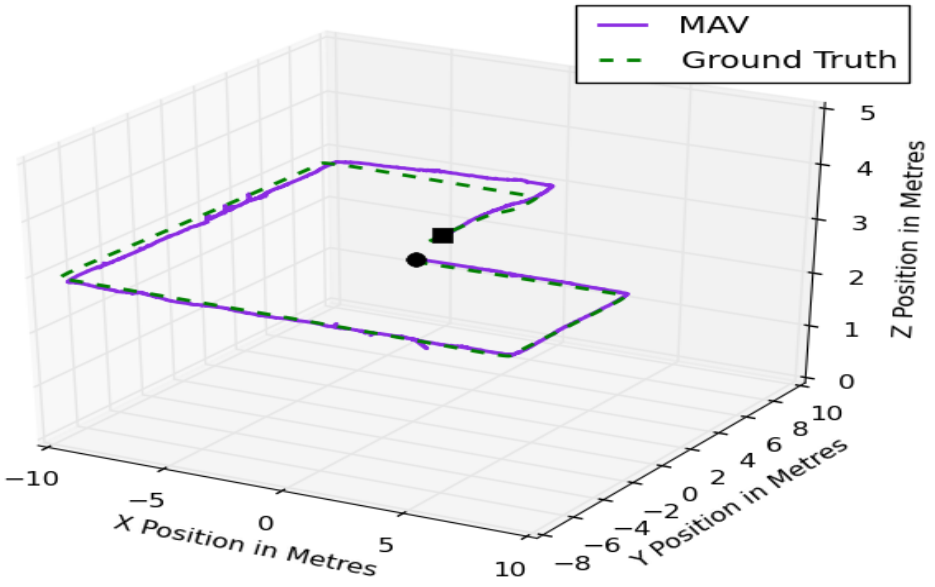


FIGURE 5.11: An example of localisation performance along a long trajectory. The trajectory length was 66 metres and the total accumulated drift was 0.66 metres.

5.8.4 Drift Analysis

One of them main limitations of CCTAM in terms of a complete Simultaneous Localisation And Mapping (SLAM) solution is the lack of loop closures (introduced in Section 3.2.3). Loop closures provide a method to correct the overall drift error introduced by the incremental motion estimation approach used by most SLAM systems. In order to determine how important this was for the work in this thesis two experiments were conducted to analyse this drift. In the first experiment a MAV was flown along a long trajectory (continuously localising and mapping) before returning to it's starting location. The results of this experiment are shown in Figure 5.11; the length of this trajectory was 66 metres and the position drift accumulated over the length of this trajectory was 0.66 metres. In this case the drift was significant enough to cause tracking to fail completely (at the point marked with a square in the Figure). The combination of the robust tracking approach and map optimisation via bundle adjustment mean CCTAM is able to implicitly close small loops. This is shown in the second experiment where a MAV flies a similar trajectory to the first experiment but half-way it crosses a

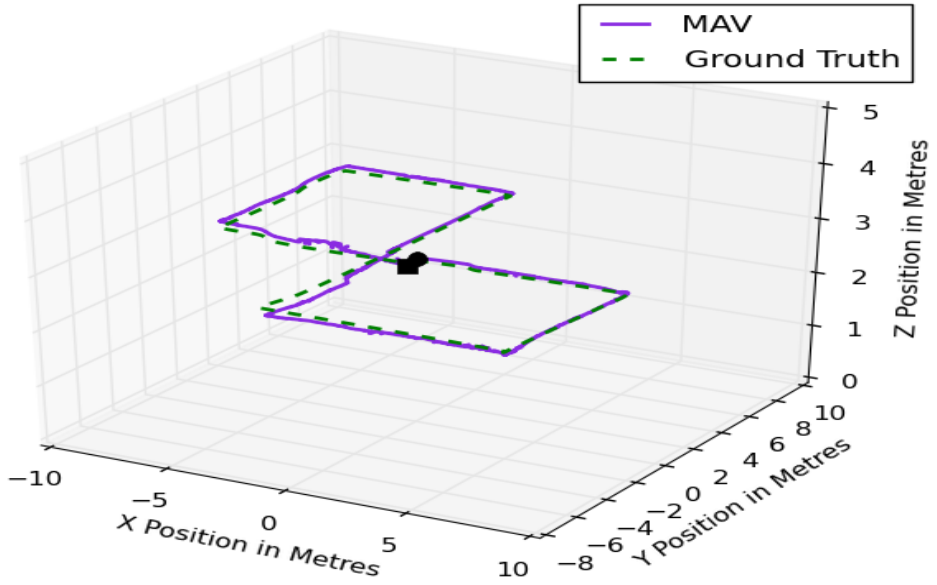


FIGURE 5.12: An example of localisation performance along a long trajectory with short loop closures. The trajectory length was 64 metres and the total accumulated drift was 0.05 metres.

previously mapped area. In this case the accumulated drift is small enough that CC-TAM is able to close the loop, this allows the system to maintain tracking and allow the MAV to complete the experiment. The results of this experiment are shown in Figure 5.12, the length of this trajectory was 64 metres and the final position drift was only 0.05 metres.

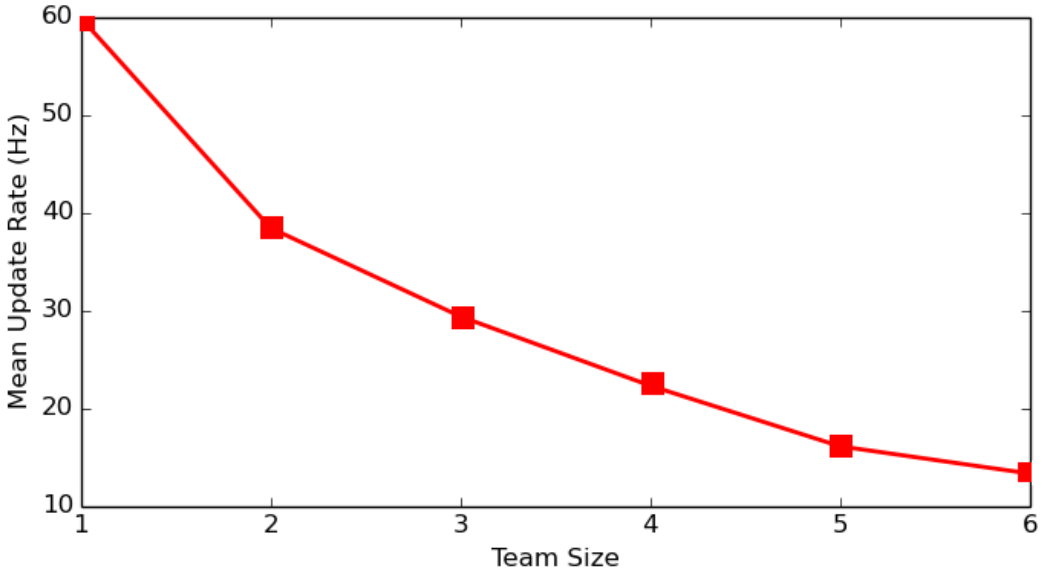


FIGURE 5.13: This graph plots the average Tracker update rate as it changes with the size of the team.

5.8.5 Scalability

In order to test the scalability of the CCTAM system the experiments from the previous Section were repeated, teams of MAVs flying a pre-defined path while constructing a map, while tracking the update rate of each tracking thread. Figure 5.13 shows how the average Tracker update rate as it changes with the size of the team the simulated camera frame rate is 60 Hz. These graphs demonstrate the limitations of the CCTAM system in terms of scalability. For the work presented in this thesis it was found reliable operation is feasible with an update rate of 20 Hz (i.e. teams of 4 MAVs). With team sizes of greater than 4 MAVs an increase in tracking failures due to the low update rate become too frequent for the framework to reliably control the MAVs. This can be mitigated by reducing the maximum speed of the MAVs to compensate for the slower update rates but this solution is less than ideal as it limits the range and capabilities of the MAVs.

5.8.6 Multi-Robot Task 1: Collision Avoidance

A discussed in Section 5.6 the position controller developed for the CCTAM framework includes a MAV-to-MAV collision avoidance system based. Full details of this system are given in Chapter 6 however for the purposes of this experiment a brief description is included here. The controller is based on a reciprocal collision avoidance approach in which each MAV communicates their current position and velocity to all the other MAVs. If a MAV detects that it's current velocity will result in a collision with another MAV (within a pre-defined time horizon) the MAV will alter it's velocity in such a way that it takes half the responsibility for avoiding the collision. It is assumed the other MAV will do the same thus the reciprocal nature of the system. The key point for the purpose of these experiments is the requirements for each MAV to share their positions and velocities. In CCTAM the positions are reported in the global map frame, if the reported position are not consistent i.e. there is some drift between the coordinate systems of each MAV this will result in a collision.

The experiment conducted is as follows, firstly the MAVs use a pre-built map of the simple simulated environment, the MAVs start in a circular configuration. Each MAV is commanded to fly to the opposite side of the circle (via the centre) meaning without collision avoidance all the MAVs would collide with one another. As before the experiments are repeated for increasing number of MAVs and the experiment repeated 100 times for each team size. Figure 5.14 shows an example of the MAV trajectories recorded for a single test with 3 MAVs and Table 5.2 shows a complete summary of the results. All collisions in these experiments occurred as a result of localisation failures as a results of the reduced update rate for the larger teams. As stated before more reliable results can be obtained by reducing the maximum speed of the MAVs.

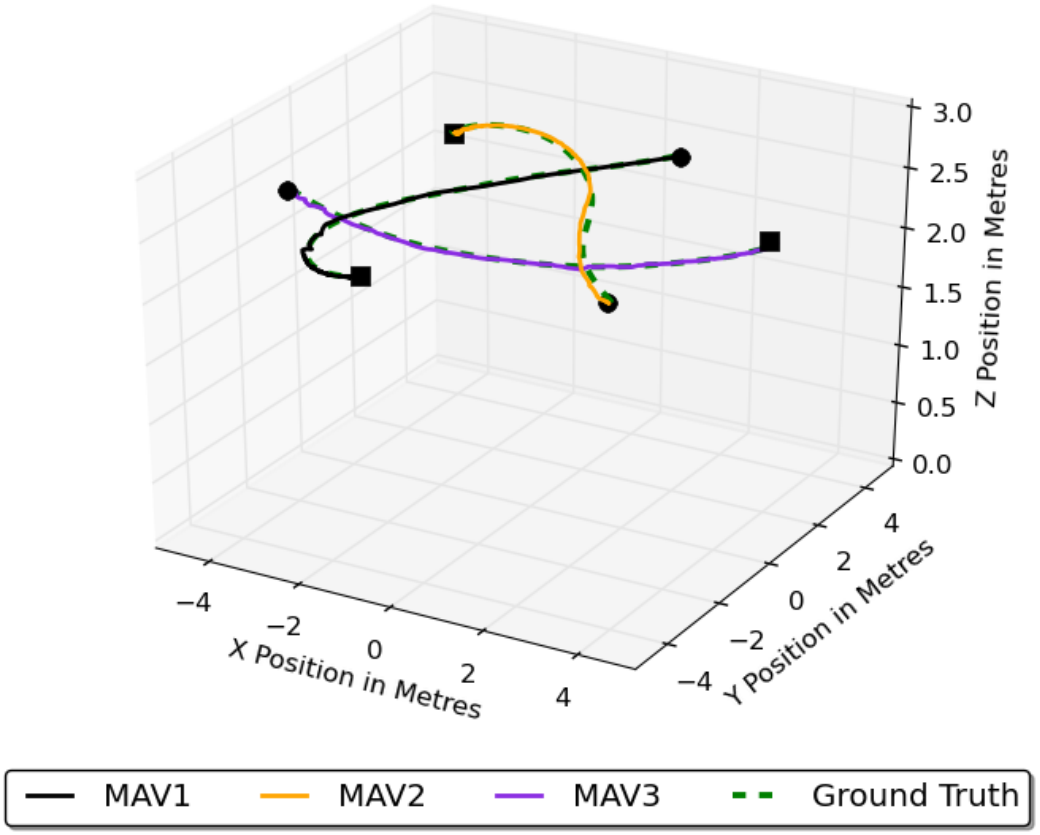


FIGURE 5.14: MAV trajectory plot for a collision avoidance experiment with 3 MAVs, the start location for each MAV is marked with a dot and the end location a square.

TABLE 5.2: Collision Avoidance Experiment Summary

Team Size	Simulated Pose	CCTAM Pose Estimate
2	100%	100%
3	100%	97%
4	100%	75%
5	100%	50%
6	100%	-

Real World Collision Avoidance Experiments

The collision avoidance experiment was repeated with two AR.Drones in an indoor environment as shown in Figure 5.15. The experiment was repeated 20 times and the resulting collision avoidance rate was 75%. This is the same collision rate for simulated team of 4 MAVs and an unexpected result given the good localisation performance on real hardware. Further investigation revealed the laboratory environment for the experiment exhibited an unusually high level of wireless network interference. Analysis of wireless network traffic revealed an unusually high rate of packet loss, this typically occurs when the frequency band is highly congested. The effect of this level of interference

was an increase in the variability of the rate at which camera images are received by the ground station computer at the worst there would be a delay of 500 milliseconds before a new camera image would be received. As already shown with the simulated experiments when the camera update rate drops sufficiently localisation becomes unreliable leading to collisions. This is a concern for this approach as there are many scenarios in which reliable wireless communication cannot be guaranteed, such as exploring an indoor environment in a disaster scenario.

FIGURE 5.15: Conducting collision avoidance experiments with real hardware

5.8.7 Multi-Robot Task 2: Exploration

In this set of experiments the performance of the entire framework is tested using the auction based exploration controller described in Chapter 6. In this experiment the MAVs start with no pre-built map and the experiment was performed the the disaster site simulated world. Using the exploration controller the MAVs autonomously explore the environment (with no pre-defined way-points) and attempt to construct a complete map of the environment. This experiment validates the performance of the mapping, localisation control and collision avoidance components of the framework.

As before experiments are performed with an increasing number of MAVs and repeated 50 times for each team size. In order to verify the accuracy of the reconstructed map-points produced by CCTAM a ground truth model was created using uniform sampling of the 3D model used in the simulation to produce a ground truth point cloud. Figure 5.16 (bottom) shows the map-points created by CCTAM and Figure 5.16 (top) the points extracted from the 3D model of the simulated world. The Iterative Closest Point (ICP) algorithm was then used to align the two point clouds and compute the final RMS error for the alignment. This is not a foolproof method of verifying the accuracy of

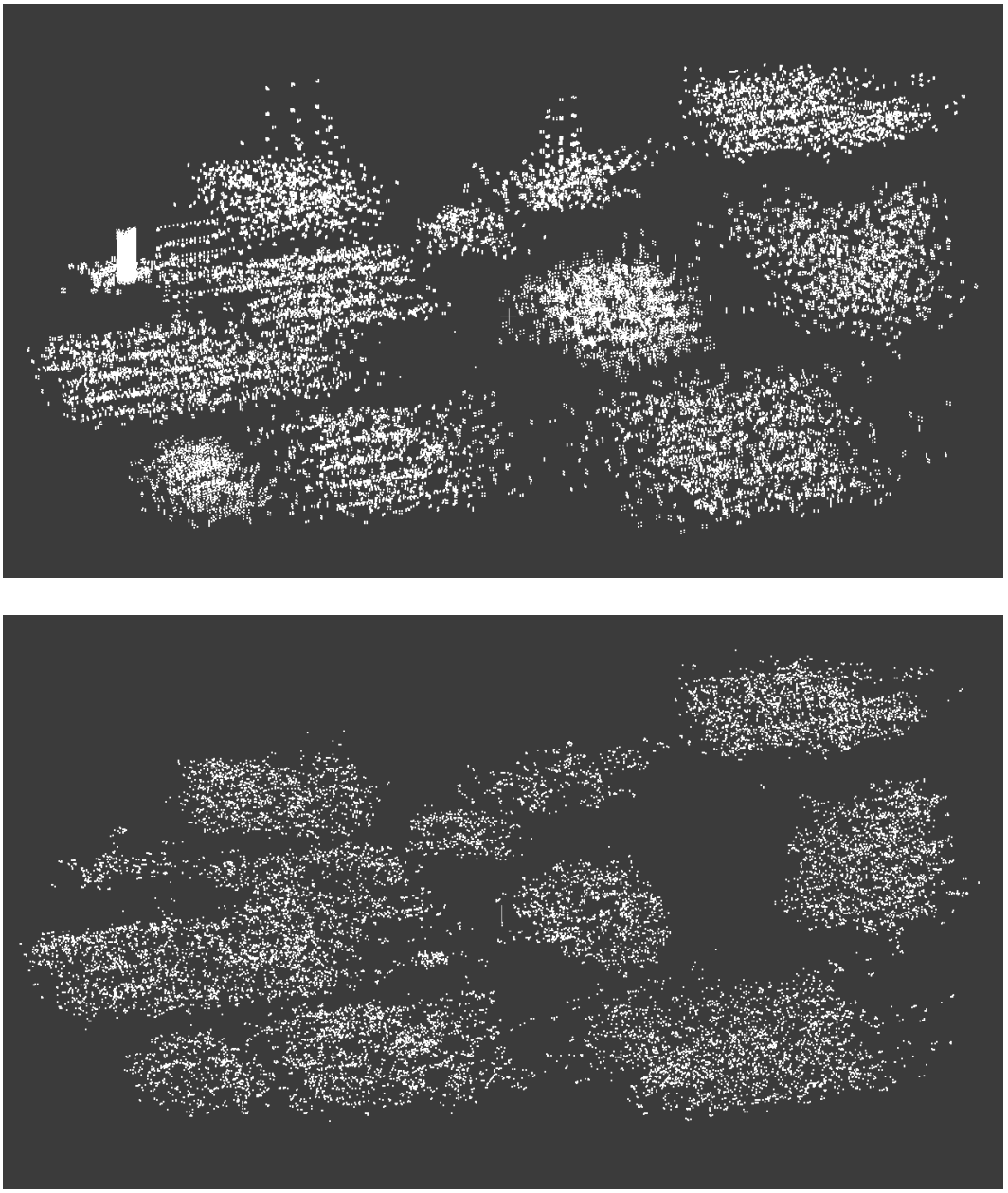


FIGURE 5.16: An example the ground truth model of the simulated environment (top) and the map-points produced by CCTAM (bottom). A comparison of these two point-clouds is used to determine the accuracy of the map produced by CCTAM.

TABLE 5.3: Exploration Experiment Summary

Team Size	**Mean Map Size**	**Mean Execution Time**	**Mean Alignment RMSE**
1	20751	900	0.105
2	21042	467	0.109
3	20053	353	0.110
4	20823	308	0.109
Mean	**20667.25**	**506.25**	**0.10825**

the map produced as the points produced by CCTAM do not match exactly to the points extracted from the 3D model (which are extracted by uniform sampling of the model surfaces). This approach is sufficient however to detect any larger inconsistencies in the map resulting from any drift in the pose of the MAVs. Table 5.3 presents a summary of the results collected for the experiment. The mean ICP error for CCTAM is 0.09 metres the corresponds to the localisation error seen in the previous localisation experiments. Again this does not provide an fully accurate measure due to the discontinuity between the points extracted from the simulator model and the points use for visual navigation.

From table 5.3 it can also be seen that increasing the team size has a significant impact on the time taken to completely map the 20×20 metre highlighting the main benefit of multi-robot systems. Another benefit is robustness to failure of a single MAV. As discussed earlier the tracking frame-rate degrades with larger teams which increases the chances of localisation failure. This occurred a number of times with a team size of 4 MAVs, however in all cases the remaining team was still able to complete the mapping task. This explains the slightly higher than expected run time for team size 4.

5.9 Conclusion

This Chapter has presented the CCTAM framework a centralised multi-robot visual tracking and mapping framework aimed at low-cost MAVs. The high level architecture of the framework was first introduced. The Chapter then continued with a description of the inner workings of each individual component. An experimental evaluation of the framework both in simulation and on real hardware was also presented.

The focus with this work was to determine the feasibility of using re-localisation to enable multi-robot visual navigation using low cost MAVs. As such there are many possible extensions or improvements to be made to the existing framework. As already mentioned the system does not include any loop closure capabilities. While we demonstrate that with the inclusion of an implicit loop closure even longer trajectories (60 metres) are possible with the system the lack of loop closure is a limiting factor if the application requires long trajectories. However this level of performance was believed to be sufficient for the work presented in this thesis and it was decided not to focus on loop closure and leave this as future work.

As discussed in the experimental evaluation the other limitation of the framework is the scalability of the centralised approach. This is a limitation of the architecture rather than the implementation; this will be demonstrated in Chapter 5. A related problem is the lack of robustness to delay in the Visual SLAM system, as demonstrated with the simulated experiments for teams of 4 MAVs or more and the real world experiments conducted in poor wireless conditions. In these cases the EKF delay compensation is not enough as, despite the fact that the EKF is still able to maintain a noisy estimate of the pose of the MAV (using Visual Odometry), the Visual SLAM system cannot continue mapping and is therefore unable to recover. It is the contention of this thesis that the

most robust and scalable solution is a distributed approach where Tracking is done on-board the MAV. This has several advantages including vastly improved scalability and improved robustness to delay. This will be demonstrated via an experimental evaluation of just such a distributed system which will be presented in the following Chapter, Chapter 5.

