Visual SLAM for Autonomous Navigation of MAVs

Visual SLAM for Autonomous Navigation of MAVs

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard-Karls-Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Shaowu Yang
aus Chongqing, China

Tübingen
2014

Tag der mündlichen Qualifikation:	28.07.2014
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Andreas Zell
2. Berichterstatter:	Prof. Dr. Andreas Schilling

Abstract

This thesis focuses on developing onboard visual simultaneous localization and mapping (SLAM) systems to enable autonomous navigation of micro aerial vehicles (MAVs), which is still a challenging topic considering the limited payload and computational capability that an MAV normally has. In MAV applications, the visual SLAM systems are required to be very efficient, especially when other visual tasks have to be done in parallel. Furthermore, robustness in pose tracking is highly desired in order to enable safe autonomous navigation of an MAV in three-dimensional (3D) space. These challenges motivate the work in this thesis in the following aspects.

Firstly, the problem of visual pose estimation for MAVs using an artificial landmark is addressed. An artificial neural network (ANN) is used to robustly recognize this visual marker in cluttered environments. Then a computational projective-geometry method is implemented for relative pose computation based on the retrieved geometry information of the visual marker. The presented vision system can be used not only for pose control of MAVs, but also for providing accurate pose estimates to a monocular visual SLAM system serving as an automatic initialization module for both indoor and outdoor environments.

Secondly, autonomous landing on an arbitrarily textured landing site during autonomous navigation of an MAV is achieved. By integrating an efficient local-feature-based object detection algorithm within a monocular visual SLAM system, the MAV is able to search for the landing site autonomously along a predefined path, and land on it once it has been found. Thus, the proposed monocular visual solution enables autonomous navigation of an MAV in parallel with landing site detection. This solution relaxes the assumption made in conventional vision-guided landing systems, which is that the landing site should be located inside the field of view (FOV) of the vision system before initiating the landing task.

The third problem that is addressed in this thesis is multi-camera visual SLAM for robust pose tracking of MAVs. Due to the limited FOV of a single camera, pose tracking using monocular visual SLAM may easily fail when the MAV navigates in unknown environments. Previous work addresses this problem mainly by fusing information from other sensors, like an inertial measurement unit (IMU), to achieve robustness of the whole system, which does not improve the robustness of visual SLAM itself. This thesis investigates solutions for improving the pose tracking robustness of a visual SLAM system by utilizing multiple cameras. A mathematical analysis of how measurements from multiple cameras should be integrated in the optimization of visual SLAM is provided. The resulting theory allows those measurements to be used for both robust pose tracking

and map updating of the visual SLAM system. Furthermore, such a multi-camera visual SLAM system is modified to be a robust constant-time visual odometry. By integrating this visual odometry with an efficient back-end which consists of loop-closure detection and pose-graph optimization processes, a near-constant time multi-camera visual SLAM system is achieved for autonomous navigation of MAVs in large-scale environments.

Kurzfassung

Diese Arbeit konzentriert sich auf die Entwicklung von integrierten Systemen zur gleichzeitigen Lokalisierung und Kartierung (Simultaneous Localization and Mapping, SLAM) mit Hilfe visueller Sensoren, um die autonome Navigation von kleinen Luftfahrzeugen (Micro Aerial Vehicles, MAVs) zu ermöglichen. Dies ist noch immer ein anspruchsvolles Thema angesichts der meist begrenzten Nutzlast und Rechenleistung eines MAVs. Die dafür eingesetzten visuellen SLAM Systeme müssen sehr effizient zu sein, vor allem wenn parallel noch andere visuelle Aufgaben durchgeführt werden sollen. Darüber hinaus ist eine robuste Positionsschätzung sehr wichtig, um die sichere autonome Navigation des MAVs im dreidimensionalen (3D) Raum zu ermöglichen. Diese Herausforderungen motivieren die vorliegende Arbeit gemäß den folgenden Gesichtspunkten:

Zuerst wird das Problem bearbeitet, die Pose eines MAVs mit Hilfe einer künstlichen Markierung visuell zu schätzen. Ein künstliches neuronales Netz wird verwendet, um diese visuelle Markierung auch in anspruchsvollen Umgebungen zuverlässig zu erkennen. Anschließend wird ein Verfahren aus der projektiven Geometrie eingesetzt, um die relative Pose basierend auf der gemessenen Geometrie der visuellen Markierung zu ermitteln. Das vorgestellte Bildverarbeitungssystem kann nicht nur zur Regelung der Pose des MAVs verwendet werden, sondern auch genaue Posenschätzungen zur automatischen Initialisierung eines monokularen visuellen SLAM-Systems im Innen- und Außenbereich liefern.

Anschließend wird die autonome Landung eines MAVs auf einem beliebig texturierten Landeplatz während autonomer Navigation erreicht. Durch die Integration eines effizienten Objekterkennungsalgorithmus, basierend auf lokalen Bildmerkmalen in einem monokularen visuellen SLAM-System, ist das MAV in der Lage den Landeplatz autonom entlang einer vorgegebenen Strecke zu suchen, und auf ihm zu landen sobald er gefunden wurde. Die vorgestellte Lösung ermöglicht somit die autonome Navigation eines MAVs bei paralleler Landeplatzerkennung. Diese Lösung lockert die gängige Annahme in herkömmlichen Systemen zum kamerageführten Landen, dass der Landeplatz vor Beginn der Landung innerhalb des Sichtfelds des Bildverarbeitungssystems liegen muss.

Das dritte in dieser Arbeit bearbeitete Problem ist visuelles SLAM mit mehreren Kameras zur robusten Posenschätzung für MAVs. Aufgrund des begrenzten Sichtfelds von einer einzigen Kamera kann die Posenschätzung von monokularem visuellem SLAM leicht fehlschlagen, wenn sich das MAV in einer unbekannten Umgebung bewegt. Frühere Arbeiten versutchen dieses Problem hauptsächlich durch die Fusionierung von Informationen anderer Sensoren, z.B. eines Inertialsensors (Inertial Measurement Unit, IMU)

zu lösen um eine höhere Robustheit des Gesamtsystems zu erreichen, was die Robustheit des visuellen SLAM-Systems selbst nicht verbessert. Die vorliegende Arbeit untersucht Lösungen zur Verbesserung der Robustheit der Posenschätzung eines visuellen SLAM-Systems durch die Verwendung mehrerer Kameras. Wie Messungen von mehreren Kameras in die Optimierung für visuelles SLAM integriert werden können wird mathematisch analysiert. Die daraus resultierende Theorie erlaubt die Nutzung dieser Messungen sowohl zur robusten Posenschätzung als auch zur Aktualisierung der visuellen Karte. Ferner wird ein solches visuelles SLAM-System mit mehreren Kameras modifiziert, um in konstanter Laufzeit robuste visuelle Odometrie zu berechnen. Die Integration dieser visuellen Odometrie mit einem effizienten Back-End zur Erkennung von geschlossener Schleifen und der Optimierung des Posengraphen ermöglicht ein visuelles SLAM-System mit mehreren Kameras und fast konstanter Laufzeit zur autonomen Navigation von MAVs in großen Umgebungen.

Acknowledgements

First of all, I am most grateful to my supervisor, Prof. Andreas Zell, not only for his guidance and support throughout my study towards this thesis, but also for this care about my daily life, especially when I just arrived in Tübingen. He has provided me enough freedom in doing research, which helps me to focus on my research topics, and furthermore, to be able to appreciate the beautiful landscape outside my office in Sand.

I am very grateful to other co-authors of my previous publications: Sebastian Scherer, Konstantin Schauwecker, and Andreas Masselli. I have benefited greatly from collaborations with them. Furthermore, I would like to thank the colleagues who have done proofreading of individual chapters of this thesis: Yuyi Liu, Konstantin Schauwecker, Jacobo Jimenez, Nedim Šrndić and Sebastian Scherer.

I would like to thank Prof. Andreas Schilling for providing us with the access to the external pose tracking system, which plays a major role in evaluating the proposed vision systems in this thesis. I am also very grateful to him for his valuable comments to this thesis.

I am very grateful to the nice study environment that the Chair of Cognitive Systems has provided during the years of my study in Tübingen. I would like to express my gratitude to all the colleagues and former colleagues in our group. Special thanks should be given to Klaus Beyreuther and Vita Serbakova who offer assistance to our work.

I would like to thank all my friends in Tübingen who have shared the great time with me during my study, and kept me away from loneliness when I was in this foreign country being far away from home.

Finally, I would like to thank my family for their great love and support, especially my parents, my parents in law, and my wife Qian Zhang.

The China Scholarship Council (CSC) is gratefully acknowledged for providing me the scholarship for my study in Tübingen.

Contents

Notation

The list below contains the symbols and notations that are most frequently used in this thesis. In addition, vectors are bold faced lower-case characters (e.g. $\mathbf{x}$). Matrices are upper-case characters (e.g. Q).

$\mathcal{W}$	the world coordinate system
$\mathcal{B}$	the body coordinate system of a robot
$\mathcal{C}$	the camera coordinate frame
$\mathcal{C}_i$	the i^{th} camera coordinate system
$\mathcal{I}$	the image coordinate system
$\mathcal{E}$	a specific external coordinate system
$\mathcal{P}$	the mapping from coordinate system $\mathcal{W}$ to $\mathcal{I}$
$\mathcal{P}_C$	the mapping from coordinate system $\mathcal{C}$ to $\mathcal{I}$
$\mathcal{P}_{C_i}$	the mapping from coordinate system $\mathcal{C}_i$ to $\mathcal{I}$
$\mathcal{G}$	a pose graph
C_i	the i^{th} camera
p_j	the j^{th} map point in a SLAM system
$\mathcal{K}$	a keyframe
$\mathbf{K}_i$	a set of keyframes from the i^{th} camera
$\mathcal{K}_{ij}$	the j^{th} keyframe from the i^{th} camera
$\mathcal{V}_j$	a vertex in a pose graph
$\mathcal{E}_{ij}$	an edge between the i^{th} and the j^{th} vertex in a pose graph
$\mathbf{t}$	a translation vector
$\mathbf{t}_{AB}$	a translation vector interpreting the origin of coordinate system $\mathcal{B}$ expressed in $\mathcal{A}$
R_{AB}	the rotation matrix rotating a vector from coordinate system $\mathcal{B}$ to $\mathcal{A}$
T_{AB}	the transformation matrix transforming a vector from coordinate system $\mathcal{B}$ to $\mathcal{A}$
E_{AB}	the transformation from coordinate system $\mathcal{A}$ to $\mathcal{B}$, represented as a member the Lie group SE(3)
$\mathbf{g}$	the gravity vector
$\mathbf{g}_A$	the gravity vector expressed in the coordinate system $\mathcal{A}$

The list below contains the abbreviations that are most frequently used in this thesis.

MAV	Micro Aerial Vehicle
UAV	Unmanned Aerial Vehicle
SLAM	Simultaneous Localization and Mapping
PTAM	Parallel Tracking and Mapping
AR	Augmented Reality
GPS	Global Positioning System
INS	Inertial Navigation System
EKF	Extended Kalman Filter
SFM	Structure from Motion
FOV	Field of View
3D	three dimensional
DOF	Degrees of Freedom
IMU	Inertial Measurement Unit
ANN	Artificial Neural Network
RANSAC	RANdom SAmpling Consensus
RMSE	Root-Mean-Square Error
P3P	Perspective-3-Point
PnP	Perspective-n-Point
PGO	Pose-Graph Optimization
BOW	Bag of Words