Chapter 6

Distributed Collaborative Tracking and Mapping

6.1 Introduction

In the previous Chapter a centralised approach to multi-robot visual navigation, based on the idea of re-localisation enabled single map approach, was presented. It was demonstrated, through a series of experiments, that while the performance of the approach in terms of localisation and mapping accuracy was sufficient for both indoor and outdoor control of a team of MAVs, the centralised approach was limited in scalability and robustness. In this Chapter a partially distributed approach to multi-robot visual navigation is presented. This approach features improved robustness and significantly increased scalability, while maintaining the same localisation and mapping performance.

The goal is to enable cooperative multi-robot navigation tasks using MAVs with very low on-board computing resources. It is desirable for the system to support as many MAVs as possible, operating simultaneously within the same area. These requirements lend themselves to a distributed system where computationally intensive tasks are run off-board on a more powerful ground-station computer. A distributed system operating in real-time over a wireless link places constraints on a system, as the wireless link can quickly become a bottleneck for the performance of the system. Therefore the interprocess communications of the proposed system must use as little bandwidth as possible in order to enable a larger number of MAVs to operate simultaneously in a reliable manner.

The three main contributions of the work presented in the chapter are as follows:

1. A partially distributed, multi-MAV visual navigation framework which is both scalable and features good localisation performance as well as robustness to wireless network delay.

2. A more general framework, in terms of the MAVs it supports. The work presented in the previous Chapter targeted a specific platform (the AR.Drone), the work in

this Chapter aims to make the framework applicable to different MAV platforms. This is done via a more flexible stereo initialisation procedure as well as making use of an more general EKF framework.

3. Two example hardware platforms, designed for use with the proposed system. The most important component of the hardware platform is the on-board computer, experiments were conducted to establish the performance of the proposed system on a number of popular on-board computers.

The remainder of this Chapter is organised as follows: Section 6.2 presents the general approach and Section 6.3 starts with an overview of the proposed system, before providing an in depth presentation of each of its main components. Section 6.5 discusses some additional components in the framework and Section 6.6 provides details of the implementation. The example hardware platforms developed as part of the work presented in this Chapter are presented in Section 6.7. The Chapter finishes with a presentation of the experimental evaluation in Section 6.8 and some conclusions in Section 6.9

6.2 Partially Distributed Visual Navigation

In this chapter we present a system for partially distributed multi-MAV visual navigation. The system is referred to as the Distributed, Collaborative Tracking and Mapping system (DCTAM). The DCTAM system is based on the Parallel Tracking and Mapping (PTAM) system developed by Klien and Murray[54], similar to CCTAM and several of the systems discussed in the previous section. In CCTAM the tasks of real-time motion estimation for each MAV and map creation/refinement were divided into separate threads running on the same computer. In DCTAM we divide these components into a distributed system where the tracking component operates on-board each MAV in parallel and the map creation/refinement component runs on a more powerful ground-station computer.

The key advantage of the approach presented in this Chapter is the ability run to the real-time critical components of the localisation framework on-board the MAV without having to limit the size of the map (mapping is still done on a ground station computer). In this Chapter we will demonstrate how the distributed architecture allows the system to maintain a high tracking frame-rate. This leads to improvements in terms of both tracking performance as well as robustness. Additionally, as tracking is no longer processed centrally the system features vastly improved scalability (up to 20 MAVs in our experiments). In contrast to the approach of Foster et al. we also ensure each tracker has a local copy of the map to ensure that drift-free tracking (within previously mapped areas) can still be performed even when the MAVs lose communication with the ground station. One of the key limitations of such a distributed approach is the communications overhead, if communication between nodes within the system is too expensive the benefits of a distributed approach is somewhat limited. The system presented in this

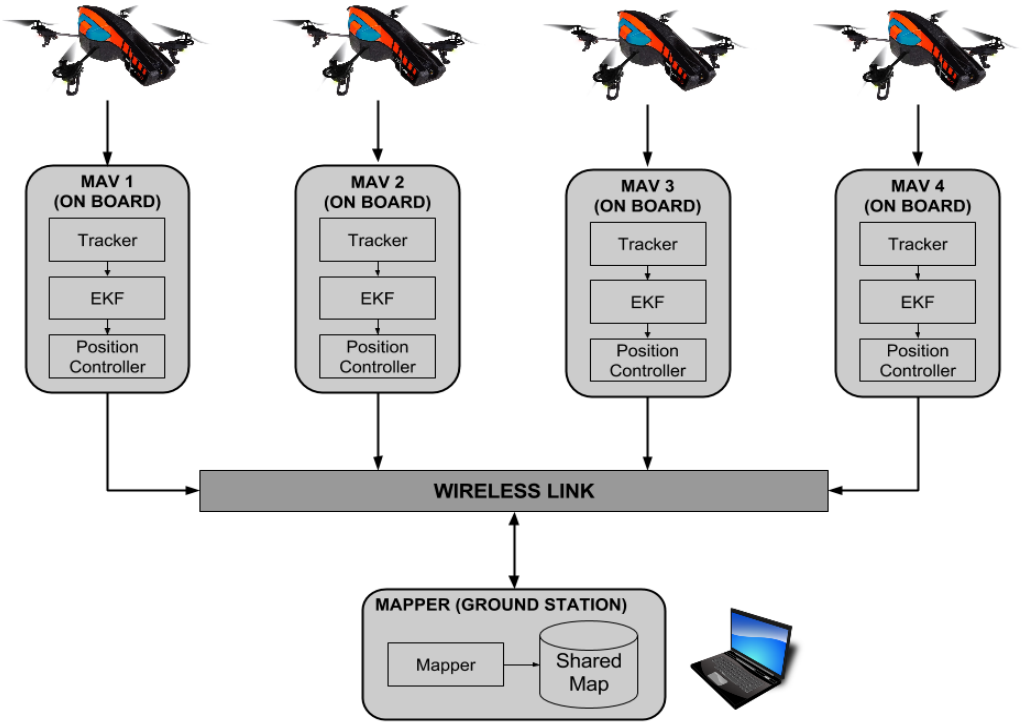


FIGURE 6.1: Basic system overview of the DCTAM system.

Chapter addressed this issue through the use of an effective communication model and the application of compression techniques. This not only ensures rapid communication but also serves to facilitate the improved scalability.

6.3 System Overview

An overview of the proposed DCTAM system is shown in Figure 6.1. In DCTAM the MAVs handle both image acquisition and frame-to-frame tracking; having tracking and image capture running on the same device is key to the performance of DCTAM (for a fuller discussion see Section 6.8). The MAVs are all connected to the ground-station computer via a wireless link. The ground-station computer runs the Mapper which handles map creation and optimisation. Each component within the system is very modular and therefore is very flexible in terms of hardware and software. Additionally, while the target platform is aerial vehicles the architecture will support ground-based robots or hand-held devices (for example tablets or mobile phones).

6.3.1 The Map

A map $\mathcal{M}$ in DCTAM system is defined as:

$$\mathcal{M} = (\mathcal{P}, \mathcal{K})$$

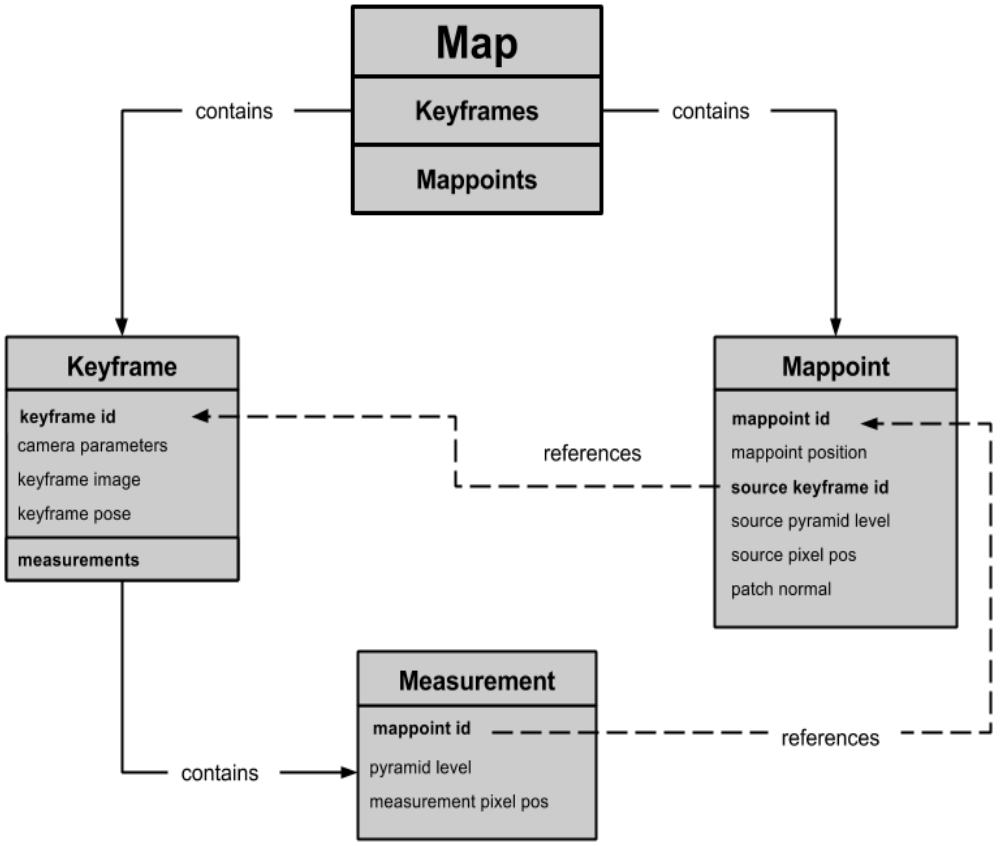


FIGURE 6.2: An overview of the DCTAM Map data structure. The Map consists of a set of keyframes and map-points, each keyframe contains a set of point measurements which reference a specific map-point. Each map-point contains a reference to the source keyframe from which the map-point was created.

where $\mathcal{P}$ is a sparse set of point features located in a global coordinate frame $\mathcal{W}$ and $\mathcal{K}$ is the set of keyframes. Each point feature is represented by an 8×8 pixel textured region in the world. The ith point in the map is stored as

$$\mathcal{P}_i = ({}^{\mathcal{W}}U_i, d_{\mathcal{P}i})$$

where ${}^{\mathcal{W}}U = ({}^{\mathcal{W}}x, {}^{\mathcal{W}}y, {}^{\mathcal{W}}z)$ is the point's 3D position and $d_{\mathcal{P}i}$ is the point descriptor. A notable aspect of Klien and Murray's approach is the fact that the pixels making up each point $\mathcal{P}_i$ are not stored. Instead the point descriptor, defined as $d_{\mathcal{P}} = (s, y, l)$, represents a pointer to the source keyframe s, the source pyramid level y and pixel location l. Therefore it is necessary to retain the original image pyramid $\mathcal{I}$ for all keyframes. The jth keyframe in the map is stored as:

$$\mathcal{K}_j = ({}^{\mathcal{W}}T_j, \mathcal{I}_j, \mathcal{N}_j, \mathcal{C}_j, d_{\mathcal{K}j})$$

where: (i) ${}^{\mathcal{W}}T = ({}^{\mathcal{W}}x, {}^{\mathcal{W}}y, {}^{\mathcal{W}}z, {}^{\mathcal{W}}\Phi, {}^{\mathcal{W}}\Theta, {}^{\mathcal{W}}\psi)$ is the 3D camera pose of the captured frame; (ii) $\mathcal{I}$ is a 4 level image pyramid and (iii) $\mathcal{N}$ is the set of intrinsic camera parameters (iv) $\mathcal{C}$ is a set of point measurements and (v) $d_{\mathcal{K}}$ is the keyframe descriptor. Each point measurement $c_j^i = (\mathcal{P}_i, \mathcal{K}_j, PL_j, PM_j)$ is a measurement of point $\mathcal{P}_i$ in keyframe $\mathcal{K}_j$ and PL_j is the image pyramid level and PM_j is the image point measurement. The keyframe descriptor is defined as $d_{\mathcal{K}} = SBI(I_k s)$, where SBI (Small Blurry Image) is the down-sampled and blurred re-localisation image (described in Section 5.2). An overview of the Map data structure is shown in Figure 6.2.

In order to achieve distributed tracking and mapping the Map data structure must be shared and updated by the trackers (on board each MAV) and the ground station. The interconnected nature of the map data structure makes this task more complex as references between map-points and keyframes must be maintained in order for the tracking and bundle adjustment operations to work correctly. A straight forward solution would be to serialise the entire map data structure and transmit these between the ground station and trackers. This has the advantage of preserving the complete data structure at the cost of a significant amount of bandwidth (as this would involve re-transmitting all the keyframe images). The goal with DCTAM is to minimise communication cost as much as possible in order to facilitate scalability. Therefore a different communication approach was chosen where the map data structure is not transmitted as a whole, instead separate messages are created for each map component (keyframes, map-points, measurements) so they can be transmitted independently. Additionally this approach allows the modification of the data structure to make communication more efficient. To each keyframe and map-points an integer identifier is added so each item can be referenced using its identifier. This facilitates the maintenance of the relationships between keyframes, measurements and map-points even when these messages are transmitted separately. This also helps reduce communication costs as a modification to the map can be done by reference. For example adjusting keyframe poses after a bundle adjustment can be accomplished by broadcasting the keyframe identifiers and new positions, instead of the entire keyframe data structure.

6.3.2 Tracking

DCTAM trackers use the same tracking approach to CCTAM, using AGAST corner features and the same two stage tracking process described in Section 5.4.2. In CCTAM a new keyframe is only passed on to the mapper if tracking quality is high (quality is based on the ratio of expected points to points actually found) and sufficient distance from any previous keyframes has been reached. In order to compensate for the potential latency between a new keyframe request and the response from the Mapper in DCTAM the tracker cannot request another new keyframe until a sufficient time t_{newkf} has elapsed (this is approach is also used by Adreas Wendel [119] for his single MAV distributed

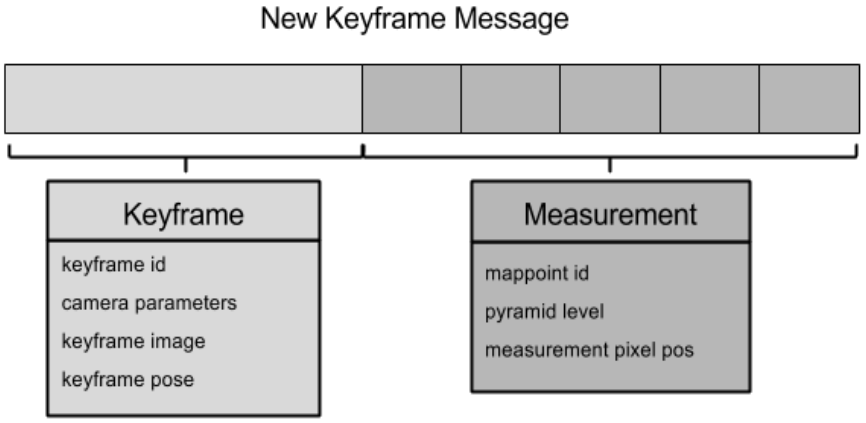


FIGURE 6.3: The structure of the New KeyFrame message in DCTAM. The message consists of a single keyframe as well as a set of measurement of existing map-points used when performing Bundle Adjustment.

system). The parameter t_{newkf} can be set manually or, in cases where network performance fluctuates, dynamically using the actual round trip time recorded from periodic ping request to the ground-station.

When the Tracker has travelled a pre-defined distance, or detected enough new features within the environment a new keyframe needs to be transmitted to the mapper. To achieve this the Tracker generates a new keyframe request message; the structure of this message is shown in Figure 6.3. The keyframe message contains a keyframe id however this is not set at this point, instead all identifiers are set by the mapper to ensure consistency. To reduce the size of the message the image is compressed using lossless Portable Network Graphics (PNG) compression. This keeps the bandwidth requirements low while ensuring the mapper does not receive an image with compression artefacts. This is important as the AGAST features extracted by the tracker are not transmitted with the new keyframe request. Instead we exploit the efficiency and repeatability of the AGAST corner detector to reduce communication costs by re-extracting the corner features when the request is received by the mapper. In addition to the new keyframe a set of measurements of existing map-point are included. These measurements are used to calculate the re-projection error for the Bundle Adjustment (BA) procedure. It was decided to include the camera parameters in each keyframe message, this is not strictly required as these parameters are fixed for each MAV, however it is useful for other high-level parts of the framework to have the image resolution and camera projection parameters available (see Chapter 6).

6.3.3 Stereo Initialisation

On system start up there are no existing points from which to triangulate, thus the map initialisation process requires a pair of keyframes with a sufficient baseline between them to triangulate the first set of map-points. The stereo initialisation message in

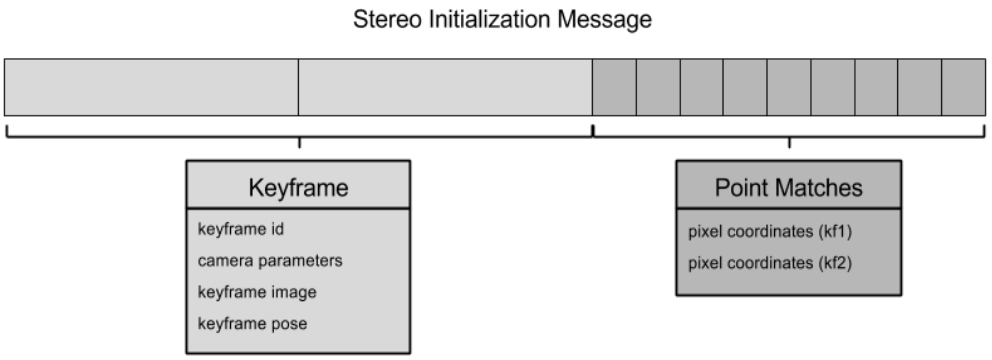


FIGURE 6.4: Structure of the Stereo Initialisation message used by the DCTAM system.

our system is shown in Figure 6.4. The message consists of the first two keyframes and a set of 2D point correspondences between the two keyframes (this is generated by the tracker during stereo initialisation). The mapper assumes the initialisation scene is planar and computes a homography to describe the relative transformation between the two keyframes. This relative transformation can then be used to compute the first set of map-points, by triangulation of the corresponding feature points in both images. After a successful stereo initialisation a new map is generated, this is broadcast to all trackers as a map update message which is described in more detail in the next section.

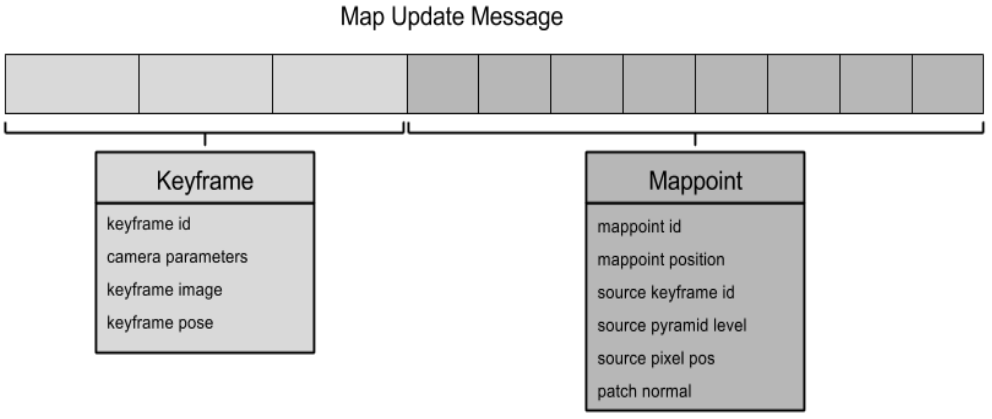


FIGURE 6.5: Structure of the Map Update message used by the DCTAM system.

6.3.4 Mapping

The Mapping process runs on the ground-station and is responsible for building the initial map using a stereo initialisation process and then further extending the map using the keyframes provided by the trackers. When the mapper receives a new keyframe request from a tracker it reconstructs the full keyframe structure including the image pyramid and AGAST features. This is a repetition of some steps performed by the

tracker, but this is done to reduce the bandwidth requirements. New map-points are added by triangulating points in neighbouring keyframes and the poses refined by local bundle adjustment as in CCTAM. As in the original PTAM we use the local bundle adjustment procedure which operates on the new keyframe and its four closest (linear distance) neighbours. As this local bundle adjustment converges quickly we only generate a new map update once it has finished.

The structure of the map update is shown in Figure 6.5. The map update contains the set of all newly created map-points as well as the source keyframes. It is at this point that each keyframe and map-point is assigned a new, unique identifier. As tracking is performed using direct matching of image patches it is necessary for all trackers to retain the original keyframe images. Therefore it is necessary for the mapper to re-broadcast the source keyframes. We again use PNG compression and transmit only the original image to keep bandwidth costs as low as possible. If only a single MAV is being used this is not necessary as the tracker will already have the keyframe image.

In addition to a local bundle adjustment a global bundle adjustment is also performed. The bundle adjustment procedure aims to reduce the re-projection error; the expected position versus measured position for all map-points in keyframes using the set of measurements. It has the effect of generating a new position for the affected map-points and a new pose for each affected keyframe.