Chapter 1

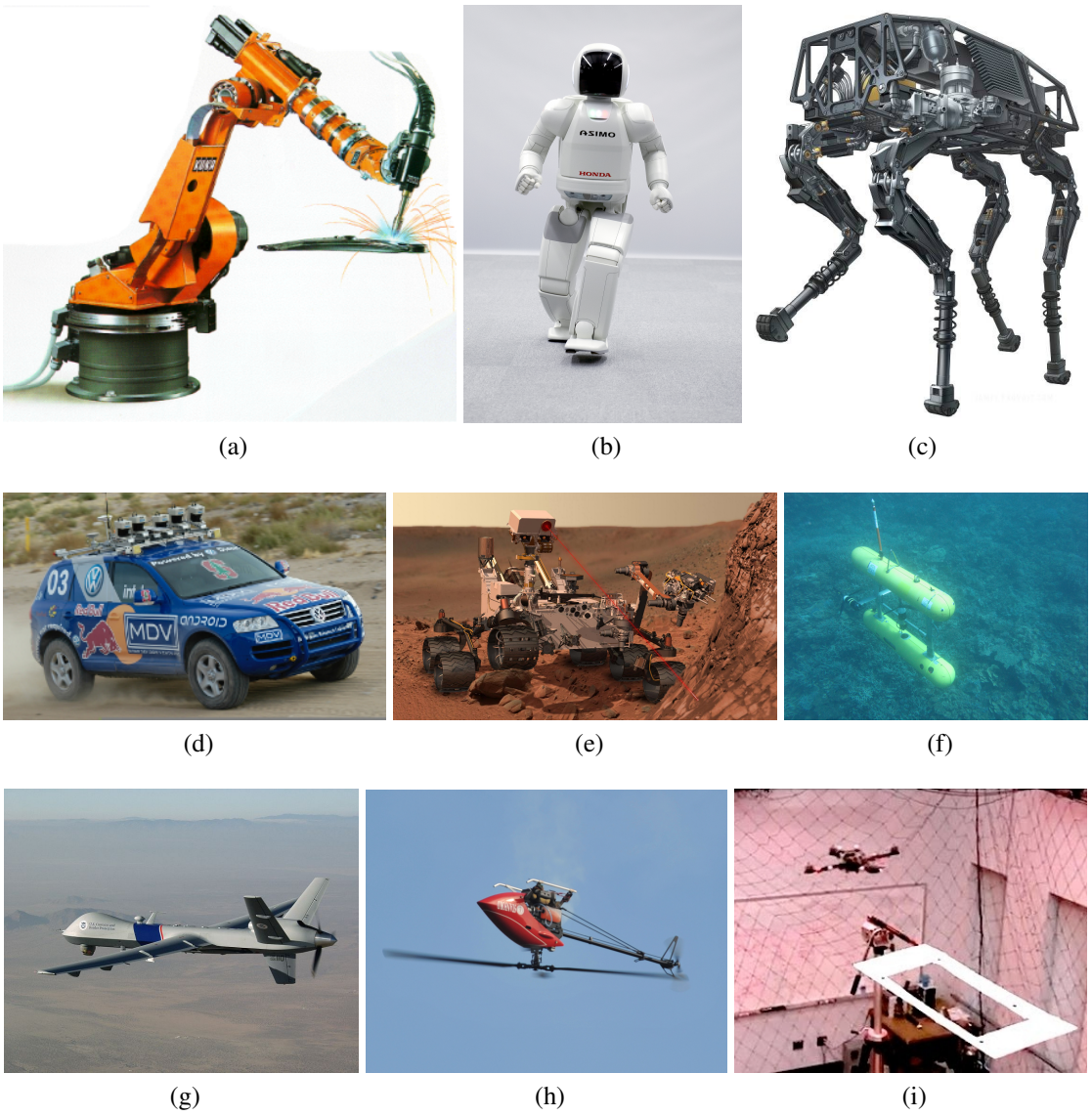
Introduction

1.1 Motivation

Robotics has been an important scientific discipline in the past decades, because of the significance of developing robots to extend the physical capability and the potential of human beings. Research on mobile robots is an especially attractive field, which seeks solutions for operating robots with mobility in unstructured and unknown environments. Unlike industrial robots, i.e. robot arms/manipulators with varying degrees of freedom, which are normally mounted on a fixed working station for industrial manufacturing, e.g. as shown in Fig. 1.1a, mobile robots are potentially able to navigate in various environments for a wide range of applications, such as service, surveillance, rescue, and transportation. Recently, more focus has been on developing autonomous mobile robots for those applications.

Many exciting developments have been seen in the field of autonomous mobile robots (Siegwart *et al.*, 2011). To date, mobile robots of different types designed for various purposes have been developed. Some state-of-the-art robots, ranging from ground robots, to marine robots and aerial robots, are shown in Fig. 1.1b–1.1i: The humanoid robot ASIMO (Honda, 2014), designed to assist humans in their daily tasks, can perform complex movements and interactions. BigDog (Raibert *et al.*, 2008), a four-legged robot, is designed to traverse rough terrains and carry heavy loads. The autonomous vehicle Stanley (Thrun *et al.*, 2006), which won the 2005 DARPA Grand Challenge, is developed for high-speed desert driving without manual intervention. In outer space, the Curiosity Mars rover (NASA, 2014) is exploring the Red Planet and has provided key information about this planet. The autonomous underwater vehicle (AUV) Sirius (ACFR, 2014) is capable of undertaking high-resolution survey works in ocean. The MQ-9 Reaper is an unmanned aerial vehicle (UAV) designed for military use with long-endurance, persistent surveillance/strike capability (GA-ASI, 2014). The autonomous helicopter at Stanford AI Lab is able to perform complicated aerobatics (Abbeel *et al.*, 2010). Fig. 1.1i shows a micro aerial vehicle (MAV) developed in GRASP Lab. This quadrotor can perform aggressive maneuvers when its accurate pose estimates are provided by an external pose tracking system (Mellinger *et al.*, 2012).

In the last decade, we have seen a growing interest in MAVs from the robotics com-



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 1.1: Some state-of-the-art robotic platforms. (a) An industrial robot from KUKA (plant-spot welding robot). ©KUKA Roboter GmbH. (b) The humanoid robot ASIMO. ©American Honda Motor Co. Inc. (c) The BigDog robot. Reprint from Wooden *et al.* (2010). (d) The autonomous vehicle Stanley described in Thrun *et al.* (2006). (e) An artist's concept depicting the Curiosity Mars rover. ©NASA/JPL-Caltech. (f) The Sirius AUV in an ocean-survey mission. ©Australian Centre for Field Robotics. (g) The MQ-9 Reaper UAV. Reprinted from Wikipedia (2014a). (h) An autonomous helicopter from SAIL performing aerobatics under computer control. ©Andrew Ng. (i) A quadrotor performing aggressive maneuvers through a narrow gap. ©GRASP Lab.

munity. One of the reasons for this trend is that MAVs are potentially able to efficiently navigate in complex 3D environments with different types of terrains, which might be inaccessible to ground vehicles or large-scale UAVs, e.g. in an earthquake-damaged building (Michael *et al.*, 2012). A basic requirement for MAVs to autonomously operate in such environments is their robust pose tracking, which is still a challenging task when the environment is previously unknown. Meanwhile, if a map of the environment can be built, it will be able to provide support to path planning of autonomous navigation of the MAV (Schauwecker and Zell, 2014). Recently, an interesting research focus has been on using onboard visual solutions to address these issues, especially using visual simultaneous localization and mapping (SLAM) systems.

The SLAM problem is one of the most fundamental problems in robotics. It asks if it is possible for a mobile robot to be placed at an unknown location in an unknown environment, and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map (Durrant-Whyte and Bailey, 2006). There are three basic questions for autonomous navigation of a mobile robot (Leonard and Durrant-Whyte, 1991): *"where am I?"*, *"where am I going?"* and *"how should I get there?"* The first question is about the localization of the robot. The second and the third questions are about specifying a goal and being able to plan a path which enables the robot to achieve this goal. A SLAM system will answer the first question, and provide the basic knowledge to a robot in order to autonomously navigate in the environment: the location of the robot, and the understanding of the environment in the form of a map. Then it is possible to find answers to the later two questions.

To achieve SLAM, different sensor modalities can be used, like laser scanners and cameras. Laser scanning systems are generally active and accurate, but with some drawbacks in MAV applications. 2D laser scanners have been well studied for autonomous navigation of MAVs in both structured and unstructured indoor environments (Grzonka *et al.*, 2009; Shen *et al.*, 2011; Bry *et al.*, 2012). However, solutions using 2D laser scanners are difficult to be extended to work in complex 3D environments and in object recognition tasks. 3D laser scanners, on the other hand, are usually heavy and slow, see Xiao *et al.* (2013), thus not suitable for onboard use of MAVs. Compared with other sensors, like laser scanners, cameras are passive, and have a superior potential for environment perception, while still being lightweight, relatively low cost and energy efficient. These advantages make vision systems quite attractive for the research on autonomous navigation of MAVs which in general have very limited payload. Thus, many MAVs have been relying on visual solutions for autonomous flight tasks, especially in GPS-denied environments, e.g. in indoor or in urban-outdoor environments. In this thesis, visual SLAM, or SLAM using cameras, for autonomous navigation of MAVs, is our main concern.

Although we have seen some successful applications of visual SLAM on ground vehicles (Cummins and Newman, 2008; Strasdat *et al.*, 2011), there are more challenges in using visual SLAM to enable autonomous navigation of MAVs. First, the payload of an MAV is rather limited. This usually leads to *limited onboard computational capability*,

which requires the visual SLAM system to be very efficient, especially when other vision tasks, e.g. object recognition, are required to be done simultaneously. Second, the 3D maneuverability of an MAV makes the *robustness of pose tracking* to be highly desired. When pose tracking fails, unlike a ground vehicle, which may be able to maintain stability simply by ceasing moving, an MAV is likely to run into catastrophic situations with its control getting lost and even to crash into obstacles. This thesis is dedicated to a systematic study of visual SLAM solutions to autonomous navigation of MAVs under these challenges. More specifically, we investigate monocular visual solutions to pose estimation of MAVs, monocular visual SLAM for autonomous landing of MAVs, as well as extending monocular visual SLAM to multi-camera visual SLAM for robust pose tracking and environmental mapping.

The first specific problem addressed in this thesis is visual pose estimation of MAVs based on an artificial landmark. Most of the existing work solves this problem by relying on infrared markers with known geometric configurations (Wenzel *et al.*, 2010a; Faessler *et al.*, 2014), or visual markers with special combinations of some basic geometric components (Merz *et al.*, 2006; Meier *et al.*, 2011). The advantage of such marker-based methods is that their computational costs are normally rather low. However, those methods are only designed for applications in certain restricted environments, e.g. in indoor environments or in environments without cluttered background for the special markers. *Can we develop a visual solution which is able to provide robust pose estimation in cluttered environments, based on some simple and regular visual marker?* This is one of the motivations for our work in Chapter 3. Another major motivation for that work directly comes from the requirement of our visual SLAM system: A monocular system is generally lacking of metric-scale information, and needs to be initialized either by a metric sensor or by a cooperative object. A robust visual solution for retrieving the metric scale will facilitate our visual SLAM system to be initialized in complex environments.

The second problem addressed in this thesis is the autonomous landing of MAVs. Autonomous landing is a basic but also challenging phase for autonomous navigation of MAVs. When the exact position of a desired landing site is unknown, an MAV should be able to search for and locate it autonomously, and then land on it to finish the autonomous flight. The existing work normally assumes that the landing site is already located inside the field of view (FOV) of the vision system onboard an MAV/UAV (Sharp *et al.*, 2001; Saripalli and Sukhatme, 2007), or the landing site searching task is enabled by global pose estimates from a GPS sensor (Cesetti *et al.*, 2010). One reason resulting in this state of research is that both pose estimation in unknown environments and object (landing site) recognition are computationally intensive tasks, and therefore, difficult to be processed in parallel on an MAV with constrained computational power. Then an open question remains: *How to search for a designated landing site in an unknown environment, and land on it, during vision-based autonomous navigation of an MAV?* Our work presented in Chapter 4 is going to answer this question.