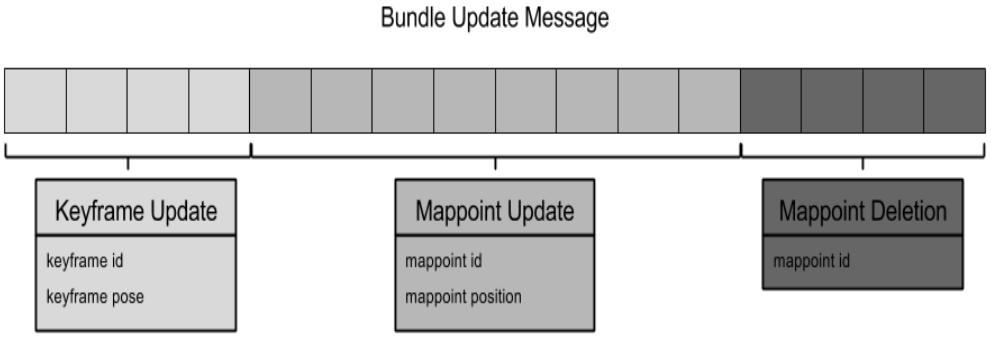


FIGURE 6.6: Structure of the Bundle Adustment Update message used by the DCTAM system.

Instead of generating a new map update, once bundle adjustment converges, we generate a bundle update. The structure of the bundle update message is shown in Figure 6.6. As each keyframe and map-point has a unique identifier we can keep the message size to a minimum and include only the identifier and new pose/position for each keyframe/map-point. The bundle update also contains a list of map-points to be deleted, these are points with few measurements which are identified to be outliers during bundle adjustment.

6.4 Message Handlers

Message handlers are added to both the Tracker and Mapper components which are responsible for processing incoming messages. For the Tracker two message handlers HandleMapUpdate and HandleBundleUpate are defined. The algorithm for HandleMapUpdate is described in Algorithm 10 and HandleBundleUpate is described in Algorithm 11. Where DECODE is a function which takes as input a PNG compressed image and will return the raw uncompressed image. The functions KFLOOKUP and MPLOOKUP take a keyframe or map-point identifier as input and will return the corresponding keyframe or map-point.

Algorithm 10 HandleMapUpdate(MU, M)

Input: a Map Update message consisting of a set of keyframe messages (kfm) and map-point messages (mpm) and the local map to update M.

for all $kfm \in MU$ **do**
  $\mathcal{I} \leftarrow$ DECODE(keyframe image)
  ${}^{\mathcal{W}}T \leftarrow$ keyframe pose
  $\mathcal{N} \leftarrow$ camera parameters
  $\mathcal{C} \leftarrow \emptyset$
  $d_{\mathcal{K}} \leftarrow$ SBI($\mathcal{I}$)
  $k \leftarrow ({}^{\mathcal{W}}T, \mathcal{I}, \mathcal{N}, \mathcal{C}, d_{\mathcal{K}})$
  $M \cup \{k\}$
end for
for all $mpm \in MU$ **do**
  ${}^{\mathcal{W}}U \leftarrow$ mappoint position
  $s \leftarrow$ KFLOOKUP(source keyframe id)
  $y \leftarrow$ source pyramid level
  $l \leftarrow$ pixel coordinates
  $d_{\mathcal{P}} = (s, y, l)$
  $m \leftarrow ({}^{\mathcal{W}}U, d_{\mathcal{P}i})$
  $M \cup \{m\}$
end for

Algorithm 11 HANDLEBUNDLEUPATE(BU)

Input: Bundle Update message consisting of a set of keyframe update messages (kfu) and map-point update messages (mpu).

1: **for all** $kfu \in BU$ **do**
2: $\quad {}^{\mathcal{W}}T \leftarrow$ keyframe pose
3: $\quad k \leftarrow$ KFLOOKUP(keyframe id)
4: $\quad k \leftarrow^{\mathcal{W}} T$
5: **end for**
6: **for all** $mpu \in BU$ **do**
7: $\quad {}^{\mathcal{W}}U \leftarrow$ keyframe pose
8: $\quad k \leftarrow$ MPLOOKUP(mapppoint id)
9: $\quad k \leftarrow^{\mathcal{W}} T$
10: **end for**

For the Mapper another two message handlers HANDLESTEREOINIT and HANDLENEWKEYFRAME are defined. The algorithm for HANDLESTEREOINIT is described in Algorithm 12 and HANDLENEWKEYFRAME is described in Algorithm 13. Where ${}^{\mathcal{W}}T \leftarrow \mathbf{0}$ initialises the transformation ${}^{\mathcal{W}}T$ with all zero values, COMPUTEHOM computes a homography from point matches in two images, BUNDLEADJUST_G computes a global bundle adjustment for a given map and BROADCAST_MU will create and broadcast a Map Update message for a given map.

Algorithm 12 HANDLESTEREOINIT(SI)

Input: A Stereo Initialisation message SI consisting of a pair of keyframe messages (kfm) and a set of point matches (pm).

Ouput: A new map M

$M \leftarrow \emptyset$ ▷ Create an new empty map
for all $kfm \in SI$ **do** ▷ Process keyframe messages
  $\mathcal{I} \leftarrow$ DECODE(keyframe image)
  ${}^{\mathcal{W}}T \leftarrow \mathbf{0}$
  $\mathcal{N} \leftarrow$ camera parameters
  $d_{\mathcal{K}} \leftarrow$ SBI($\mathcal{I}$)
  $\mathcal{C} \leftarrow \emptyset$
  $k_i \leftarrow ({}^{\mathcal{W}}T, \mathcal{I}, \mathcal{N}, \mathcal{C}, d_{\mathcal{K}})$
end for
$HT \leftarrow$ COMPUTEHOM$(k_1, k_2, pm_0, \ldots, pm_n)$
${}^{\mathcal{W}}T \leftarrow HT$ for ${}^{\mathcal{W}}T \in k_2$)
for all $pm \in SI$ **do** ▷ Create mappoints from point matches
  $mp \leftarrow$ POINTTRIANG$(kf1, kf2, pm)$
  $mp \cup \{m\}$
end for
BUNDLEADJUST_G(M)
BROADCAST_MU(M)
return M

Algorithm 13 HANDLENEWKEYFRAME(NK, Q_{kf})

Input: A New Keyframe message NK including a set of measurements mes and the new keyframe processing queue Q_{kf}.

$\mathcal{I} \leftarrow$ DECODE(keyframe image)
${}^{\mathcal{W}}T \leftarrow$ keyframe pose
$\mathcal{N} \leftarrow$ camera parameters
$d_{\mathcal{K}} \leftarrow$ SBI($\mathcal{I}$)
$\mathcal{C} \leftarrow \emptyset$
for all $mes \in NK$ **do**
  $\mathcal{P} \leftarrow$ MPLOOKUP(mapppoint id)
  $\mathcal{K} \leftarrow$ keyframe id
  $PL \leftarrow$ pyramid level
  $CM \leftarrow$ pixel measurement
  $c \leftarrow (\mathcal{P}, \mathcal{K}, PL, CM)$
  $\mathcal{C} \cup \{c\}$
end for
$k \leftarrow ({}^{\mathcal{W}}T, \mathcal{I}, \mathcal{N}, \mathcal{C}, d_{\mathcal{K}})$
$Q_{kf} \cup k$

The new keyframe processing queue Q_{kf} is a First-In-First-Out (FIFO) queue to which each new keyframe is added by the new keyframe handler. Each new keyframe is processed and added to the map, the algorithm for processing new keyframes is given in Algorithm 14. Where AGAST returns all AGAST features extracted from a given keyframe image, FINDCLOSEST returns the closest, in terms of linear distance, keyframe to a given keyframe, EPISEARCH uses epipolar search to find matching points in two images and BUNDLEADJUST_L performs bundle adjustment on a given keyframe and its four closest (linear distance) neighbours.

Algorithm 14 PROCESSNEWKEYFRAMES(Q_{kf}, M)

Input: The new keyframe processing queue Q_{kf} and the map M.

for all $kf \in Q_{kf}$ **do**
  $fp \leftarrow \text{AGAST}(kf)$
  $M \cup kf$
  $kf_c \leftarrow \text{FINDCLOSEST}(kf, M)$
  $fp_c \leftarrow fp \in kf_c$
  for all $l \in PL$ **do**
    $pm \leftarrow \text{EPISEARCH}(kf, fp, kf_c, fp_c)$
    $mp \leftarrow \text{POINTTRIANG}(kf, kf_c, pm)$
    $mp \cup \{m\}$
  end for
  BUNDLEADJUST_L(kf, M)
  BROADCAST_MU(M)
end for

6.4.1 Automated Stereo Initialisation Methods

There are many alternatives to generating the stereo pair required for map initialisation such as using fiducial markers [120], multiple model filtering [95] or use of a stored map. CCTAM relied on sensor information from the AR.Drone, particularly optical flow and sonar. This worked well but required all platforms to have the same suite of sensors. A more generic approach will increase the number of potential platforms supported by the framework; therefore, DCTAM supports multiple automated initialisation methods.

The first initialisation approach tracks the camera baseline in pixel space and will trigger stereo initialisation once a user-defined pixel baseline has been reached. This is the most general approach as the only data required is image data. However as this approach does not have a metric estimate of the scene depth the map will be initialised with an arbitrary scale. This is sufficient in applications where metric scaling is not required or the visual scale is estimated other means (e.g. using an EKF). The second approach relies on metric depth data provided by a sensor such as sonar or laser and applies metric scaling to the optical flow using the method described in Section 3.2.3. This allows the camera baseline to be tracked in metric space; meaning the map can

also be initialised in metric space. The final approach also requires no additional sensors by making use of a fiducial marker based initialisation method similar to [120]. The know dimensions of the marker can be used to track the camera baseline in metric space ensuring the map is also initialised in metric space. These initialisation methods provide a more general solution to stereo initialisation allowing the framework to be used on a larger variety of MAV platforms.

6.5 Additional Components

Similar to the discussion presented earlier regarding CCTAM, a discussion of the additional components required to make DCTAM a full visual navigation solution is presented in this section. These components include a more general Extended Kalman Filter (EKF) framework as well as the collision avoiding trajectory controller (introduced in Chapter 4).

6.5.1 Extended Kalman Filter (EKF)

In order to make the framework more generic it was decided not to re-use the EKF from CCTAM discussed in Section 5.5. Instead the more generic Multi-Sensor-Fusion Extended Kalman Filter (MSF-EKF) approach of Lynen et al. [71] was employed. MSF-EKF is a modular, generic EKF framework which supports a wide variety of absolute sensors (for example vision, Global Positioning System (GPS)) as well as relative sensors (for example Optical Flow, wheel odometry). The only sensor explicitly assumed by the framework is an Inertial Measurement Unit (IMU). The IMU is used as a fixed prediction sensor; the IMU measurements are integrated as a pseudo control signal in the EKF prediction step (see Section 2.5). Additional sensors, such as Visual SLAM, Visual Odometry or GPS, can then be integrated in the update step of the filter. As this filter already includes the necessary sensor models no modification was necessary in order to use this filter framework with DCTAM.

Note that while this configuration is more generic it does have some disadvantages, as the EKF relies purely on inertial data when no vision estimates from DCTAM are available. The double integration of accelerometer data to estimate position is acceptable for only short periods and in the absence of vision estimates the EKF will diverge very quickly from the true estimate. This makes the system more sensitive to tracking failures; however if, as in the case of the AR.Drone, the MAV platform includes a optical flow sensor then this data can easily be integrated into the EKF to improve robustness.

6.5.2 Trajectory Control

For trajectory control we employ the collision avoidance trajectory controller described in Chapter 4. For all hardware experiments the trajectory controller runs on-board the MAV together with the DCTAM Tracker and EKF. The same MAV radius scaling approach described in Section 5.6 was also used for DCTAM.

6.6 Implementation

The DCTAM framework was implemented in C++ and integrated into the Robot Operating System (ROS) [93]. To further reduce bandwidth requirements the multi-master package multimaster_fkie[1] is used to ensure only the DCTAM messages are transmitted over the wireless link between ground-station and MAVs.

The bundle adjustment procedure used in the previous work on CCTAM had some limitations in terms of it's performance. With very large maps ≈ 20000 map-points the global bundle adjustment can often take tens of seconds to converge. Additionally, as bundle adjustment must be aborted in order to add new keyframes to the map, this means the frequency with which the global bundle adjustment successfully converges reduces drastically as the map size grows. This can lead to problems with drift, to partially address this issue bundle adjustment was implemented using the generalised graph optimisation framework G2O [58]. This framework provides a framework for efficiently solving least squares optimisation problems like bundle adjustment. The main advantage of this approach is its generic structure, which means that implementing other optimisation techniques, such as loop closures, can be done more easily. It has the added benefit of being moderately more efficient (as demonstrated by the experiments reported on in the next section) which results in a increased completion rate for the global bundle adjustment.

FIGURE 6.7: The first DCTAM hardware platform based on the AR.Drone frame and PX4 flight controller.

[1] `http://wiki.ros.org/multimaster_fkie`

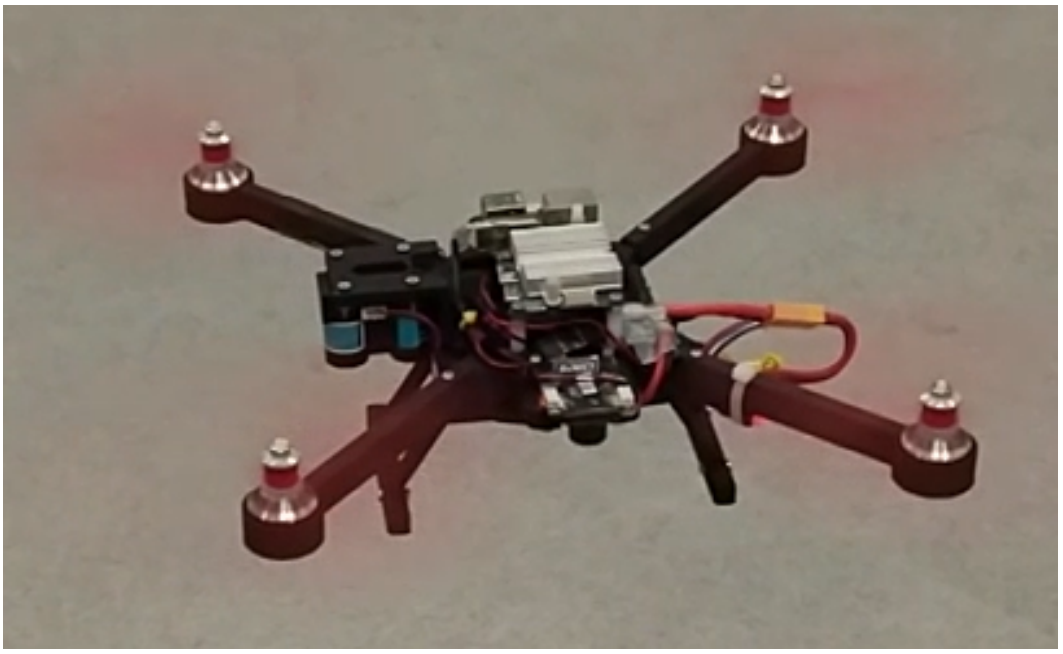


FIGURE 6.8: The second DCTAM hardware platform based on a custom designed 3D printed frame.

6.7 DCTAM Hardware Platforms

Two hardware platforms were developed to both serve as a reference platform for the DCTAM system as well as providing platforms for the experimental evaluation. For these platforms a dual processor architecture was chosen consisting of a flight controller and an on-board computer. The architecture of this approach is illustrated in Figure 6.9. The flight controller serves as a sensor and motor interface as well as an attitude controller. The on-board computer handles the higher level tasks including running DCTAM, the EKF and position controller, as well as interfacing with the camera sensor. This dual processor architecture was chosen for its ease of implementation using existing flight controllers and on-board computers. This approach has the added benefit in terms of robustness, as a complete failure of the on-board computer does not result in complete failure of the aircraft and the flight controller is still available to stabilise and land the aircraft.

Two separate hardware platforms were developed for use with DCTAM. While comprised of different hardware, each had the required components to run, DCTAM (flight controller, on-board computer with WiFi link and a single camera). The first reference platform is based on the popular AR.Drone frame (see Figure 6.7) and is based on the platform developed by Faessler et al [25]. The AR.Drone electronics were replaced with a PX4-based flight-control system consisting of a PX4 Flight Management Unit (PX4FMU) and an AR.Drone adapter board (PX4IOAR) which interfaces with the AR.Drone motors. The on-board companion computer is an Odroid U3, an ARM-based single board computer with a 1.7 Ghz Quad-core processor with 2 GB of RAM. The MAV has a single MatrixVision mvBlueFOX-MLC200w 752x480 pixel monochrome

camera fitted with a 100° wide-angle lens. The total cost of this platform is £750 and the average flight time is 10 minutes. The second platform is a custom 3D-printed quad-copter frame (see Figure 6.8), the flight controllers is another PX4-based flight controller and the on-board computer is the Odroid U3. The total cost of this platform is £400 and the average flight time is 12 minutes. Full details of the design of the 3D printed platform as well as links to repositories containing all hardware designs and software for the DCTAM framework are described in Appendix A.

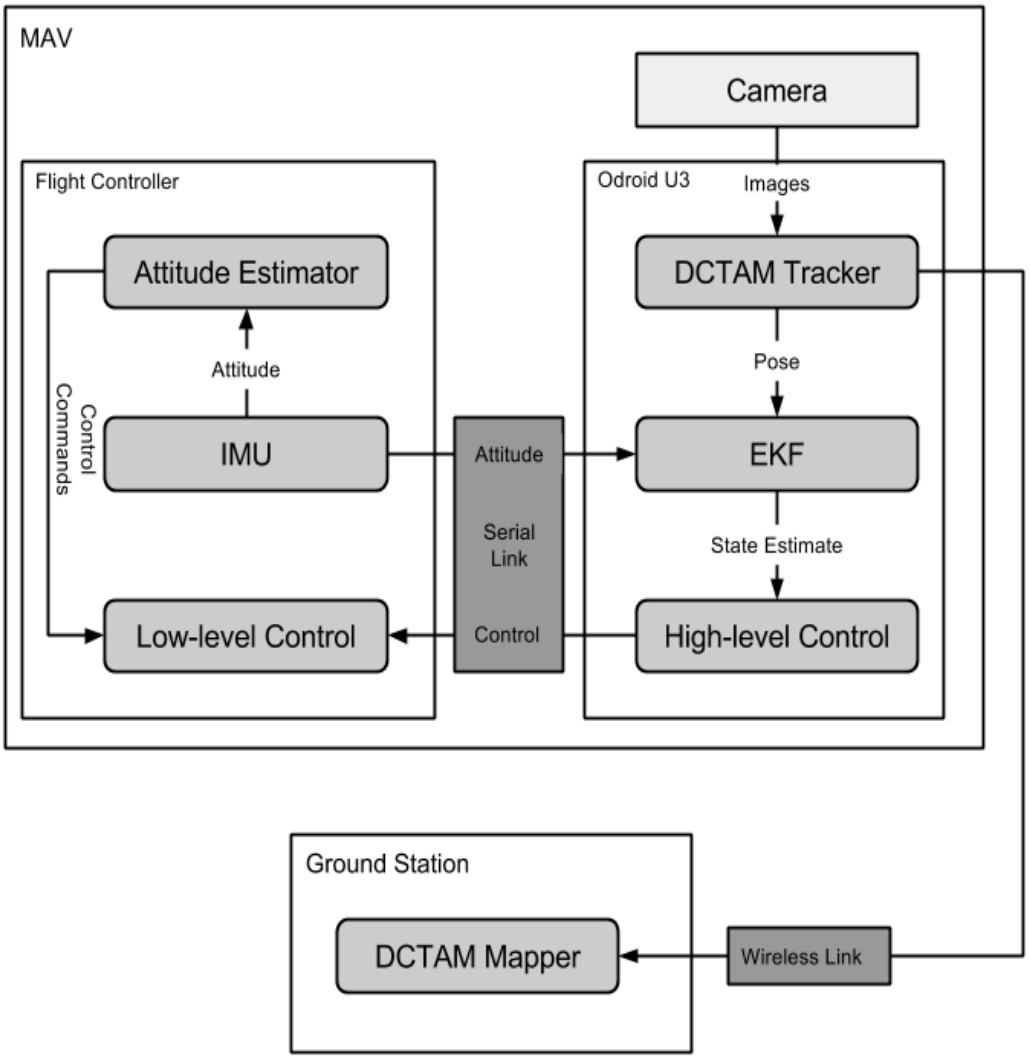


FIGURE 6.9: The dual processor architecture of the DCTAM hardware platforms.

The memory requirements of the DCTAM system are not significant, the largest component required to store in memory is the Map, of which the keyframe images make up the bulk of the data. For the largest maps used in the work in this thesis the number of keyframes is in the order of 150 keyframes. This means a memory requirement of $\approx$ 46 Megabytes and given typical memory capacity for single board computers of this

class is in the gigabyte range it can be concluded that memory requirements are not a limiting factor for the DCTAM system.

Processor performance however is a limiting factor, tests were conducted with several common on-board computers and the results are presented in Section 6.8.10. Of the on-board computers tested the one with the worst performance was the Intel Atom-based computer. While the computer is still able to run the DCTAM system it is only capable of an update rate of 14 Hz. As discussed previously in our evaluation of CCTAM's scalability (Section 5.8.5) an update rate of 20 Hz or more is best for robust performance with fast moving MAVs.