In order to achieve more robust pose tracking of visual SLAM for MAVs, previous work has made much effort in sensor fusion, e.g. fusing inertial measurements for vi-

sual SLAM (Shen *et al.*, 2013b; Weiss *et al.*, 2013). However, *how to improve the pose tracking robustness of the visual SLAM system itself* has not been equally emphasized. Pose tracking of a monocular visual SLAM system may easily fail in complex environments due to poor visual features which can be observed by the camera. This also applies to stereo visual SLAM, which employs cameras looking in one specific direction with limited FOVs. Actually, a larger effective FOV of a vision system can be obtained by integrating multiple cameras in it. This implies that better pose tracking robustness can be achieved by extending the monocular visual SLAM to utilize measurements from multiple cameras. This motivates our work in Chapter 5, which presents a multi-camera visual SLAM system. In such a visual SLAM system, multiple cameras can be mounted pointing to different directions, so that more reliable visual features can be observed to fulfil the pose tracking and map updating tasks.

Employing the work in Chapter 5, we address the problem of improving the efficiency of visual SLAM for MAVs. A multi-camera visual SLAM system provides us with more robustness in pose tracking. However, a multi-camera system can also be more time-consuming than a monocular system, since it is required to process images from multiple cameras at each time instance. To apply multi-camera visual SLAM systems in autonomous navigation of MAVs, it is natural for us to ask: *How to achieve constant-time operations for large-scale explorations of MAVs?* This motivates our work in Chapter 6, which modifies our multi-camera visual SLAM system developed in Chapter 5 to be a constant-time visual odometry, and presents an efficient back-end to correct the pose drift resulting from the visual odometry.

1.2 Contributions

The research described in this thesis mainly focuses on developing robust and efficient onboard visual SLAM systems for autonomous navigation MAVs. The specific contributions made in each of the four technical chapters are the following:

Chapter 3:

- A real-time artificial-landmark recognition method is presented, which is robust to cluttered environment.
- To compute the six degrees-of-freedom (6DOF) pose estimates of MAVs, a computational projective-geometry solution is presented, which is based on the retrieved geometry information of the artificial landmark and attitude estimates from the onboard IMU.
- Low-altitude autonomous takeoff, hovering, and landing of an MAV is enabled by using the proposed monocular-vision system to provide pose estimates of the MAV.

Large parts of this work have been pre-published in the following papers:

1. Yang, S., Scherer, S. A., and Zell, A. (2012). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. In 2012 International Conference on Unmanned Aircraft Systems (ICUAS'12), Philadelphia, PA, USA.

2. Yang, S., Scherer, S. A., and Zell, A. (2013b). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. Journal of Intelligent & Robotic Systems, 69, 499–515.

The following co-authored paper is partially based on this work:

3. Masselli, A., Yang, S., Wenzel, K., and Zell, A. (2014). A cross-platform comparison of visual marker based approaches for autonomous flight of quadrocopters. Journal of Intelligent & Robotic Systems, 73(1-4), 349–359.

Chapter 4:

- An efficient monocular visual SLAM system is implemented for autonomous navigation of MAVs, which provides accurate pose estimates of the MAV and an environment map.

- To efficiently detect an arbitrarily textured landing site for an MAV, a local-feature-based object detection algorithm is presented, which is integrated into the SLAM system and makes use of the feature corners detected on the multi-scale images by the SLAM system.

- The absolute pose of the detected landing site is estimated by using a novel method which utilizes those map points of the SLAM system associated with the landing site. This method does not require a known size of the landing site.

- The presented vision system enables an MAV searching for the landing site while autonomously navigating in an unknown environment. Autonomous landing on a designated landing site with a previously unknown location is also achieved.

Large parts of this work have been pre-published in the following papers:

4. Yang, S., Scherer, S. A., Schauwecker, K., and Zell, A. (2013a). Onboard monocular vision for landing of an MAV on a landing site specified by a single reference image. In 2013 International Conference on Unmanned Aircraft Systems (ICUAS'13), pages 317–324, Atlanta, GA, USA.

5. Yang, S., Scherer, S. A., Schauwecker, K., and Zell, A. (2014a). Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision. Journal of Intelligent & Robotic Systems, 74(1-2), 27–43.

Chapter 5:

- To integrate measurements from multiple cameras into a single SLAM system, a mathematical analysis of the optimization problems in the SLAM system is provided.

- An efficient multi-camera visual SLAM system is developed, which utilizes feature points detected in the images from multiple cameras for pose tracking and map updating. It can provide robust pose estimates of MAVs in real-time in complex environments, in which a monocular visual SLAM system may easily fail.

- Autonomous navigation of an MAV in a complex environment is achieved by using the proposed multi-camera visual SLAM system for the pose estimation.

Large parts of this work have been pre-published in the following paper:

6. Yang, S., Scherer, S. A., and Zell, A. (2014c). Visual SLAM for autonomous MAVs with dual cameras. In 2014 International Conference on Robotics and Automation (ICRA'14), pages 5227–5232, Hong Kong, China.

Chapter 6:

- A robust visual SLAM system consisting of a constant-time visual odometry and an efficient back-end for loop closure detection and pose-graph optimization is developed. The multi-camera visual SLAM system presented in Chapter 5 is modified to be the constant-time visual odometry.

- An adaptive-window pose-graph optimization algorithm is proposed for efficient global relaxation of the SLAM system. Constant-time pose-graph optimization within a small window defined by a uniform-cost search is processed during regular exploration of the SLAM system, utilizing pose constraints from the bundle adjustment of the visual odometry. When a loop closure is detected, pose-graph optimization is done within a large window, which can efficiently correct long-term pose drift of the visual odometry.

- Autonomous navigation of an MAV in an unknown environment is achieved using the proposed SLAM system for the pose estimation.

Large parts of this work have been pre-published in the following paper:

7. Yang, S., Scherer, S. A., and Zell, A. (2014b). Robust onboard visual SLAM for autonomous MAVs. In 2014 International Conference on Intelligent Autonomous Systems (IAS-13), Padova, Italy. Accepted.

The thesis is structured in the following way: In this chapter the motivations and contributions of this thesis have been introduced at a high level. In Chapter 2, the background related to MAV control and visual SLAM is presented. The following four technical chapters report the main work of this thesis which has been enumerated above. The specific related work is discussed in individual technical chapters which address the corresponding problems. Finally, Chapter 7 concludes the thesis by summarizing the main points and drawing outlook for future research.

Chapter 2

Background

This chapter presents basic principles and related work on MAV control and visual SLAM. The quadrotor-MAV platform used for experiments throughout this thesis is introduced in Sec. 2.1. Furthermore, we present quadrotor control algorithms in Sec. 2.2. In Sec. 2.3, the camera model and its calibrations are explained, which are the foundations to our vision systems. Finally, in Sec. 2.4, we briefly introduce the SLAM problem, the PTAM system (Klein and Murray, 2007) which forms the basis for the visual SLAM systems developed in this thesis, and provide an overview of previous work on visual SLAM and its applications in autonomous navigation of UAVs/MAVs.

2.1 The MAV Platform

In this thesis, we choose an open source quadrotor developed in the Pixhawk project at ETH Zürich to be our MAV platform. The design of this quadrotor is described by Meier *et al.* (2011). A quadrotor provides us with advantages like being simple in mechanics, and having a moderate size and payload for indoor or low-altitude outdoor applications.

As shown in Fig. 2.1, our quadrotor is equipped with four motors and 10-inch propellers, which enable it to lift approximately 400 g payload at a total system weight of about 1.2 kg, including battery. Its onboard computer is a Kontron microETXexpress computer-on-module (COM) featuring an Intel Core 2 DUO 1.86 GHz CPU, 2 GB DDR3 RAM and a 32Gb SSD. The pxIMU inertial measurement unit and autopilot board that we use mainly consists of a micro computer unit (MCU) and inertial sensors, including a tri-axes accelerometer and a tri-axes gyroscope. The MCU is a 60MHz ARM7 microcontroller, which is used for sensor-data readout and fusion, as well as position and attitude control of the MAV. The accelerometer and gyroscope provide three-dimensional (3D) linear acceleration (± 6 g) and 3D angular velocity (± 500 deg/s) measurements of the MAV.

Each camera used in our vision systems in this thesis is a PointGrey Firefly MV monochrome camera of only 37 g weight. It has an image resolution of 640×480 pixels, a maximum frame rate of 60 fps, and is equipped with a lens featuring an approximately 90° viewing angle. In the cases of a monocular vision system as in Chapter 3 and Chapter 4, one such camera is mounted on the MAV in a downward-facing pose. In the later

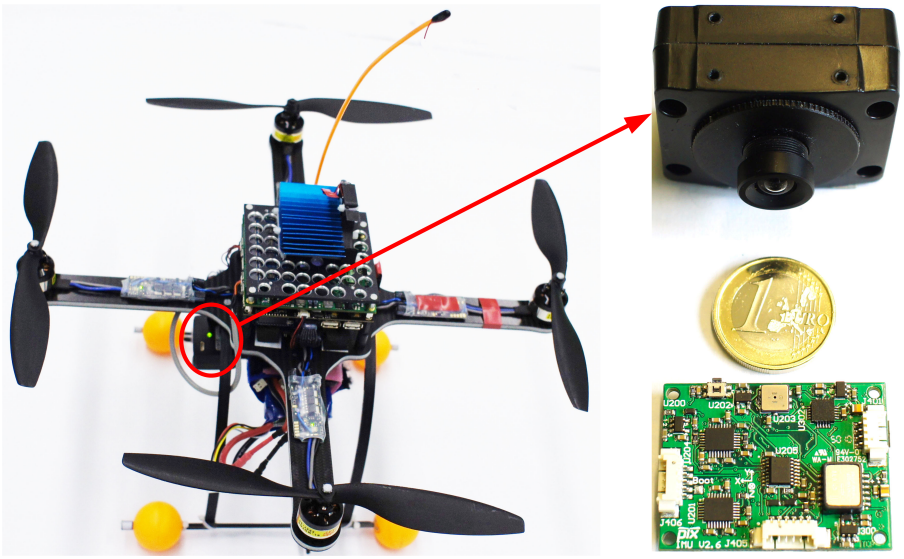


Figure 2.1: The quadrotor MAV (*left*) which is used as the research platform in our experiments in this thesis, and a camera (*top right*) and the IMU (*bottom right*) which are used on the MAV, with their sizes compared with that of a coin.

two chapters, which describe the implementations of multi-camera visual SLAM systems, two cameras with non-overlapping fields of view are used: one facing downward, and the other one facing forward. More details will be provided in the individual technical chapters.

The original quadrotor mechanical frame of the Pixhawk project is based on carbon-fiber material. Throughout the four technical chapters, the quadrotor frame may be slightly changed: We may also use a frame from MikroKopter (HiSystems, 2014), which is made of aluminum material and is more durable. However, the effect of such a change to the quadrotor control is ignored in this thesis. Thus, we do not distinguish the MAV platform used in each chapter. Since there is no easy way to locate the exact center of mass of our quadrotor, we assume it to coincide with the geometric center of the quadrotor frame. We expect the controllers of our quadrotor to be robust to the errors caused by this assumption.