6.8 Evaluation

This section describes the experiments conducted on the Distributed Collaborative Tracking and Mapping (DCTAM) framework and is organised as follows: it begins with a description of the experimental setting including the simulation and hardware platforms. Section 6.8.3 describes the robustness experiments performed on the DCTAM framework, Section 6.8.4 goes on to discuss the scalability experiments performed on the framework. And Sections 6.8.6 and 6.8.7 present the results obtained using DCTAM for two multi-agent coordination tasks, collision avoidance and exploration. The section concludes with an analysis of the performance of DCTAM on a number of on-board computer in Section 6.8.10.

6.8.1 Simulation Environment

The same simulator (Gazebo) and simulated environments described in Section 5.8.1 are used for the experiments described in this section. During the course of this work a limitation with the Gazebo simulator was identified; increasing the number of MAVs being simulated affects the frame rate of the simulated cameras. Note this is not the tracking frame rate as discussed in Section 5.8.5 but the ability of the simulator itself to generate images for the simulated cameras at a reliable frame rate. Due to this limitation for the larger experiments described in this section a distributed simulation architecture is employed. Good performance is maintained when a maximum of 4 MAVs per simulator are used. Therefore a separate simulator instance is created (with the same simulated world) for each 4 MAVs in our experiments. All MAVs share the same map and communicate their positions (for collision avoidance) as before. This means as far as the localisation, control and decision making components of the system are concerned the MAVs are all operating within the same environment. This approach facilitated the larger scale experiments described in this section.

6.8.2 DCTAM Hardware Platforms

The hardware platforms used in the hardware experiments described in this section were the DCTAM reference platforms introduced in Section 6.7. These experiments

were conducted in the same lab environment as those described in Section 5.8 under the same conditions in terms of both lighting and wireless network conditions.

6.8.3 Robustness

A common artefact of wireless networks is high latency which can have a negative impact on a distributed system like ours. To investigate this simulated experiments were conducted where artificial random delays were introduced to determine the effect of delay on our system. Figure 6.10 shows the results of two delay experiments conducted, here a single MAV is flown at a constant speed in a single direction. DCTAM is compared to the previous approach, CCTAM, where the video feed is streamed from the MAV to the ground-station. As the MAV is flying into a previously unmapped area both systems must capture new keyframes and add new points to the map.

As the results show DCTAM system performs well even with a maximum delay of 1000 milliseconds. Delay of this kind can be compensated for using an EKF and motion model such as those used in CCTAM. However as mapping is vision based and new keyframes can only be reliably included in the map when tracking is stable such a solution is not effective in the presence of sufficiently large delays. This is demonstrated in the second experiment where the delay is large enough for tracking and mapping to fail completely. The experiment was extended to test the limits of our system and it was found that even with an extreme maximum delay of 30 seconds DCTAM can continue to operate. This is only possible if the MAV is moving sufficiently slowly that newly added map-points reach the tracker before it needs them to maintain stable tracking. It is fairly trivial to keep track of updates from the mapper and adjust the behaviour of the MAV's controller based on the current state of the system. If a tracker is waiting for new map-points from the mapper the MAV can continuously reduce its speed in order to ensure it does not leave the mapped area before the new map-points arrive.

Another important consideration in terms of robustness is the maximum speed the system allows the MAVs to travel. A number of tests were conducted and they determined that for a camera with a standard 60 FPS update rate we are able to achieve speeds up to 7 m/s before tracking performance becomes unreliable. Of interest is the fact that past 7 m/s it is not the delay in processing introduced by the distributed architecture of DCTAM that impacts performance. Instead the main factor is the pitch/roll rate of the MAV as it transitions from hover to forward flight. Reducing the maximum pitch and roll rates leads to more stable visual tracking performance at the expense of the reducing the acceleration and controllability of the craft.

6.8.4 Scalability

One of the most important factors affecting the scalability of a distributed system is the communication cost. If communication is expensive the benefits of running a distributed system are reduced or negated completely. Figure 6.11 shows the bandwidth requirements of a single MAV exploring in the simulated world; the final map for this

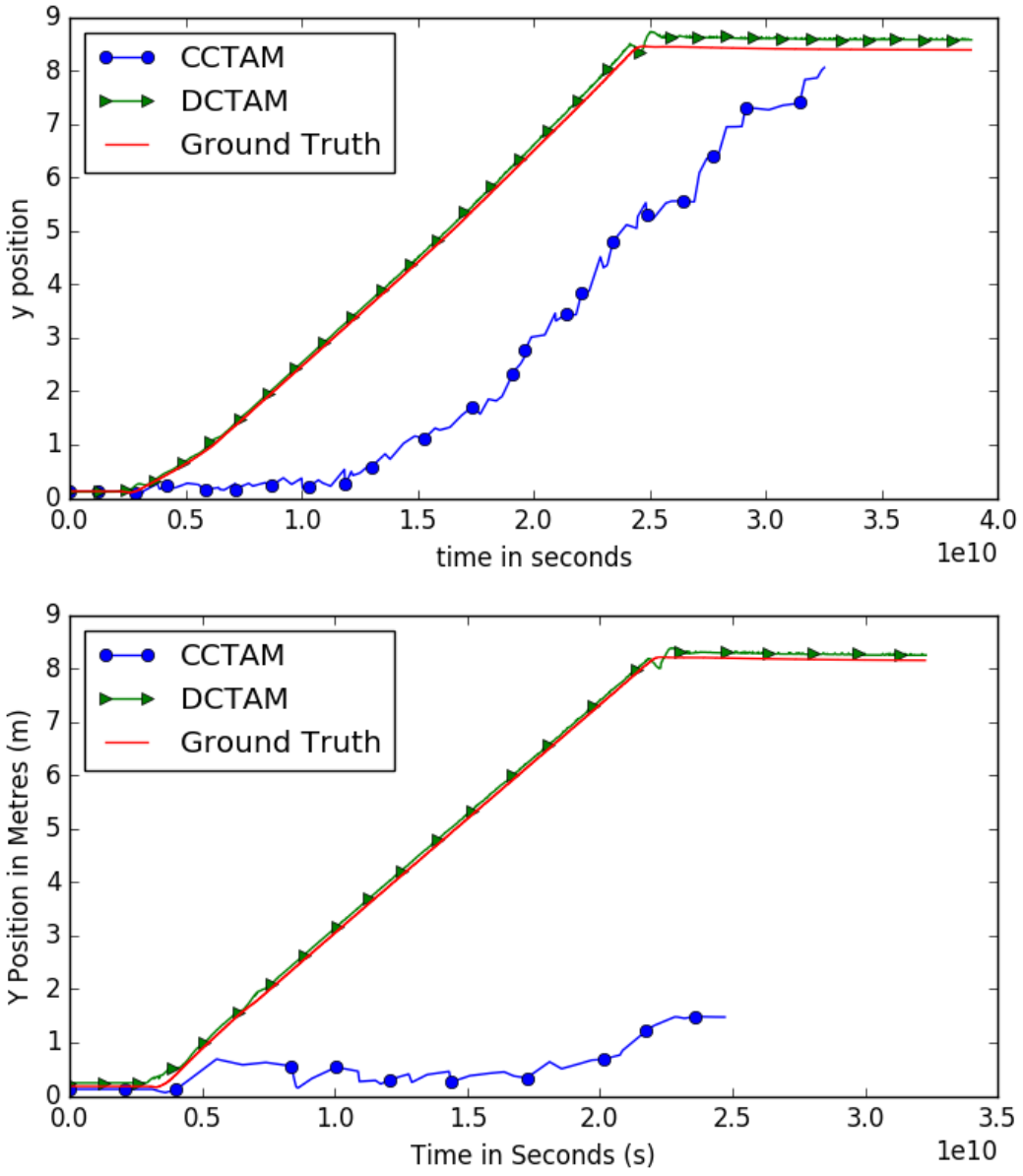


FIGURE 6.10: Results of the delay experiment showing a delay of 600 ms (top) and 1000 ms (bottom).

experiment was 52 keyframes and 7240 map-points. Even with a very large map the required bandwidth remains very low for the DCTAM system. Significantly lower than streaming video directly (56 MB/s) to the ground-station as we did for CCTAM (see Chapter 4) and even lower than the 1 MB/s required by [95] who use the same library as CCTAM and DCTAM but who send the full colour image captured by the camera. We instead send only a compressed grey-scale image (we use lossless PNG compression with a low compression rate to limit computation time) and are able to achieve a requirement of only 9 Kb/s for a single MAV and 42 Kb/s for a single MAV operating as part of a team. The additional bandwidth is required for a tracker operating as part of a team as this requires broadcasting the new keyframes to all MAVs in the team.

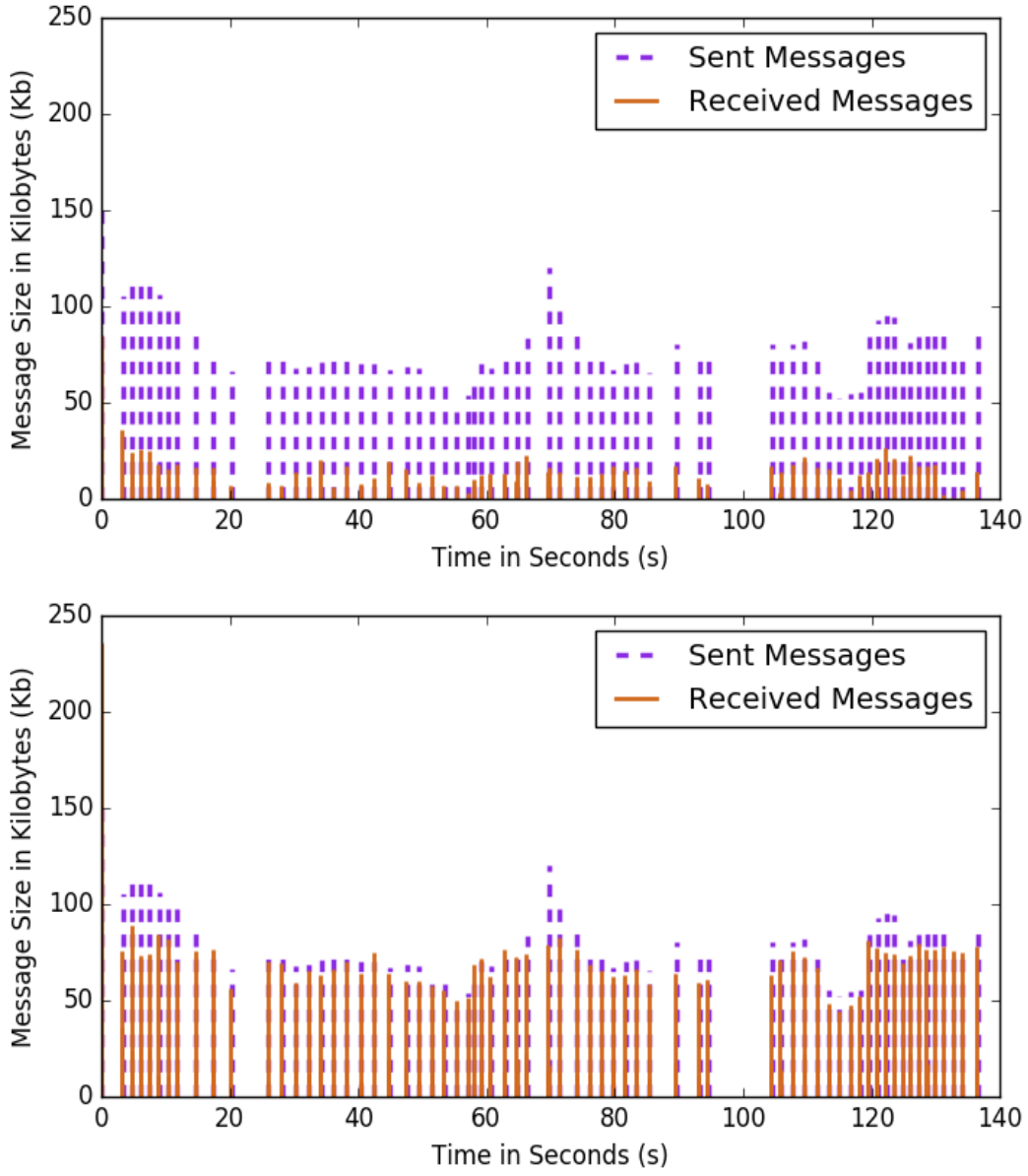


FIGURE 6.11: Results of the bandwidth experiment showing the bandwidth requirements for a single drone when operating alone (top) and as part of a team (bottom). Note how the requirement to transmit the keyframes to all MAVs increases the received messages whereas the sent messages remain the same.

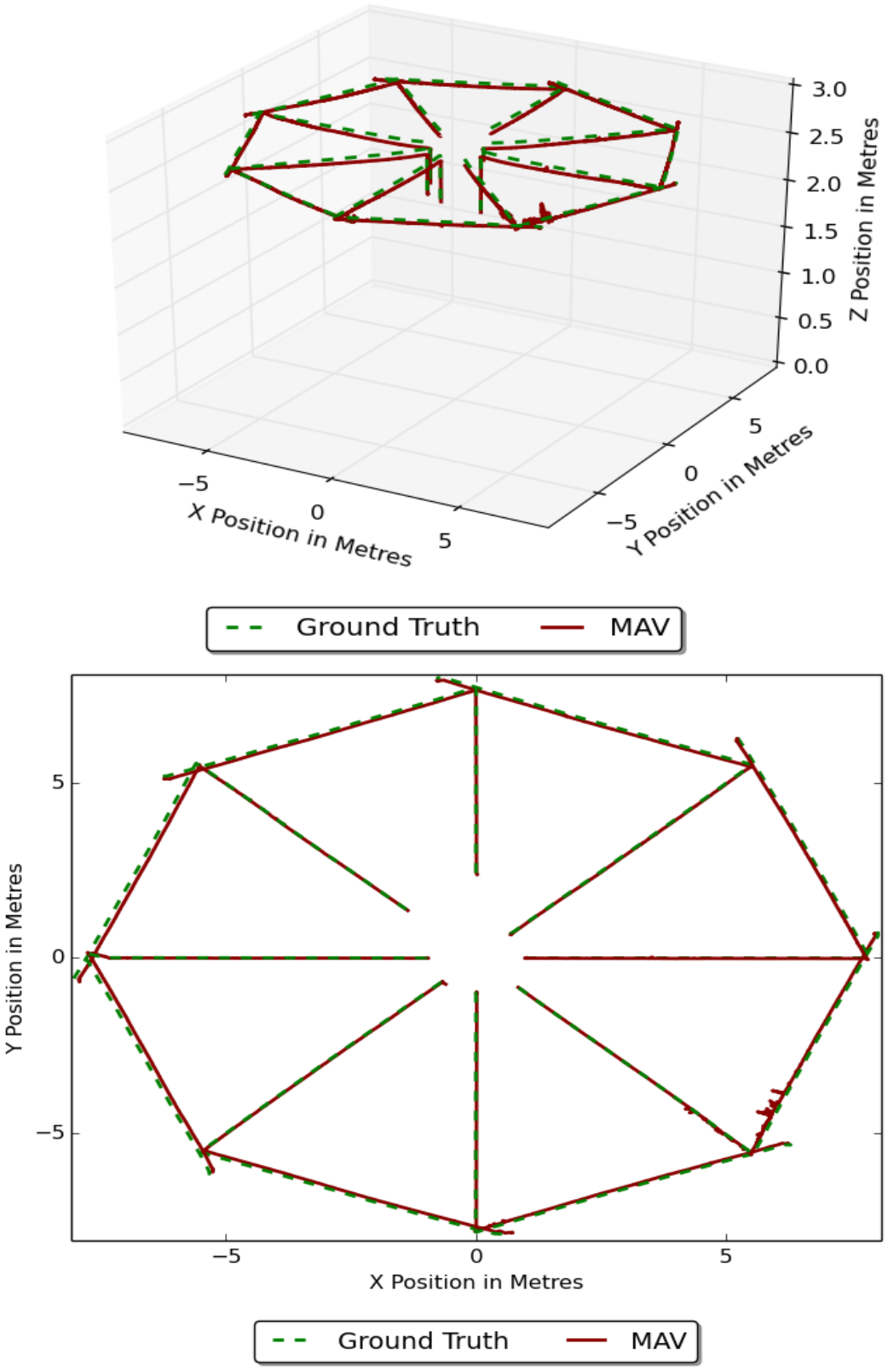


FIGURE 6.12: Results of the scalability experiments, these graphs show the trjacectories in 2D (top) and 3D (bottom) of 8 MAVs simultaneously exploring the same environment.

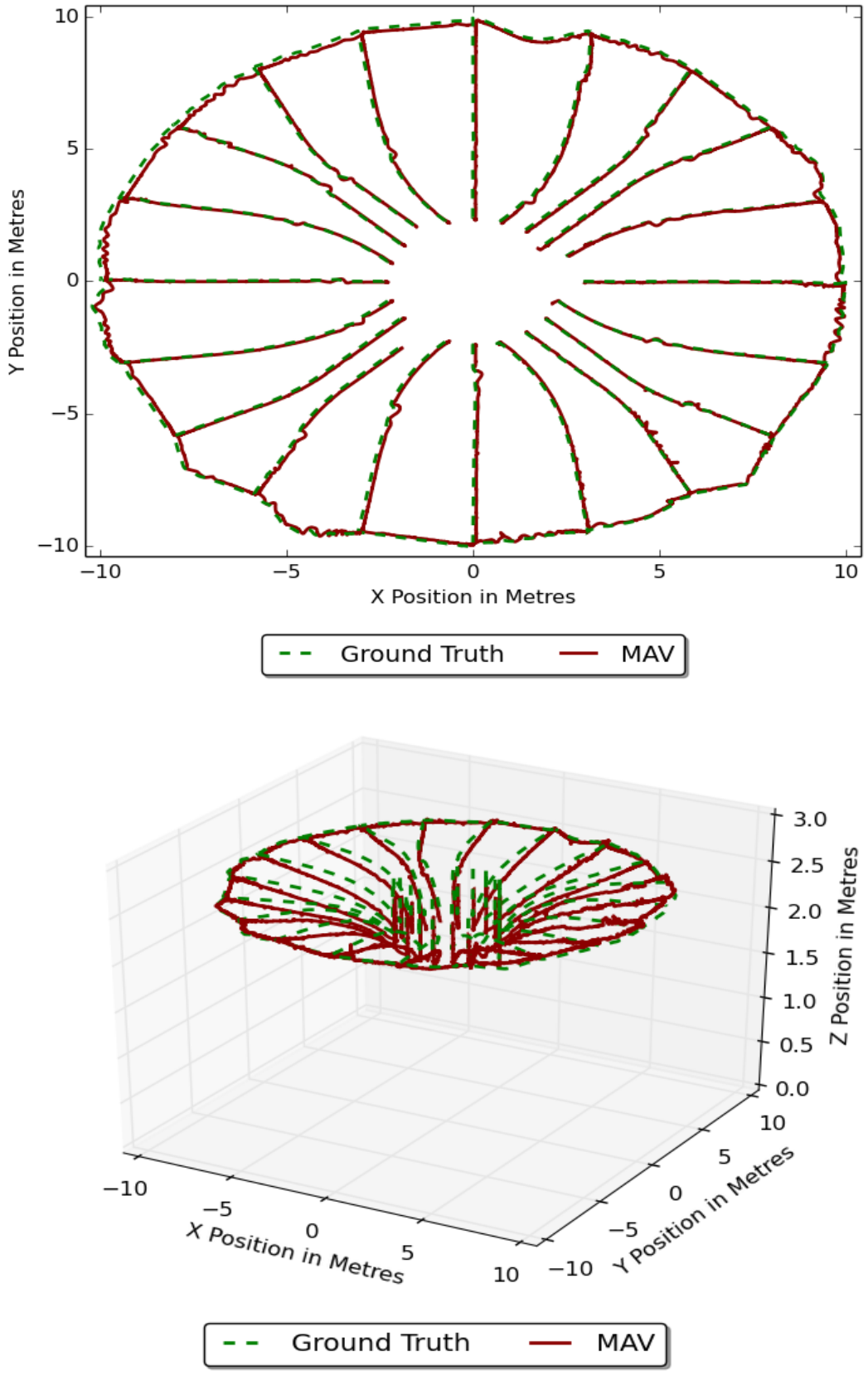


FIGURE 6.13: Results of the scalability experiments, these graphs show the trjacectories in 2D (top) and 3D (bottom) of 20 MAVs simultaneously exploring the same environment.

With the reductions in bandwidth and processing requirements (from the distributed architecture) comes the ability to handle larger teams of MAVs. Assessing the bandwidth requirements for a DCTAM is somewhat challenging as communication only occurs when a new keyframe is added to the map. In order to achieve reproducible results the experiment is structured as follows. Firstly each MAV starts in the middle of our simulated world, they are commanded to fly in a straight path outward until reaching the edge of the environment. During this outward flight the MAVs will all be simultaneously mapping new areas and therefore communicating with the ground-station regularly. Once each MAV reaches the edge of the area it is commanded to fly across to the position occupied by the adjacent MAV completing a polygon pattern (see Figures 6.12 and 6.13). In this phase each MAV flies into an area mapped by another MAV and any inconsistency in the MAVs map caused by communication errors would lead to tracking failures. Figure 6.12 shows the trajectories for the experiment conducted with a team of 8 MAVs. This experiment was run with a maximum random delay of 1000 milliseconds to simulate a poor wireless network (typical examples of delay in commercial or home wireless networks are from 250-500 milliseconds). The total distance travelled was 139.4 metres with a final RMS error of 0.09 metres.