In order to use the attitude estimates from the IMU for the pose controllers of our quadrotor and for our vision systems, the relation between the IMU coordinate system ($\mathcal{U}$) and the quadrotor body coordinate system ($\mathcal{B}$) needs to be calibrated. As the IMU does not directly provide position estimates, we assume the origin of the IMU frame to coincide with the MAV body frame, as illustrated in Fig. 2.2. Thus, we only need to find the rotation matrix R_{BU} between these two coordinate systems. We expect that, if the IMU was perfectly mounted, $\mathcal{U}$ would exactly coincide with $\mathcal{B}$, i.e. $R_{BU} = I$. However, this is usually not the case in practice. The actual rotation between the IMU frame and the

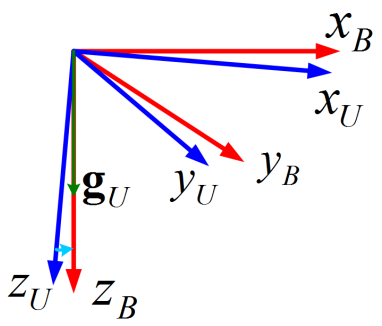


Figure 2.2: The IMU frame and the quadrotor body frame.

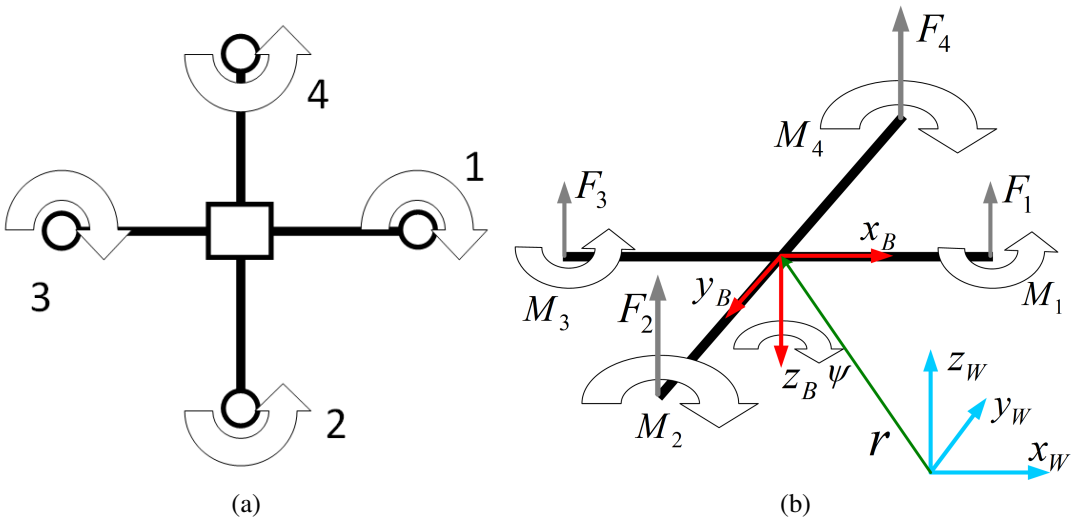
body frame can be estimated by measuring the normalized gravity vector $\mathbf{g}_U$ (expressed in $\mathcal{U}$) using the accelerometer data when the quadrotor is placed on a horizontal plane. In this case, the gravity vector should be parallel to the z_B axis, i.e. in $\mathcal{B}$, it can be expressed as $\mathbf{g}_B = [0,\ 0,\ 1]^T$. Therefore, the rotation matrix R_{BU} can be found as the shortest rotation which satisfies $\mathbf{g}_B = R_{BU}\mathbf{g}_U$. Since yaw angle estimates from the IMU will not be used in our system, we ignore the errors in the alignment along x and y direction.

2.2 MAV Flight Control

This section introduces the controllers of our MAV. Since this thesis focuses on the vision-system aspect of MAVs, we generally implement the controllers based on the previous work in the literature, mainly the work presented in Michael *et al.* (2010) and a similar implementation from the Pixhawk project.

For robust flight control of a quadrotor, the work in Michael *et al.* (2010) presents a nested controller, which consists of an attitude controller and a position controller. A hover controller and a 3D trajectory controller have been developed for position control in different stages of a flight. In Mellinger *et al.* (2010), robust perching and landing a quadrotor on a landing pad based on this nested controller was demonstrated. Further in Mellinger *et al.* (2012), precise aggressive maneuvers of quadrotors are achieved using the previously proposed controller and a 3D trajectory generation algorithm. Some very impressive aggressive maneuvers of an MAV are performed in this work, such as flying through narrow, vertical gaps and perching on inverted surfaces. However, we should note that the above work achieving autonomous flight is based on accurate 6DOF pose estimates from an external Vicon tracking system (Vicon, 2014).

For flight control of our quadrotor, we adapt the quadrotor dynamic model and the nested controller presented in Michael *et al.* (2010) in this thesis. We will briefly describe them in the following sections. Furthermore, we modify the original method implemented in the pxIMU to comply with this controller.



(a) (b)

Figure 2.3: (a) A top view of rotation directions of the four rotors on a quadrotor, producing moments acting on the quadrotor frame in counter directions. (b) Forces and moments acting on the quadrotor frame, with a depiction of the principle of yaw control to clockwise rotation.

2.2.1 Quadrotor configuration and dynamic model

A common configuration of quadrotors is illustrated in Fig. 2.3a, with four fixed-pitch propellers configured in a symmetric cross, which also applies to our quadrotor. The quadrotor body frame $\mathcal{B}$ is attached to the center of mass of the quadrotor, with x_B coinciding with the preferred forward direction along one arm of the quadrotor frame and z_B pointing to the downward direction perpendicular to the quadrotor frame, as depicted in Fig. 2.3b. The rotors 1 and 3 rotate in the z_B direction, while 2 and 4 rotate in the $-z_B$ direction. In addition to a force in the $-z_B$ direction, each rotor produces a moment perpendicular to the rotation plane of its propeller, i.e. the plane of the quadrotor frame. Since the moments acting on the quadrotor are opposite to the rotation directions of the respective propellers, driving the two pairs of propellers in opposite directions removes the need for a tail rotor (Bouabdallah, 2007). By adjusting the rotation speed of each rotor, corresponding forces and moments acting to the quadrotor frame can be generated. Consequently, the attitude and position of the quadrotor can be controlled, as described in Nonami *et al.* (2010).