The distributed simulation architecture helps maintain a good frame rate while scaling up to larger experiments. Figure 6.13 shows the results from the largest scalability experiment conducted featuring a team of 20 MAVs flying pre defined paths as before mapping as they go. This experiment was run with a maximum random delay of 1000 and the mean RMSE was 0.09 metres. Figure 6.14 shows the bandwidth requirements for all the MAVs. The bandwidth required for all sent messages (MAV to ground station) was 561.45 Kb/s and the bandwidth for all received messages (ground station to all MAVs) was 243.67 Kb/s. As can be seen from the results reliable localisation can be maintained with a large team with very low bandwidth requirements making it possible to deploy the system with a large team using no specialised communication equipment.

6.8.5 Hardware Scalability Experiments

It may be argued that, in the experiments shown in Figures 6.12 and 6.13, artificial random delay does not approximate real world network conditions. Additionally lacking the physical resources (both space and MAVs) to repeat the experiment in the real world the next best approach was used. Both experiments were repeated under more realistic conditions using a set of netbook computers which served as proxy-MAVs, each playing back recorded images from the previous simulated experiment to on-board trackers. Each tracker was connected via 802.11n wireless network to the mapper running on the same ground-station computer. The results of these repeated experiments are shown in Figures 6.15 and 6.16 and the final RMS error for these experiments both improved to 0.06 metres for the 8 MAV experiment and 0.067 metres for the 20 MAV experiment. This improvements is down the running each tracker on a completely separate computer,

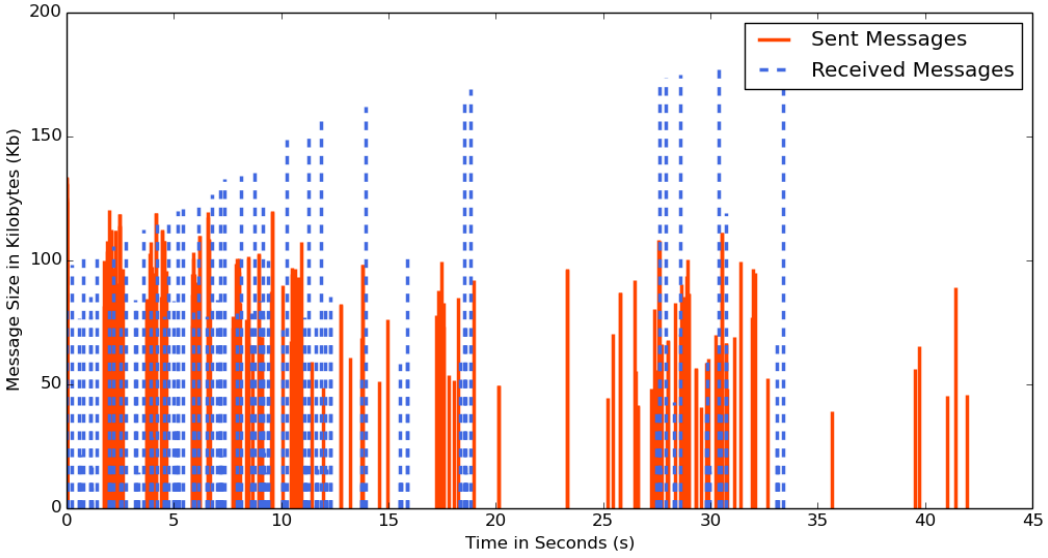


FIGURE 6.14: This graph plots the bandwidth requirements for a team of 20 MAVs.

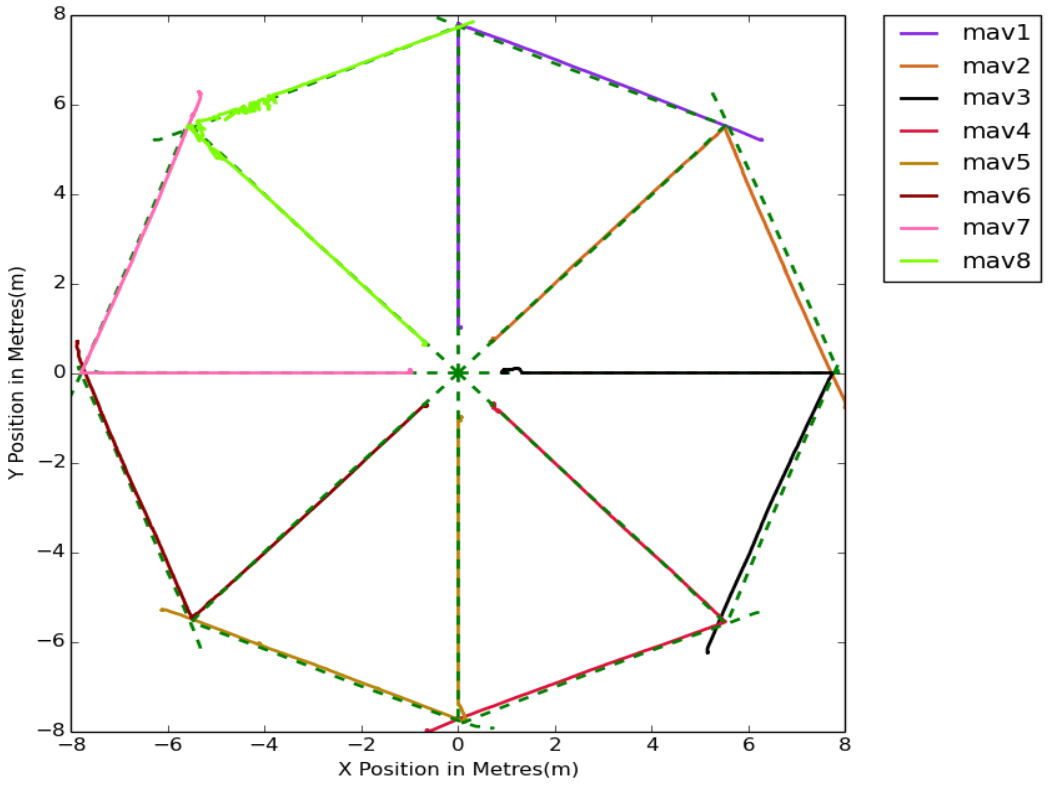


FIGURE 6.15: Results of the repeated experiment featuring 8 MAVs, with the Trackers running on real hardware, the final RMS error was 0.06 metres.

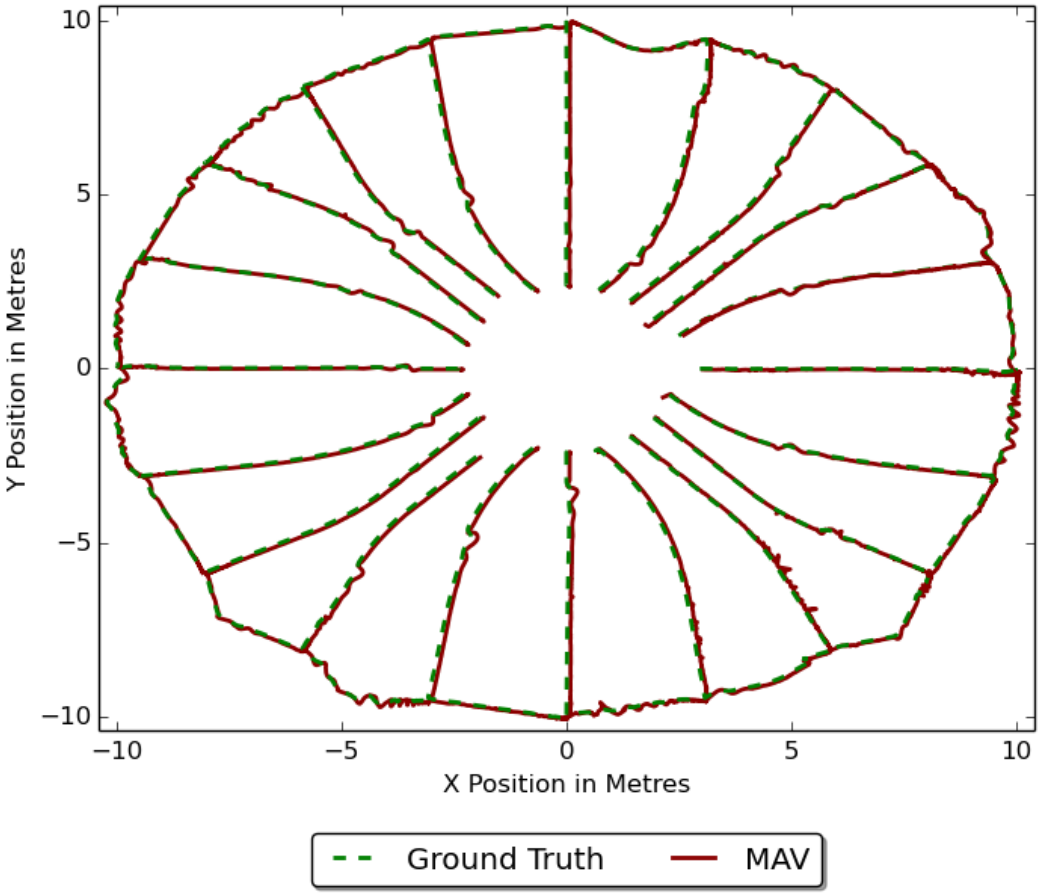


FIGURE 6.16: Results of the repeated experiment featuring 20 MAVs, with the Trackers running on real hardware, the final RMS error was 0.08.

as opposed to all trackers running on the same computer as is done for the simulated experiments.

6.8.6 Multi-Robot Task 1: Collision Avoidance

The collision avoidance experiment conducted with CCTAM (Section 5.8.6) produced interesting results with respect to the reliability of the framework. As DCTAM featured improved scalability (enabling experiments with larger teams of MAV) and robustness (allowing the MAVs to run at full speed with reliable tracking performance) the collision avoidance experiment was repeated to determine if these improvement yield results in a practical scenario. In the previous experiment as the size of the team increased (causing the tracking update rate to reduce) an increase in the rate of collisions occurring due to tracking failures was seen. However as the results show in table 6.1 using the DCTAM pose estimate all MAVs are able to avoid 100% of collisions as the team size scales. An example of the results from these experiments is shown in Figure 6.17. It is interesting to note that a comparison between the trajectories of the CCTAM collision avoidance experiments (Figure 5.14) and the DCTAM experiments (Figure 6.18) that the trajectories of the DCTAM experiments appear smoother. With the distributed architecture the

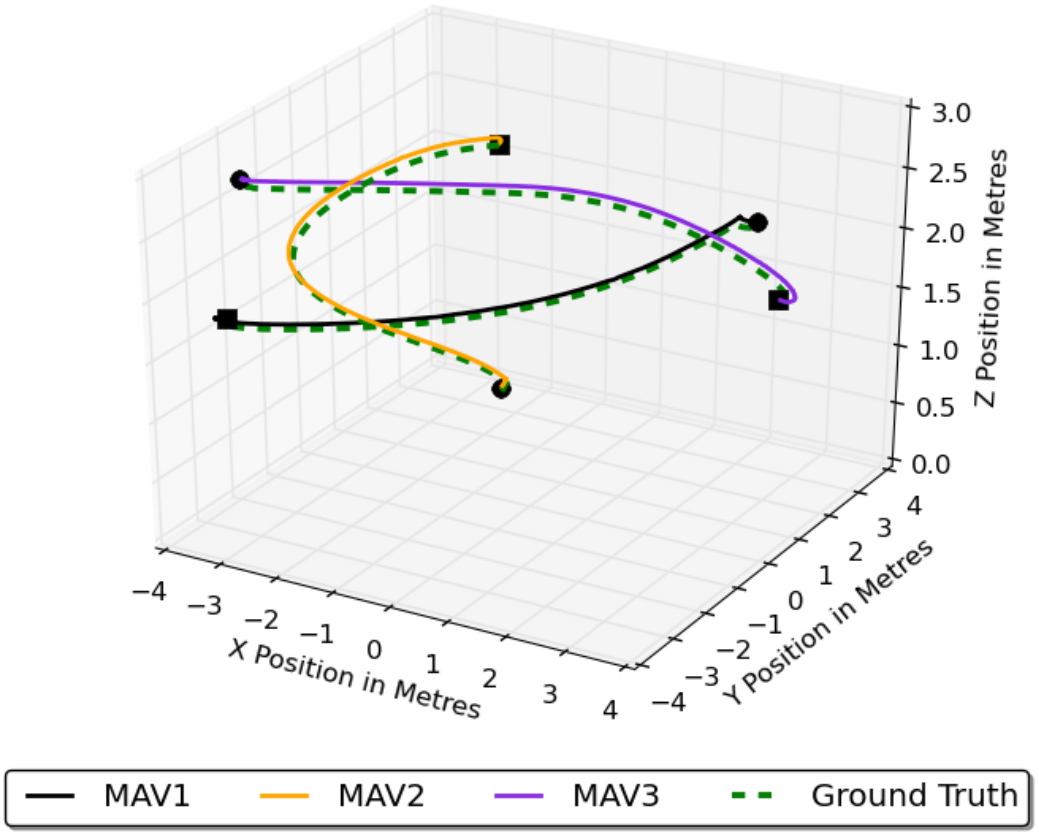


FIGURE 6.17: A representative example of the simulated collision avoidance experiments. The start position of each trajectory is indicated by a circle marker and the end position a square.

MAVs are able to run at full tracking speed (60 frames per second (FPS)) which results in smoother trajectories than the 30 FPS (for a team size of 3 MAVs using CCTAM).

TABLE 6.1: Collision Avoidance Experiment Summary

Team Size	**Simulated Pose**	**CCTAM Pose Estimate**	**DCTAM Pose Estimate**
2	100%	100%	100%
3	100%	97%	100%
4	100%	75%	100%
5	100%	50%	100%
6	100%	-	100%
7	100%	-	100%
8	100%	-	100%

Collision Avoidance Hardware Experiments

The collision avoidance experiment was repeated with the two MAV platforms described in section 6.7 in the same indoor environment as shown in Figure 5.15. The experiment

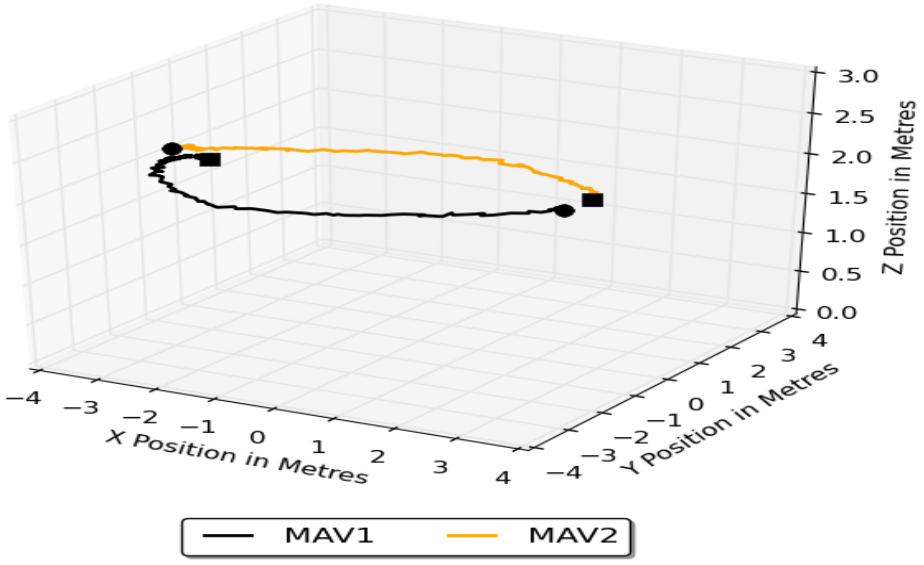


FIGURE 6.18: A representative example of the hardware collision avoidance experiments. The start poisition of each trajectory is indicated by a circle marker and the end position a square.

was repeated 20 times and the resulting collision avoidance rate was 100%. The improved hardware performance matches the results of the simulated experiments. This demonstrates under real world conditions the benefits of running the tracking on-board, state estimation and control on-board.

6.8.7 Multi-Robot Task 2: Exploration

An exploration experiment was conducted using the auction-based multi-robot exploration application described in Chapter 6. The results of this series of experiments is shown in Table 6.2. Unlike CCTAM, DCTAM is not so constrained in terms of team size, it was therefore possible to run the experiment with teams of up to 8 MAVs which resulted in improved execution times. Note however that this reduction reaches a saturation point (after the team size increases past 4 MAVs) where adding additional MAVs does not significantly reduce the execution time. Therefore for an environment of the size used in this simulation (a larger environment would see this saturation point later) it can be seen that 4 MAVs is sufficient to map the environment effectively.

The alignment test (described in Section 5.8.7) is used to verify the accuracy of the resulting map with respect to the ground truth CAD model. As before this experiment does not give precise results as the points generated by mapping correspond to visual features and the ground truth model is a uniform sampling of the model surfaces. An example of the map produced and the reference model are given in Figure 6.19.

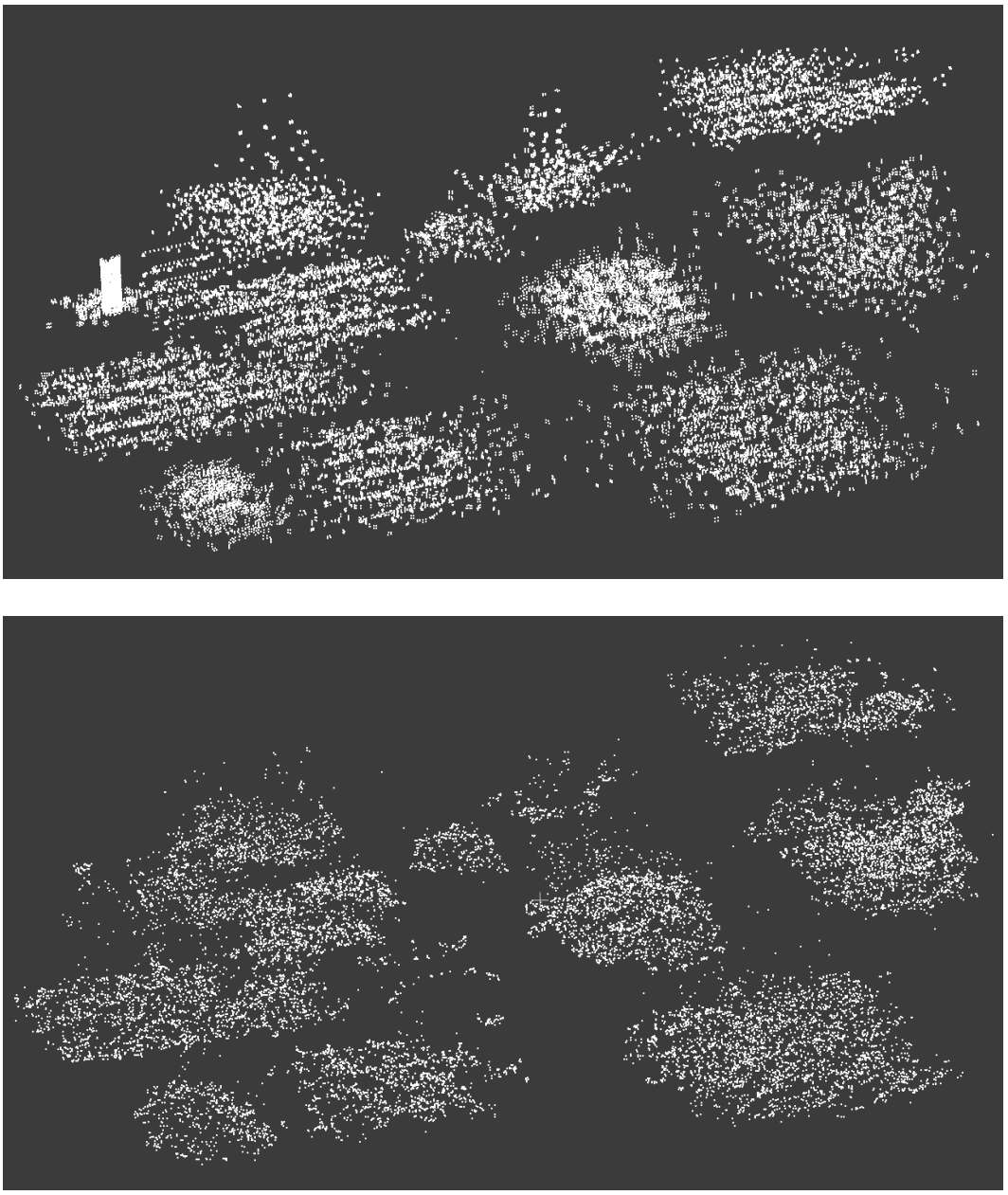


FIGURE 6.19: An example the ground truth model of the simulated environment (top) and the map-points produced by DCTAM (bottom). A comparison of these two point-clouds is used to determine the accuracy of the map produced by DCTAM.

6.8.8 Drift Analysis

Several experiments were also conducted to verify the performance of the G2O implementation of bundle adjustment used in DCTAM. In the first experiment we compare the global bundle adjustment completion times for a single execution on the same map, we ensure both bundle adjustment algorithms used the same parameters in terms of their termination criterion. As the bundle adjustment procedure is non-deterministic the experiment was repeat 20 times (reloading the map for each iteration). The results of this experiment are shown in Table 6.3. As the results show the average completion

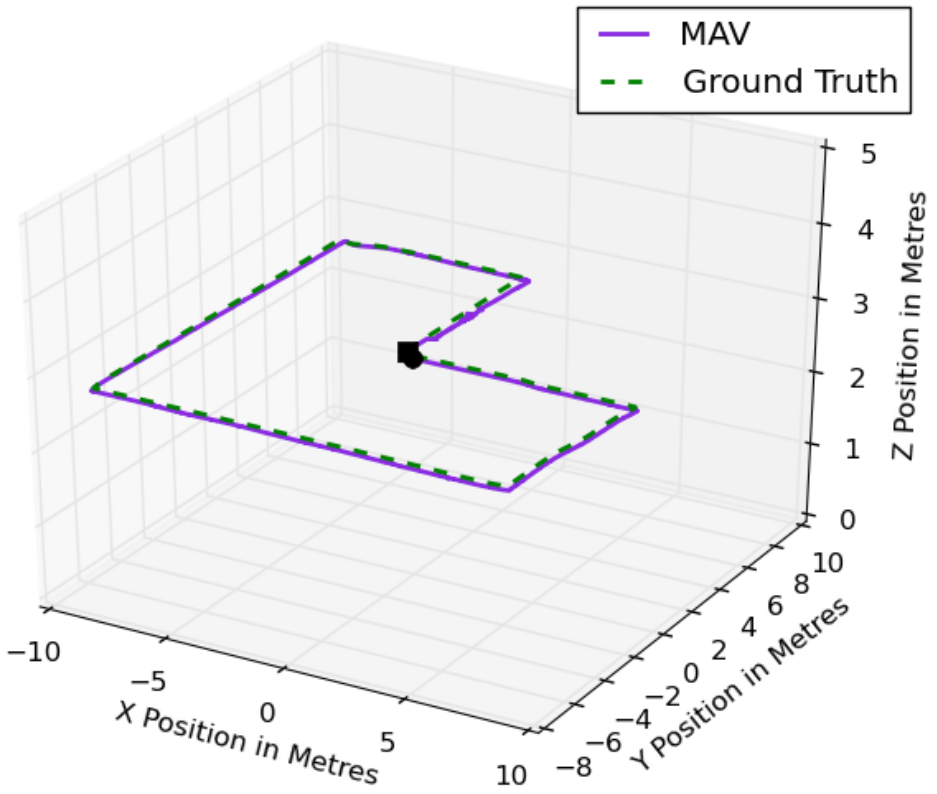


FIGURE 6.20: The figure show the performance of the new bundle adjustment implementation with respect to reducing accumulated drift. The trajectory length was 63 metres and the total accumulated drift was 0.34 metres.

TABLE 6.2: Exploration Experiment Summary

Team Size	Mean Map Size	Mean Execution Time	Mean Alignment RMSE
3	20996	342	0.0972
4	20655	300	0.1035
6	20322	280	0.1002
8	20174	272	0.0966
Mean	**20537**	**299.25**	**0.099375**

time for the G2O bundle adjustment used by DCTAM is significantly smaller particularly on larger maps. This is significant as global bundle adjustment must be interrupted when a new keyframe is added to the map and if this occurs frequently global bundle adjustment may never be performed resulting in a poorly optimised map or even tracking failures. The reduction in completion time afforded by the new implementation allows the global bundle adjustment to complete more frequently generally resulting in more optimised map. This could also have the effect of reducing drift on longer trajectories. To verify this the drift experiment described in Section 5.8.4 was repeated to determine if any improvement was made in the accumulation of drift along a long trajectory. The results of this experiment are shown in Figure 6.20. The length of the trajectory was 63 metres and the accumulated drift was 0.34 metres. However in contrast to CCTAM

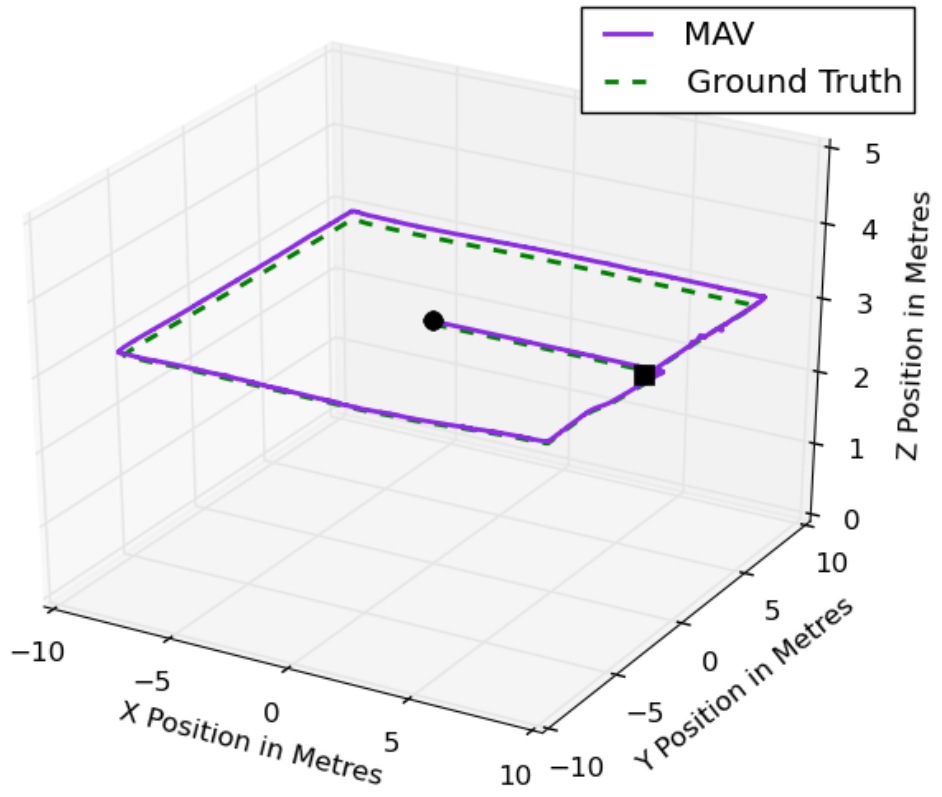


FIGURE 6.21: The figure show the performance of the new bundle adjustment implementation with respect to reducing accumulated drift. The trajectory length was 74 metres and the total accumulated drift was 0.33 metres.

TABLE 6.3: Bundle adjustment computations times on various map sizes

KeyFrames	**MapPoints**	**CCTAM BA Time (mean)**	**DCTAM BA Time (mean)**
98	16791	2.359	1.108
188	10069	4.836	3.621
200	22454	5.196	3.673
337	26885	22.638	10.181

the drift was small enough for DCTAM to close the loop and tracking did not fail. This experiment was repeated only a even longer trajectory of 74 metres (shown in Figure 6.21) where again tracking did not fail and DCTAM was able to implicitly close the loop.

6.8.9 Localisation Performance Hardware

To verify localisation performance with both the DCTAM framework and DCTAM hardware platform we perform a real-world localisation experiment. Here, due to limitations in available space in our Motion Capture Lab only a single MAV is used. The MAV is commanded to fly pre-defined way points describing a square pattern and the experiment was repeated 20 times. A representative example of the results of this experiment

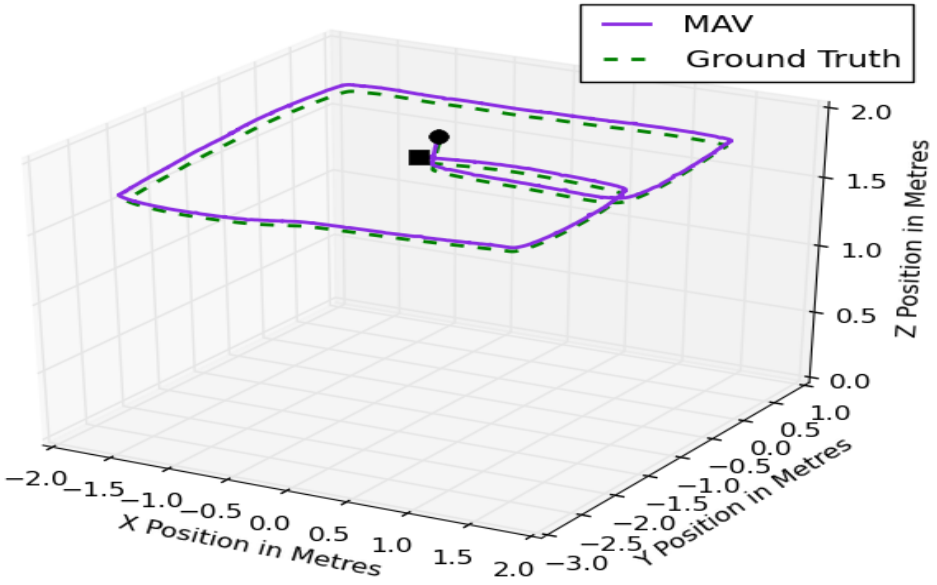


FIGURE 6.22: Results of a physical experiment with a single MAV navigating in a 2mx2m area; the RMS Error for this trajectory was 0.11 metres.

are shown in Figure 6.22. The mean RMSE for these experiments was 0.096 metres demonstrating the similar localisation performance to both our previous experiments with CCTAM and our simulated experiments.

6.8.10 Tracking Performance on Hardware

In this section we conduct experiments to analyse the performance of the DCTAM tracker on a number of commonly available on-board computers. The list of computers as well as their features is given in table 6.4. All on-board DCTAM components (Tracker and EKF) are running on-board and the mapper is running on the same ground-station computer. We use a pre-recorded dataset for consistency although as can be noted there is some variation in the final map sizes. This is accounted for by the non-determinism inherent in systems like DCTAM as even a small fraction of difference in the time a new keyframe is captured can results in a different outcome (in terms of the map produced). Figures 6.23 to 6.26 are dual plots showing the tracking time (left-hand scale) and number of map-points (right-hand scale) for each experiment. The main take-away from these results is the marginal increase in tracking time as the map size grows. From the experimental results tracking time appears almost constant however as discussed in Section 5.4.2 the search for map-points visible in the current frame scales linearly with map size. The experimental results show this has a negligible effect on the tracking time which is instead dominated by the feature extraction and error minimisation steps.

Figure 6.27 shows the average tracking time for each computer together with the variance. The baseline computer, the I7 Desktop, has an average tracking time of 0.01 seconds, meaning on average a maximum update rate of 100 Hz is possible. The Odroid U3 and Raspberry Pi are capable of a maximum update rate of 33 Hz and 25

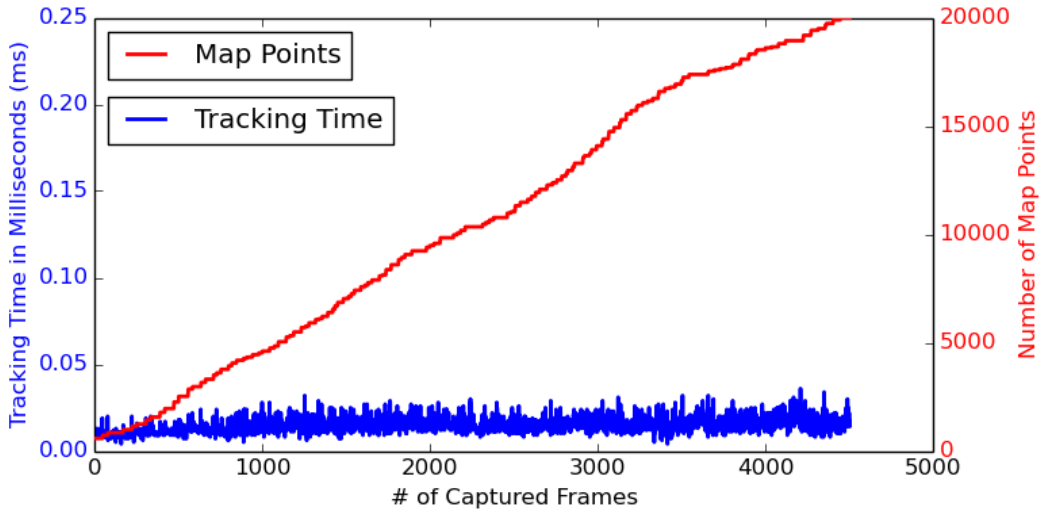


FIGURE 6.23: I7 desktop tracking time.

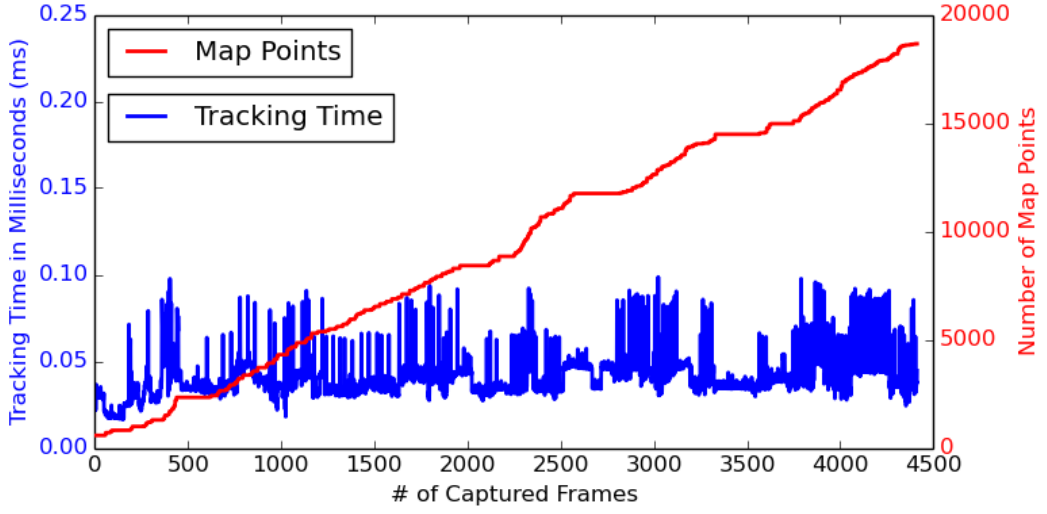


FIGURE 6.24: Odroid U3 tracking time.

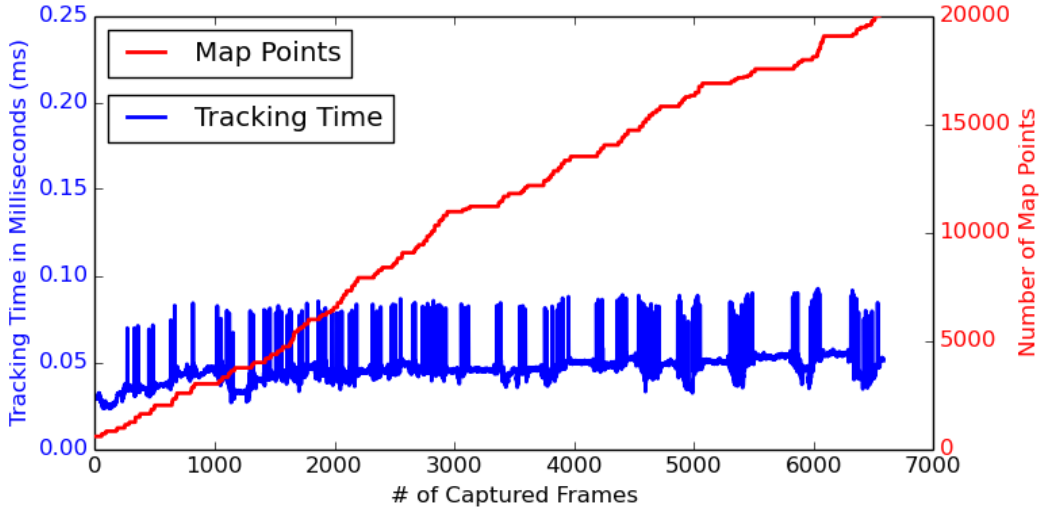


FIGURE 6.25: Raspberry Pi3 tracking time.

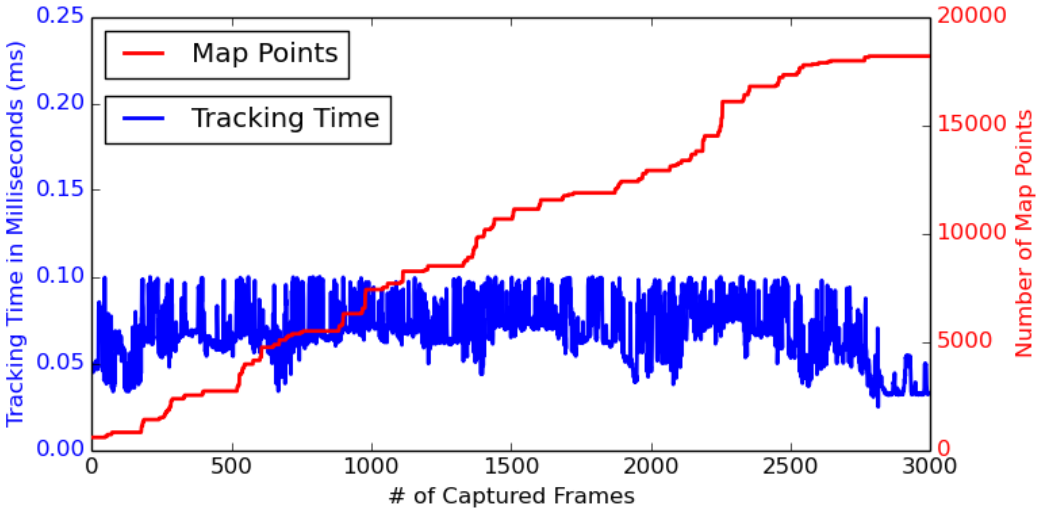


FIGURE 6.26: Intel Atom tracking time.

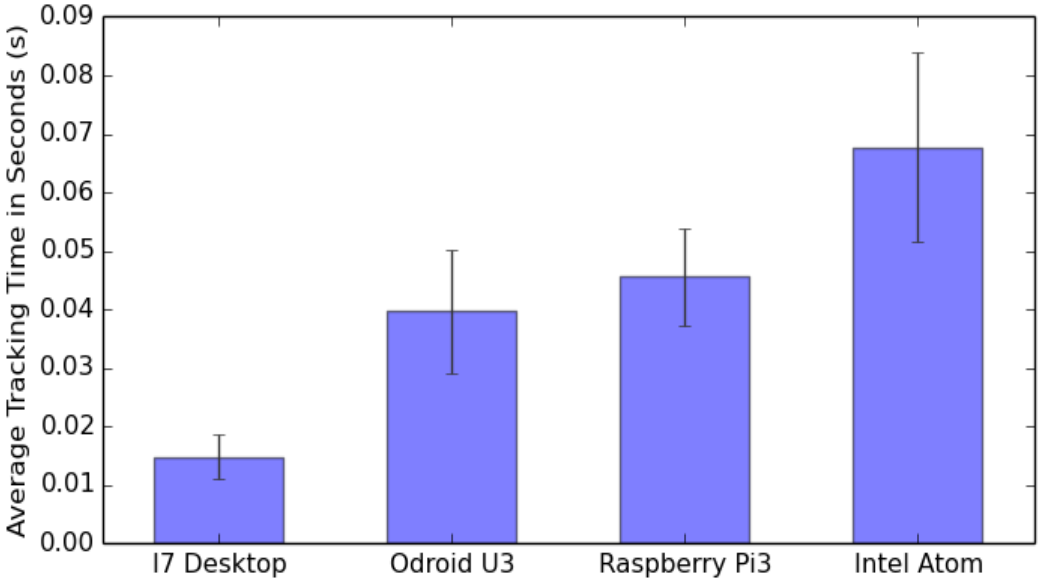


FIGURE 6.27: This figure shows the average computation time for the tracking threads on a number of processors.

TABLE 6.4: Computer platforms used in the tracking performance experiments.

Platform	Processor	Memory
I7 Desktop	3.4 Ghz 8 Core Intel i7	16 GB
Odroid U3	1.7GHz 4 Core ARM Cortex-A9	2 GB
Raspberri Pi3	1.2GHz 4 Core ARMv8	2 GB
Intel ATOM	1.6 GHz Intel Atom N270	2 GB

Hz respectively. The computer with the worst performance is the Intel Atom, with maximum update rate of 14 Hz. Recalling the scalability experiment from Section 5.8.5 where it was found an update rate of 20 Hz is required for reliable operation, before is would be required to reduce the maximum speed of the MAVs. From this experiment it can be seen that DCTAM can be run at an acceptable rate even on low cost, low power hardware such as the Raspberry Pi 3 and Odroid U3. Additionally it can also be used on an Intel Atom based MAV with a reduction in speed.

6.9 Conclusion

This chapter presented a distributed approach to multi MAV visual navigation. The approach expands on the previously presented centralised approach (CCTAM) with a view to improving scalability and robustness. The reported results, from an extensive set of experiments, demonstrated the significant improvements in both scalability and robustness over CCTAM. With these improvements it was demonstrated that the performance of DCTAM with respect to the two multi-robot coordination problems; collision avoidance and exploration also improved. The results obtained demonstrated significant improvement over CCTAM and that complex multi-agent tasks, such as collision avoidance and exploration, can be performed using light weight MAVs with some on-board computational resources.

This distributed approach is also more general than CCTAM and this was verified by performing real world experiments using two custom hardware platforms developed using commonly available off the shelf hardware. The performance of the proposed system (DCTAM) using a number of conventional MAV on-board computers was demonstrated, highlighting the flexibility of the proposed system with respect to its hardware requirements.

The proposed system still had some limitations in terms of its long range capabilities due to the scalability of the optimisation approach employed in this work. This issue was partially addressed with a more efficient bundle adjustment implementation using the generalised graph optimisation framework G2O. The generic nature of this optimisation framework facilitates implementation of more robust and scalable map optimisation techniques such as double-window bundle adjustment and loop closures (as discussed in Section 3.2.3).

Chapter 7

Conclusions and Future Work

This thesis explored the problem of multi-robot visual navigation for MAVs, where a single camera is the primary sensor on board the MAV. Chapter 1 introduced the MAV platform as a small scale aerial robot capable of operating in both indoor and outdoor environments. A specific application, that of disaster response, was introduced in detail to highlight the difficulties with current GPS based navigation solutions and partially motivate the work in this thesis.

Chapter 2 introduced the underlying principles of Computer Vision and Structure From Motion. The focus of this thesis was the application of these techniques to a real-time, multi-robot setting. Specifically the chapter introduced the classical pinhole camera model as well as Devernay and Faugeras' FOV distortion model. These concepts were expanded on in the description of the structure-from-motion problem and the related algorithms (including Perspective-n-Point, Triangulation, Epipolar Geometry and Bundle adjustment) which present key methods to allow the 3D structure and/or camera motion to be reconstructed from a sequence of images.

In Chapter 3 the related work was considered in the context of MAV navigation. Other navigation approaches such a Radio-based, motion capture and Simultaneous Localisation and Mapping (SLAM) were considered. The bulk of the chapter is dedicated to discussing various SLAM approaches and their application to MAV navigation. We highlighted the difficulties of applying the techniques commonly found on ground-based robots (laser-based SLAM) to aerial vehicles, particularly the 3D nature of the problem together with the size and power restrictions of the MAV platform itself. In addition to discussing previous visual SLAM approaches two key problems which motivated the work in this thesis were also highlighted:

1. The high computational complexity of the visual SLAM problem, which means being able to run a full visual SLAM system on-board a MAV is only possible if the map size is severely restricted (8-10 keyframes at most). This limits the usefulness of the visual SLAM approach as no map of any practical use can be created and maintained. Additionally, while overall position drift is reduced, it is not removed using this keyframe-limited approach.

2. Most previous multi-robot visual SLAM approaches focus on the issue of map merging, that is each having each robot create an independent local map and merging these into a common map when an overlap is detected between these local maps. The main drawback of these previous approaches is scalability, each local map must be compared to all other local maps to look for points of overlap so they can be merged. Given that each map can consist of several thousand map-points this is a problem that grows exponentially. This is one of the reasons why the largest team size supported by previous multi-robot visual SLAM approaches was 3 MAVs.

One of the challenges of the work in this thesis was the difficulty in evaluating a multi-robot navigation system beyond a quantitative evaluation of its localisation and mapping performance. A case study approach was used to provide an additional means of evaluation. The performance of the mutli-robot navigation approaches developed in this thesis were analysed in terms of their execution of two typical multi-robot coordination problems. The two scenarios chosen were collision avoidance and exploration. In Chapter 4 these two coordination problems were introduced together with a description of the particular approaches chosen to address them. Specifically a velocity-obstacle based collision avoidance controller was introduced together with an auction-based exploration approach. The velocity obstacle-based collision avoidance approach was chosen as it required accurate position and velocity data to be shared between each MAV; this provided another method to test the consistency and reliability of the underlying localisation approach in a real-time setting. The auction-based exploration controller directly uses the sparse map-points generated by a feature-based visual SLAM system; this provides a method to verify the consistency of this data as well as it's application to the problem of exploration. That is to answer the question is the sparse map data generated by a feature-based visual SLAM system provide sufficient information to allow a team of robots to autonomously explore an environment.

The limitations of previous multi-robot visual SLAM systems (discussed in Chapter 3) in terms of map merging motivated the work presented in Chapter 5. This Chapter introduced a different approach to multi-robot visual SLAM where appearance-based localisation was used to ensure each MAV was able to localise within a common, global map frame before proceeding to map an environment. The feasibility of this approach was explored via the creation of a proof-of-concept visual navigation system we called Centralised Collaborative Tracking and Mapping (CCTAM). CCTAM was a centralised approach to multi-robot visual SLAM, meaning the full system ran completely off-board. This allowed us to evaluate the feasibility of the appearance-based localisation approach to facilitate multi-robot visual navigation. The main benefit of this approach was the ability to perform multi-robot visual navigation without the requirement to do map overlap detection or map merging. This is in contrast to previous approaches such as Foster et al. [28], Riazuelo et al. [95] and Cherbrolu et al. [12]. These map merging

approaches introduce additional computation overhead which limit the scalability (in terms of the number of robots supported) of the system.

The main limitations of the work presented in Chapter 5 were twofold. First, because all computation was carried out off-board, the MAVs were dependant on a reliable, low latency wireless link to ensure good performance. In the presence of network delay the system became unreliable, as demonstrated with the hardware collision avoidance experiments in Section 5.8.6. The second limitation was inherent in the architecture of the system. The multi-threaded approach limited the scalability of the framework. This was demonstrated experimentally in Section 5.8.5 which saw the update rate for each MAV decay almost exponentially with the size of the team. This not only affected the scalability of the system but also the localisation performance. It was discovered that a low update rate (below 15 Hz) drastically reduces the reliability of the framework. This was evident by the noticeable increase in collisions in the simulated experiments (25% collision rate for a team of 4 MAVs and 50% for a team of 5 MAVs). These results motivated the development of a more distributed approach to address the issues of scalability and robustness.

In Chapter 6 a partially distributed framework for multi-robot visual navigation was presented called DCTAM (Distributed Collaborative Tracking and Mapping). The framework built on the previous centralised approach but replaced the multi-threaded architecture with a multi-processing one. This allowed the time critical tasks of frame-to-frame motion tracking to be carried out on board the MAV. This made the system significantly more robust to wireless network delay in comparison to CCTAM. Indeed, this approach allowed the system to handle significant wireless network delay (up to 30 seconds in extreme cases). Additionally, because each MAV stores the complete visual feature map locally a complete network failure only results in a loss of the mapping capability of the MAV. Under these conditions tracking could continue provided that the MAVs stayed within previously mapped areas. DCTAM also made use of an efficient communication model (as well the use of lossless data compression) to ensure the necessary bandwidth required for each MAV was kept low. In addition to the improvements in robustness the framework also demonstrated vastly improved scalability. Indeed DCTAM was capable of operating with as many as 20 MAVs simultaneously exploring an environment. Based on the bandwidth requirements recorded during these experiments the system could potentially scale up to hundreds of MAVs before the communications link became a bottleneck. However such experiments were beyond the limits of our current simulation capabilities and would not have provide more insight into the benefits of the approach beyond what has already been gained. The main insight gained from these scalability experiments was the utility of the distributed architecture.

In comparison to related work (particularly Foster et al. [28]) DCTAM significantly improved on the scalability of previous visual navigation frameworks, as stated previously the largest team supported by previous systems was 3 MAVs (Foster et al.) and

DCTAM was able to scale to 20 MAVs operating simultaneously. This significant increase in scalability opens the door for the wider application of vision-guided MAVs to a host of domains from mapping to surveillance. Additionally in previous work the computational complexity of bundle adjustment forced users to limit the size of the map when running a visual SLAM system on-board a MAV (as discussed in Section 3.2.3). With the partially distributed architecture of DCTAM this limitation was removed and the map size was no longer limited by the processing capabilities of the on-board computer. Additionally in contrast to approaches such as Foster et al. [28] and Cherbrolu et al. [12] the full global map was stored on-board the robots. This provides increased robustness as, if a robot looses contact with the ground station, it will still be able to reliably (i.e. without accumulating position drift) navigate within the previously mapped areas. Another benefit of the DCTAM approach was the low computational requirements of the on-board processes (the DCTAM Tracker). Our experiments showed the system was able to run on very low cost, low power hardware such as the Raspberry Pi. This means the system could be used on small (< 500 grams), more computationally constrained platforms, or in the case of platforms with more computational resources, run along side other software systems to perform tasks such as object recognition or high-level planning.

The remaining limitations of DCTAM are primarily due to the choice of Klien and Murray's Parallel Tracking and Mapping (PTAM) [54] as our starting point for the system. PTAM was originally released in 2009 and since then many advancements in the fields of visual tracking and mapping have been made. Several of these advancements were discussed in Sections 3.2.3 and 3.2.3. In particular the semi-dense tracking approach used in systems like SVO and LSD-SLAM show improved tracking performance for fast motions and self similar structures. These approaches have the added benefit of producing semi-dense depth maps which provide more information for robots for tasks such as static obstacle avoidance. Improvements in optimisation methods, such as Strasdat et al.'s [106] double window optimisation approach, significantly improve the scalability of map optimisation over the standard global bundle adjustment approach used for the work in this thesis.

One possible improvement to DCTAM is the removal of the centralised mapper. The centralised mapper performs two main functions, processing new keyframes to find new map points and local and global map optimisation. Weiss et al. [118] already showed that it was possible to run keyframe limited visual SLAM on board a MAV, this included keyframe processing and bundle adjustment on a very restricted map size. The difficult problem to address would be how best to distribute the computationally complex problem of global map optimisation. A possible solution would be a distributed bundle adjustment. Ni et al. [83] presented such a distributed bundle adjustment solver based on the idea of sub maps. A key insight into their work was the use of local coordinate systems for sub maps. For each sub map a base node was selected and all camera poses and feature positions become relative to the base node. The use of a local

coordinate system allowed the caching of measurement linearisation for use in subsequent iterations of the sub map optimisation. Once the optimisation of sub maps converges, a global optimisation is run over all the base nodes to account for any residual error. Using a similar approach in the context of DCTAM may facilitate a fully distributed visual navigation solution. Each MAV could create a sub map (or multiple sub maps) to optimise only the map features it creates and broadcast the adjustment in a similar fashion to the centralised mapper in DCTAM. Then a single MAV (or the MAVs can take it in turns) could perform the global adjustment over the sub map nodes. The benefit of this approach is to remove the central point of failure of the mapper in DCTAM.

Another limitation of DCTAM is the constraints on map size as a result of the global map optimisation approach used, which has limited scalability and does not address the issues of scale drift and loop closures. One may replace the current mapper with an approach based on more modern map optimisation techniques. These include full seven Degree of Freedom (7DoF) bundle adjustment to correct for both pose and scale drift, the inclusion of loop detection and loop closure and finally a large scale double window optimisation approach. This would allow the system to work with significantly larger maps make it applicable for long term navigation. The would allow the system to be used for long range autonomous vehicles such as cars or fixed wing aircraft.

Although the work in this thesis has focused on robotic applications, and MAVs in particular, there are many other possible applications, particularly for DCTAM. A large number of mobile phone manufacturers including Google and Apple are investing heavily in indoor localisation technologies. This is due in part to the fact that one of the leading uses for modern smart-phones is as a navigation tool. Current GPS technology means this is limited to outdoor navigation. The ability to reliably localise a mobile phone indoors would open up a whole host of indoor navigation based applications such as automated tour guides for popular tourist locations or indoor navigation for large office buildings or shopping centres. These are just some of the possible future research avenues available based on the results presented in this thesis. It is an exciting time in the field of visual navigation and computer vision in general and I look forward to seeing the future research inspired by my thesis as well as the research of my peers.

This work addressed the issues of scalability and robustness for multi-robot visual navigation. A partially distributed approach was presented, which makes use of appearance based localisation to avoid the scalability issues inherent in map-merging approaches and allows the visual SLAM system to run on-board (partially) a MAV without the strict limitation in map size of previous approaches. It was shown via experiment that this approach scales significantly better than previous approaches enabling up to 20 MAVs to navigate within a common coordinate system. This facilitates coordination with large teams of MAVs as shown in the collision avoidance and exploration case studies.

7.1 Publications

The material from Chapters 4 and 5 were presented in publication 1. Material from Chapters 4 and 6 is featured in publications 2 and 3.

[1] **R. Williams**, B. Konev, F. Coenen. Multi-agent environment exploration with AR.Drones. In: Advances in Autonomous Robotics Systems, pp. 60-71. Springer (2014) [122].

[2] **R. Williams**, B. Konev, F. Coenen. Scalable Distributed Collaborative Tracking and Mapping with Micro Aerial Vehicles. In: International Conference on Intelligent Robots and Systems (IROS), 2015 [124].

[3] **R. Williams**, B. Konev, F. Coenen. Collaborating Low Cost Micro Aerial Vehicles: a Demonstration. In: Towards Autonomous Robotic Systems (TAROS), 2015 [123].

Appendix A

Open Source Hardware and Software

FIGURE A.1: The AscTec Firefly MAV platform [110].

A.1 DCTAM Hardware and Software

While there are many commercial MAV platforms available for purchase they are primarily geared towards GPS-based navigation in outdoor environments and thus feature sensor suites and on-board processors geared to those requirements. Of the platforms geared towards research the most popular are the Ascending Technologies line of MAVs such as the AscTec Firefly shown in Figure A.1. Ascending technologies offer a range of configurations in terms of sensors and on-board computing which allows the MAV to be tailored to a specific research application. However the drawback of such as research platform is the cost prohibitive with the firefly costing between four and six thousand

U.S. dollars. Given one of the aims of the research presented in this thesis is making autonomous MAVs more accessible to the research community we decided to focus on more low cost platforms in this work. Therefore for the work on CCTAM we made use of the Parrot AR.Drone platform, a low cost off-the-shelf MAV platform costing between 250 and 300 U.S. dollars. The drawback to using a commercial platform for research is the limited flexibility in terms of sensor and on-board computing.

The recent advent of low cost desktop 3D printers has made a significant impact on the accessibility and cost of small scale custom manufacturing. It has become possible for individuals or institutions to manufacture high quality, custom parts at very low unit cost. For example the parts necessary to construct the MAV platform described in this Chapter can be manufactured for a cost of less than 5 U.S. dollars. The accessibility and low unit cost was the motivation behind the development of a custom 3D printed MAV platform for use with system such as CCTAM and DCTAM. The design features of this MAV platform are as follows:

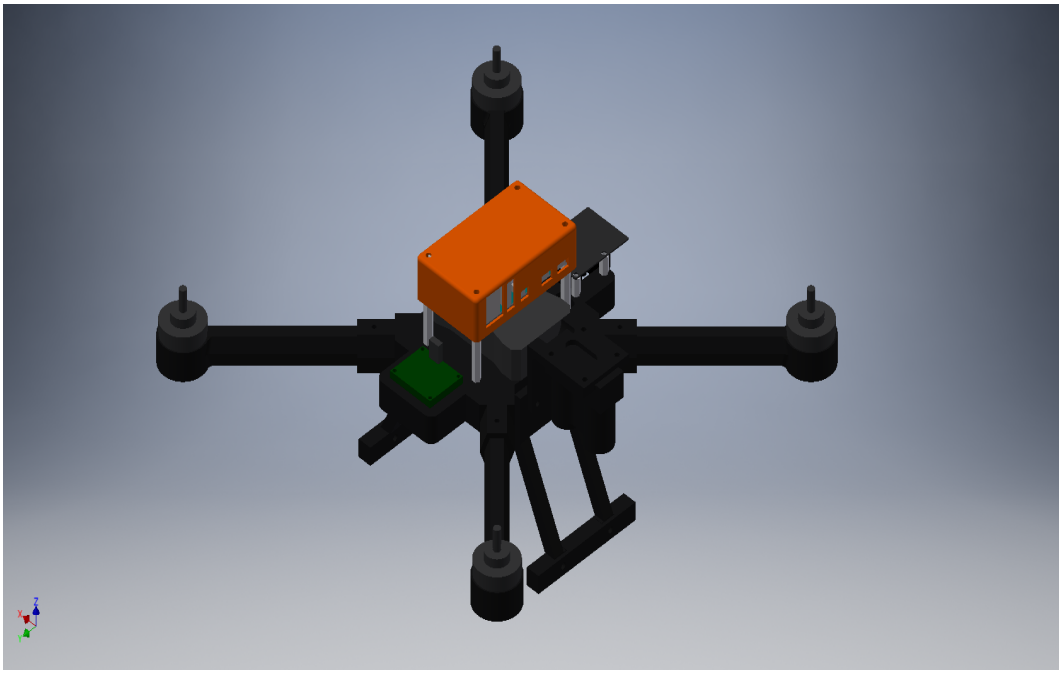


FIGURE A.2: A rendering of the 3D CAD model for the DCTAM hardware platform.

1. Physical dimensions of 26.5 $\times$26.5 $\times$15.5 centimetres for length, width and height respectively, the small size allows the MAV to be flown both indoors and outdoors.
2. Propeller size: 8 inches or 20.32 centimetres.
3. Propulsion system consisting of four Turnigy Aerodrive SK3 2822-1275 brushless motors, a Q-Brain 4 x 25A brushless motor controller. This produces a peak thrust of 1.85 Kilograms.
4. The maximum payload (in addition to the full computer and sensor suite) is 400 grams.

5. Flight time with a 2200 Milli-Amp-Hours (MaH) lithium polymer battery are 10-12 minutes.

6. On-board system feature a Harkernel Odroid U3 single board computer with a Pixhawk Flight controller.

7. Safety Remote Control (RC) is provided via a Fr-Sky D4R-II receiver.

8. Sensor suite includes a Matrixvision BlueFOX MV grayscale camera and Lidar Lite V2 laser distance sensor (for altitude estimation).

FIGURE A.3: The constructed DCTAM platform.

With the goal of making such platforms more accessible the authors make all the CAD models and 3D printable files for this platform available at `https://github.com/richardw347/dctam_hardware`. The 3D printed parts for the second DCTAM platform (based on the AR.Drone frame) are also made available in this repository. A rendering of the full 3D model of the DCTAM MAV platform is shown in Figure A.2 and the completed MAV is shown in Figure A.3.

The complete DCTAM software framework include the on-board components (Tracker, EKF, Controller) and off-board components (Mapper) are also made available together with some installation instructions and example datasets available at `https://www.csc.liv.ac.uk/~rmw/DCTAM.php`

Bibliography

[1] Bardia Alavi and Kaveh Pahlavan, *Modeling of the TOA-based distance measurement error using UWB indoor radio measurements*, IEEE Communications Letters, vol. 10, IEEE, 2006, pp. 275–277.

[2] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart, *Optimal reciprocal collision avoidance for multiple non-holonomic robots*, Distributed Autonomous Robotic Systems, Springer, 2013, pp. 203–216.

[3] Federico Augugliaro, Ammar Mirjan, Fabio Gramazio, Mark Kohler, and Raffaello D'Andrea, *Building tensile structures with flying machines*, Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, 2013, pp. 3487–3492.

[4] Abraham Bachrach, Ruijie He, and Nicholas Roy, *Autonomous Flight in Unknown Indoor Environments*, International Journal of Micro Air Vehicles, vol. 1, 2009, pp. 217–228.

[5] Tim Bailey and Hugh Durrant-Whyte, *Simultaneous localization and mapping (slam): Part ii*, IEEE Robotics & Automation Magazine, vol. 13, IEEE, 2006, pp. 108–117.

[6] Randal Beard, *Quadrotor Dynamics and Control Rev 0.1*, `http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub`, 2008, [Online; accessed 21-September-2016].

[7] Alessandro Benini, Adriano Mancini, and Sauro Longhi, *An imu/uwb/vision-based extended kalman filter for mini-uav localization in indoor environment using 802.15. 4a wireless sensor network*, Journal of Intelligent & Robotic Systems, vol. 70, Springer, 2013, pp. 461–476.

[8] Joydeep Biswas and Manuela Veloso , *Wifi localization and navigation for autonomous indoor mobile robots*, Robotics and Automation (ICRA), 2010 IEEE International Conference on, May 2010, pp. 4379–4384.

[9] Jack E Bresenham, *Algorithm for computer control of a digital plotter*, IBM Systems journal, vol. 4, IBM, 1965, pp. 25–30.

[10] Luca Carlone, M. Kaouk Ng, Jingjing Du, Basilio Bona, and Marina Indri, *Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication*, International Conference on Robotics and Automation (ICRA), IEEE, 2010, pp. 243–249.

[11] Robert Castle, Georg Klein, and David W. Murray, *Video-rate localization in multiple maps for wearable augmented reality*, Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on, IEEE, 2008, pp. 15–22.

[12] Nived Chebrolu, David Marquez-Gamez, and Philippe Martinet, *Collaborative Visual SLAM Framework for a Multi-Robot System*, 7th Workshop on Planning, Perception and Navigation for Intelligent Vehicles IROS 2015, 2015, pp. 59–64.

[13] Alessandro Chiuso, Paolo Favaro, Hailin Jin, and Stefano Soatto, *Structure from motion causally integrated over time*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, IEEE, 2002, pp. 523–535.

[14] Javier Civera, Andrew J. Davison, and J.M. Martinez Montiel, *Inverse depth parametrization for monocular SLAM*, Robotics, IEEE Transactions on, vol. 24, IEEE, 2008, pp. 932–945.

[15] Mark Cummins and Paul Newman, *Appearance-only slam at large scale with fab-map 2.0*, The International Journal of Robotics Research, vol. 30, SAGE Publications, 2011, pp. 1100–1123.

[16] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse, *MonoSLAM: Real-time single camera SLAM*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 29, IEEE, 2007, pp. 1052–1067.

[17] ArduPilot Developers, *Mouse based Optical Flow Sensor (ADNS3080)*, `http://ardupilot.org/copter/docs/common-mouse-based-optical-flow-sensor-adns3080.html`, 2016, [Online; accessed 21-September-2016].

[18] Frédréric Devernay and Olivier Faugeras, *Straight Lines Have to Be Straight: Automatic Calibration and Removal of Distortion from Scenes of Structured Enviroments*, Mach. Vision Appl. (Secaucus, NJ, USA), vol. 13, Springer-Verlag New York, Inc., August 2001, pp. 14–24.

[19] Civil Engineering Dictionary, *Sources of errors in gps*, `http://www.aboutcivil.org/sources-of-errors-in-gps.html`, Accessed: 2016-01-26.

[20] Hugh Durrant-Whyte and Tim Bailey, *Simultaneous localization and mapping: part I*, IEEE Robotics & Automation Magazine, vol. 13, IEEE, 2006, pp. 99–110.

[21] Ethan Eade and Tom Drummond, *Scalable monocular SLAM*, Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, vol. 1, IEEE, 2006, pp. 469–476.

[22] Marcus Eliasson, *Crazyflie Nano Quadcopter*, `https://www.bitcraze.io/wp-content/uploads/2013/01/cf_video.png`, 2013, [Online; accessed 21-September-2016].

[23] Jakob Engel, Thomas Schöps, and Daniel Cremers, *LSD-SLAM: Large-scale direct monocular SLAM*, Computer Vision–ECCV 2014, Springer, 2014, pp. 834–849.

[24] Jakob Engel, Jürgen Sturm, and Daniel Cremers, *Scale-aware navigation of a low-cost quadrocopter with a monocular camera*, Robotics and Autonomous Systems, vol. 62, Elsevier, 2014, pp. 1646–1656.

[25] Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza, *Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle*, Journal of Field Robotics, vol. 33, 2016, pp. 431–450.

[26] Paolo Fiorini and Zvi Shiller, *Motion planning in dynamic environments using velocity obstacles*, The International Journal of Robotics Research, vol. 17, SAGE Publications, 1998, pp. 760–772.

[27] Flyability, *Gimball: collision-tolerant flying robot*, `http://www.flyability.com/`, 2016, [Online; accessed 21-September-2016].

[28] Christian Forster, Simon Lynen, Laurent Kneip, and Davide Scaramuzza, *Collaborative monocular SLAM with multiple micro aerial vehicles*, International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ, IEEE, 2013, pp. 3962–3970.

[29] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza, *SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems*, IEEE Transactions on Robotics, IEEE, 2016, pp. 249–265.

[30] Joscha Fossel, Daniel Hennes, Daniel Claes, Sjriek Alers, and Karl Tuyls, *OctoSLAM: A 3D mapping approach to situational awareness of unmanned aerial vehicles*, Unmanned Aircraft Systems (ICUAS), 2013 International Conference on, IEEE, 2013, pp. 179–188.

[31] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun, *A probabilistic approach to collaborative multi-robot localization*, Autonomous Robots, vol. 8, Springer, 2000, pp. 325–344.

[32] Friedrich Fraundorfer, Petri Tanskanen, and Marc Pollefeys, *A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles*, Computer Vision–ECCV 2010, Springer, 2010, pp. 269–282.

[33] S. Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez, *Automatic generation and detection of highly reliable fiducial markers under occlusion*, Pattern Recognition, vol. 47, Elsevier, 2014, pp. 2280–2292.

[34] S. Gecizi, Z. Tian, G.B. Giannakis, Z. Sahinoglu, H. Kobayashi, A.F. Molisch, and H.V. Poor, *Localization via ultra-wideband radios*, IEEE Communications Magazine, vol. 43, 2005, pp. 70–84.

[35] Daniel Gilamn and Matthew Easton, *Unmanned Aerial Vehicles in Humanitarian Response*, Tech. report, United Nations Office for the Coordinations of Humanitarian Affaris (OCHA), 2014.

[36] Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard, *Towards a navigation system for autonomous indoor flying*, Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE, 2009, pp. 2878–2883.

[37] Shweta Gupte, Paul Infant Teenu Mohandas, and James M Conrad, *A survey of quadrotor unmanned aerial vehicles*, Southeastcon, 2012 Proceedings of IEEE, IEEE, 2012, pp. 1–6.

[38] Adam Harmat, Michael Trentini, and Inna Sharf, *Multi-Camera Tracking and Mapping for Unmanned Aerial Vehicles in Unstructured Environments*, Journal of Intelligent and Robotic Systems, vol. 78, 2015, pp. 291–317.

[39] Chris Harris and Mike Stephens, *A combined corner and edge detector*, In Proc. of Fourth Alvey Vision Conference, 1988, pp. 147–151.

[40] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2003.

[41] Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys, *Autonomous visual mapping and exploration with a micro aerial vehicle*, Journal of Field Robotics, vol. 31, Wiley Online Library, 2014, pp. 654–675.

[42] S. Heymann, K. Müller, A. Smolic, B. Froehlich, and T. Wiegand, *SIFT implementation and optimization for general-purpose GPU*, International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), Computer Science Research Notes, 2007, pp. 317–322.

[43] Martin Hirzer, *Marker detection for augmented reality applications*, Seminar/Project Image Analysis Graz, 2008, pp. 1–2.

[44] Gabe Hoffmann, Dev Gorur Rajnarayan, Steven L. Waslander, David Dostal, Jung Soon Jang, and Claire J. Tomlin, *The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)*, Digital Avionics Systems Conference, 2004. DASC 04. The 23rd, vol. 2, IEEE, 2004, pp. 12.E.4–121–10.

[45] Dominik Honegger, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys, *An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications*, Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 1736–1741.

[46] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Dieter Fox, and Nicholas Roy, *Visual odometry and mapping for autonomous flight using an rgb-d camera*, In Proc. of the Intl. Sym. of Robot. Research, 2011.

[47] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin, *Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering*, Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE, 2009, pp. 3277–3282.

[48] Joseph Huang, David Millman, Morgan Quigley, David Stavens, Sebastian Thrun, and Alok Aggarwal, *Efficient, generalized indoor wifi graphslam*, Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 1038–1043.

[49] Arnold Irschara, Christof Hoppe, Horst Bischof, and Stefan Kluckner, *Efficient structure from motion with weak position and orientation priors*, CVPR 2011 WORKSHOPS, IEEE, 2011, pp. 21–28.

[50] Rudolph Emil Kalman, *A new approach to linear filtering and prediction problems*, Journal of basic Engineering, vol. 82, American Society of Mechanical Engineers, 1960, pp. 35–45.

[51] Farid Kendoul, *Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems*, Joural of Field Robotics (Chichester, UK), vol. 29, John Wiley and Sons Ltd., March 2012, pp. 315–378.

[52] Christian Kerl, Jurgen Sturm, and Daniel Cremers, *Dense visual slam for rgb-d cameras*, Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, 2013, pp. 2100–2106.

[53] Seung-Hun Kim, Chi-Won Roh, Sung-Chul Kang, and Min-Yong Park, *Outdoor navigation of a mobile robot using differential gps and curb detection*, Proceedings 2007 IEEE International Conference on Robotics and Automation, IEEE, 2007, pp. 3414–3419.

[54] Georg Klein and David Murray, *Parallel tracking and mapping for small AR workspaces*, International Symposium on Mixed and Augmented Reality ISMAR 2007, IEEE, 2007, pp. 225–234.

[55] Georg Klein and David Murray, *Improving the agility of keyframe-based SLAM*, Computer Vision–ECCV 2008, Springer, 2008, pp. 802–815.

[56] Sven Koenig, C. Tovey, M. Lagoudakis, V. Markakis, David Kempe, Pinar Keskinocak, A. Kleywegt, Adam Meyerson, and Sonal Jain, *The power of sequential single-item auctions for agent coordination*, Proceedings of the national conference on artificial intelligence, vol. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, pp. 1625–1630.

[57] Sivanand Krishnan, Pankaj Sharma, Zhang Guoping, and Ong Hwee Woon, *A uwb based localization system for indoor robot navigation*, 2007 IEEE International Conference on Ultra-Wideband, IEEE, 2007, pp. 77–82.

[58] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard, *g2o: A general framework for graph optimization*, International Conference on Robotics and Automation (ICRA), IEEE, 2011, pp. 3607–3613.

[59] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar, *Towards a swarm of agile micro quadrotors*, Autonomous Robots, vol. 35, Springer, 2013, pp. 287–300.

[60] Michail G. Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J. Kleywegt, Sven Koenig, Craig A. Tovey, Adam Meyerson, and Sonal Jain, *Auction-Based Multi-Robot Routing.*, Robotics: Science and Systems, vol. 5, Rome, Italy, 2005, pp. 343–350.

[61] Joon-Yong Lee and Robert A. Scholtz, *Ranging in a dense multipath environment using an UWB radio link*, IEEE Journal on Selected Areas in Communications, vol. 20, IEEE, 2002, pp. 1677–1683.

[62] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart, *Brisk: Binary robust invariant scalable keypoints*, Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2548–2555.

[63] Stefan Leutenegger, Paul Timothy Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart, *Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization.*, Robotics: Science and Systems, 2013.

[64] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale, *Keyframe-based visual–inertial odometry using nonlinear optimization*, The International Journal of Robotics Research, vol. 34, SAGE Publications, 2015, pp. 314–334.

[65] Kenneth Levenberg, *A method for the solution of certain non-linear problems in least squares*, Quarterly of applied mathematics, vol. 2, JSTOR, 1944, pp. 164–168.

[66] H. C. Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections*, Readings in Computer Vision: Issues, Problems, Principles, and Paradigms (Martin A. Fischler and Oscar Firschein, eds.), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987, pp. 61–62.

[67] Manolis IA Lourakis and Antonis A. Argyros, *SBA: A software package for generic sparse bundle adjustment*, ACM Transactions on Mathematical Software (TOMS), vol. 36, ACM, 2009, pp. 2–30.

[68] David G. Lowe, *Distinctive image features from scale-invariant keypoints*, International journal of computer vision, vol. 60, Springer, 2004, pp. 91–110.

[69] Bruce D. Lucas, Takeo Kanade, et al., *An iterative image registration technique with an application to stereo vision.*, IJCAI, vol. 81, 1981, pp. 674–679.

[70] Carlos Luis and Jérôme Le Ny, *Design of a trajectory tracking controller for a nanoquadcopter*, arXiv preprint arXiv:1607.02565, 2016.

[71] Simon Lynen, Markus W. Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart, *A robust and modular multi-sensor fusion approach applied to mav navigation*, 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 3923–3929.

[72] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff, *Methods for non-linear least squares problems*, `http://soe.rutgers.edu/~meer/GRAD561/ADD/nonlinadvanced.pdf`, 2004, [Online; accessed 21-September-2016].

[73] Elmar Mair, Gregory D. Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger, *Adaptive and generic corner detection based on the accelerated segment test*, Computer Vision–ECCV, Springer, 2010, pp. 183–196.

[74] Donald W. Marquardt, *An algorithm for least-squares estimation of nonlinear parameters*, Journal of the society for Industrial and Applied Mathematics, vol. 11, SIAM, 1963, pp. 431–441.

[75] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys, *Pixhawk: A system for autonomous flight using onboard computer vision*, Robotics and automation (ICRA), 2011 IEEE international conference on, IEEE, 2011, pp. 2992–2997.

[76] Daniel Mellinger, Nathan Michael, and Vijay Kumar, *Trajectory generation and control for precise aggressive maneuvers with quadrotors*, The International Journal of Robotics Research, vol. 31, SAGE Publications Sage UK: London, England, 2012, pp. 664–674.

[77] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar von Stryk, *Comprehensive simulation of quadrotor UAVs using ROS and Gazebo*, Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2012, pp. 400–411.

[78] Marius Muja and David G. Lowe, *Scalable nearest neighbor algorithms for high dimensional data*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 36, IEEE, 2014, pp. 2227–2240.

[79] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, *Orb-slam: A versatile and accurate monocular slam system*, IEEE Transactions on Robotics, vol. 31, Oct 2015, pp. 1147–1163.

[80] Robin R Murphy, Satoshi Tadokoro, and Alexander Kleiner, *Disaster robotics*, Springer Handbook of Robotics, Springer, 2016, pp. 1577–1604.

[81] United States Navy, *United states navy photo database*, `http://www.navy.mil/management/photodb/photos/061114-N-9671T-146.jpg`, 2016, [Online; accessed 21-September-2016].

[82] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison, *DTAM: Dense tracking and mapping in real-time*, Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2320–2327.

[83] Kai Ni, Drew Steedly, and Frank Dellaert, *Out-of-core bundle adjustment for large-scale 3d reconstruction*, 2007 IEEE 11th International Conference on Computer Vision, IEEE, 2007, pp. 1–8.

[84] Matthias Nieuwenhuisen, David Droeschel, Marius Beul, and Sven Behnke, *Autonomous Navigation for Micro Aerial Vehicles in Complex GNSS-denied Environments*, Journal of Intelligent & Robotic Systems, Springer, 2015, pp. 1–18.

[85] Janosch Nikolic, Joern Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T. Furgale, and Roland Siegwart, *A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM*, Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 431–437.

[86] David Nistér, *An efficient solution to the five-point relative pose problem*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 26, IEEE, 2004, pp. 756–770.

[87] David Nister and Henrik Stewenius, *Scalable recognition with a vocabulary tree*, Computer vision and pattern recognition, 2006 IEEE computer society conference on, vol. 2, IEEE, 2006, pp. 2161–2168.

[88] M. Ocana, L.M. Bergasa, M.A. Sotelo, J. Nuevo, and R. Flores, *Indoor robot localization system using wifi signal measure and minimizing calibration effort*, Proceedings of the IEEE International Symposium on Industrial Electronics, vol. 4, 2005, pp. 1545–1550.

[89] Kazunori Ohno, Takashi Tsubouchi, Bunji Shigematsu, Shoichi Maeyama, and Shinichi Yuta, *Outdoor navigation of a mobile robot between buildings based on dgps and odometry data fusion*, Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, vol. 2, IEEE, 2003, pp. 1978–1984.

[90] Helen Oleynikova, Dominik Honegger, and Marc Pollefeys, *Reactive avoidance using embedded stereo vision for mav flight*, 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 50–56.

[91] Vicente Matellán Olivera, José María Cañas Plaza, and Oscar Serrano Serrano, *Wifi localization methods for autonomous robots*, Robotica, vol. 24, Cambridge Univ Press, 2006, pp. 455–461.

[92] Cedric Pradalier, Samir Bouabdallah, Pascal Gohl, Matthias Egli, Gilles Caprari, and Roland Siegwart, *The coax micro-helicopter: A flying platform for education and research*, Advances in Autonomous Mini Robots, Springer, 2012, pp. 89–99.

[93] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng, *ROS: an open-source Robot Operating System*, ICRA workshop on open source software, vol. 3, 2009, pp. 5–10.

[94] B. D. W. Remes, P. Esden-Tempski, F. van Tienen, E. Smeur, C. De Wagter, and G. C. H. E. de Croon, *Lisa-s 2.8 g autopilot for GPS-based flight of MAVs*, IMAV 2014: International Micro Air Vehicle Conference and Competition 2014, Delft, The Netherlands, August 12-15, 2014, Delft University of Technology, 2014, pp. 280–285.

[95] Luis Riazuelo, Javier Civera, and J.M.M Montiel, *C 2 TAM: A Cloud framework for cooperative tracking and mapping*, Robotics and Autonomous Systems, vol. 62, Elsevier, 2014, pp. 401–413.

[96] Edward Rosten and Tom Drummond, *Fusing points and lines for high performance tracking*, Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, vol. 2, IEEE, 2005, pp. 1508–1515.

[97] Edward Rosten and Tom Drummond, *Machine learning for high-speed corner detection*, Computer Vision–ECCV 2006, Springer, 2006, pp. 430–443.

[98] Jose Luis Sanchez-Lopez, Jesús Pestana, Paloma de la Puente, and Pascual Campoy, *A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation*, Journal of Intelligent & Robotic Systems, Springer, 2015, pp. 1–19.

[99] Davide Scaramuzza, *1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints*, International journal of computer vision, vol. 95, Springer, 2011, pp. 74–85.

[100] Davide Scaramuzza and Friedrich Fraundorfer, *Visual odometry [tutorial]*, Robotics & Automation Magazine, IEEE, vol. 18, IEEE, 2011, pp. 80–92.

[101] Adam Schmidt, *Multi-robot, EKF-Based Visual SLAM System*, Computer Vision and Graphics, Springer, 2014, pp. 562–569.

[102] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar, *Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor*, Robotics: Science and Systems, vol. 1, 2013.

[103] Jianbo Shi and Carlo Tomasi, *Good features to track*, Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on, IEEE, 1994, pp. 593–600.

[104] Jamie Snape and Dinesh Manocha, *Navigating multiple simple-airplanes in 3d workspace*, Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE, 2010, pp. 3974–3980.

[105] Jamie Snape, Jur Van den Berg, Stephen J. Guy, and Dinesh Manocha, *The hybrid reciprocal velocity obstacle*, IEEE Transactions on Robotics, vol. 27, IEEE, 2011, pp. 696–706.

[106] Hauke Strasdat, Andrew J Davison, JMM Montiel, and Kurt Konolige, *Double window optimisation for constant time visual slam*, 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2352–2359.

[107] Hauke Strasdat, JMM Montiel, and Andrew J. Davison, *Scale Drift-Aware Large Scale Monocular SLAM.*, Robotics: Science and Systems, vol. 2, 2010, pp. 73–80.

[108] Kelvin Sung, Peter Shirley, and Steven Baer, *Essentials of interactive computer graphics: concepts and implementation*, CRC Press, 2008.

[109] Richard Szeliski, *Computer vision: algorithms and applications*, Springer Science & Business Media, 2010.

[110] Ascending Technologies, *AscTec Firefly*, `http://wiki.asctec.de/display/AR/AscTec+Firefly`, 2013, [Online; accessed 21-September-2016].

[111] Sebastian Thrun, Wolfram Burgard, Dieter Fox, et al., *Probabilistic robotics*, vol. 1, MIT press Cambridge, 2005.

[112] Sebastian Thrun et al., *Robotic mapping: A survey*, Exploring artificial intelligence in the new millennium, vol. 1, 2002, pp. 1–35.

[113] Sjoerd Tijmons, Guido de Croon, Bart Remes, Christophe De Wagter, Rick Ruijsink, Erik-Jan van Kampen, and Qiping Chu, *Stereo vision based obstacle avoidance on flapping wing mavs*, Advances in aerospace guidance, navigation and control, Springer, 2013, pp. 463–482.

[114] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon, *Bundle adjustmenta modern synthesis*, International workshop on vision algorithms, Springer, 1999, pp. 298–372.

[115] Jur Van den Berg, Ming Lin, and Dinesh Manocha, *Reciprocal velocity obstacles for real-time multi-agent navigation*, Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, IEEE, 2008, pp. 1928–1935.

[116] Veeravalli Seshadri Varadarajan, *Lie groups, Lie algebras, and their representations*, vol. 102, Springer Science & Business Media, 2013.

[117] Eric A. Wan and Ronell Van der Merwe, *The unscented Kalman filter for nonlinear estimation*, Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000, IEEE, 2000, pp. 153–158.

[118] Stephan M. Weiss, *Vision based navigation for micro helicopters*, Ph.D. thesis, Eidgenssische Technische Hochschule (ETH) Zurich, 2012.

[119] Andreas Wendel, *Scalable Visual Navigation for Micro Aerial Vehicles using Geometric Prior Knowledge*, Ph.D. thesis, Graz University of Technology, 2013.

[120] Andreas Wendel, Michael Maurer, Gottfried Graber, Thomas Pock, and Horst Bischof, *Dense reconstruction on-the-fly*, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1450–1457.

[121] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós, *A comparison of loop closing techniques in monocular slam*, Robotics and Autonomous Systems, vol. 57, Elsevier, 2009, pp. 1188–1197.

[122] Richard Williams, Boris Konev, and Frans Coenen, *Multi-agent Environment Exploration with AR. Drones*, Advances in Autonomous Robotics Systems, Springer, 2014, pp. 60–71.

[123] ———, *Collaborating low cost micro aerial vehicles: A demonstration*, Conference Towards Autonomous Robotic Systems, Springer, 2015, pp. 296–302.

[124] ———, *Scalable distributed collaborative tracking and mapping with micro aerial vehicles*, Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, 2015, pp. 3092–3097.

[125] Brian Yamauchi, *A frontier-based approach for autonomous exploration*, Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on, IEEE, 1997, pp. 146–151.

[126] Xiang Zhang, Stephan Fronz, and Nassir Navab, *Visual marker detection and decoding in AR systems: A comparative study*, Proceedings of the 1st International Symposium on Mixed and Augmented Reality, IEEE Computer Society, 2002, pp. 97–106.

[127] Xiaoming Zheng, Sven Koenig, and Craig Tovey, *Improving sequential single-item auctions*, 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, pp. 2238–2244.

[128] Danping Zou and Ping Tan, *Coslam: Collaborative visual slam in dynamic environments*, IEEE transactions on pattern analysis and machine intelligence, vol. 35, IEEE, 2013, pp. 354–366.

[129] Andrew Zulu, Samuel John, et al., *A Review of Control Algorithms for Autonomous Quadrotors*, Open Journal of Applied Sciences, vol. 4, Scientific Research Publishing, 2014, pp. 547–556.