Fig. 2.3b illustrates forces and moments produced by the rotation of the four rotors in the directions shown in Fig. 2.3a. In this example, the control of the yaw rotation in clockwise direction is depicted. We use $Z-X-Y$ Euler angles (Schilling, 1990) to express the rotation of the quadrotor in the world frame $\mathcal{W}$, which is defined with z_W pointing upward. When considering to rotate $\mathcal{W}$ to the quadrotor body frame $\mathcal{B}$, we first

rotate around z_W by the yaw angle ψ, then rotate around the intermediate $x-$axis by the roll angle ϕ, and finally rotate around the y_B axis by the pitch angle θ. Following the notations in Michael *et al.* (2010), the resulting rotation matrix for transforming a vector from $\mathcal{B}$ to $\mathcal{W}$ is in the form of

$$R_{WB} = \begin{bmatrix} c_\psi c_\theta - s_\phi s_\psi s_\theta & -c_\phi s_\psi & c_\psi s_\theta + c_\theta s_\phi s_\psi \\ c_\theta s_\psi + c_\psi s_\phi s_\theta & c_\phi c_\psi & s_\psi s_\theta - c_\psi c_\theta s_\phi \\ -c_\phi s_\theta & s_\phi & c_\phi c_\theta \end{bmatrix},$$

where c_θ and s_θ denote $\cos(\theta)$ and $\sin(\theta)$, respectively, and similarly for ϕ and ψ. The forces acting on the quadrotor are the gravity in the $-z_W$ direction, and the forces F_i generated by each of the four rotors in the $-z_B$ direction. Let $\mathbf{r}$ denotes the position vector of the center of mass of the quadrotor in the world frame $\mathcal{W}$, the equation governing the acceleration of the quadrotor is

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R_{WB} \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_i \end{bmatrix}. \tag{2.1}$$

The angular velocities around x_B, y_B and z_B are p, q and r, respectively. They can be calculated by transforming the derivatives of the roll, pitch and yaw angles to the quadrotor body frame (Murray *et al.*, 1994) according to

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c_\theta & 0 & -c_\phi s_\theta \\ 0 & 1 & s_\phi \\ s_\theta & 0 & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{2.2}$$

Among the moments M_i produced by each of the four motors, M_1 and M_3 act in the $-z_B$ direction while M_2 and M_4 act in the z_B direction. Let L be the distance from the axis of rotation of each rotor to the center of the quadrotor, and I be the moment of inertia matrix referenced to the center of mass along the $x_B - y_B - z_B$ axes, the angular acceleration determined by the Newton-Euler equations (Murray *et al.*, 1994) is

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_1 - F_3) \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.3}$$

2.2.2 Attitude control and hover control

The nested controller proposed in Michael *et al.* (2010) is illustrated in Fig. 2.4. We briefly introduce the attitude controller and the hover controller here, which are used for the position control of our quadrotor. These controllers rely on small angle assumptions, i.e. assuming that the quadrotor attitude is very close to a level attitude.

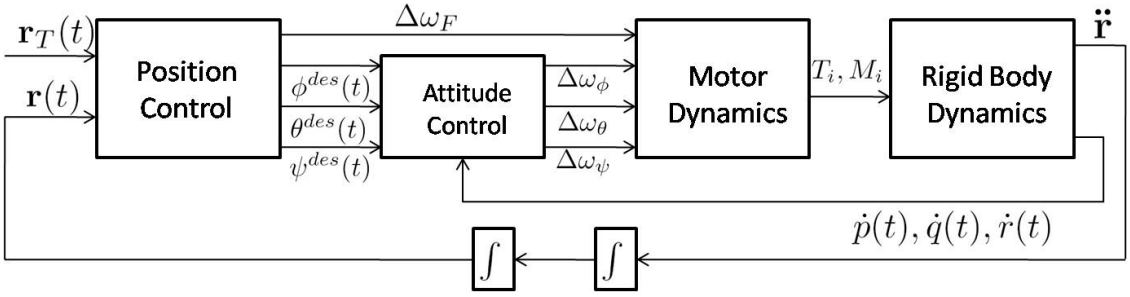


Figure 2.4: The nested controller for position and attitude control in Michael *et al.* (2010).

Following the notations in Michael *et al.* (2010), the desired four rotor speeds of the quadrotor are set to be

$$\begin{bmatrix} \omega_1^{des} \\ \omega_2^{des} \\ \omega_3^{des} \\ \omega_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_h + \Delta\omega_F \\ \Delta\omega_\phi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix}, \tag{2.4}$$

where ω_h is the nominal rotor speed which keeps a steady hovering state, and $\Delta\omega_F$, $\Delta\omega_\phi$, $\Delta\omega_\theta$, and $\Delta\omega_\psi$ are the deviations from the nominal vector $[\omega_h,\ 0,\ 0,\ 0]^T$. $\Delta\omega_F$ results in a net force along the z_B axis, while $\Delta\omega_\phi$, $\Delta\omega_\theta$, and $\Delta\omega_\psi$ produce moments which cause roll, pitch and yaw, respectively. Then a proportional-derivative (PD) controller can be used to control the attitude in the form of

$$\begin{cases} \Delta\omega_\phi = k_{p,\phi}\left(\phi^{des} - \phi\right) + k_{d,\phi}\left(p^{des} - p\right), \\ \Delta\omega_\theta = k_{p,\theta}\left(\theta^{des} - \theta\right) + k_{d,\theta}\left(q^{des} - q\right), \\ \Delta\omega_\psi = k_{p,\psi}\left(\psi^{des} - \psi\right) + k_{d,\psi}\left(r^{des} - r\right). \end{cases} \tag{2.5}$$

Substituting Eq. (2.5) along with a chosen $\Delta\omega_F$ into Eq. (2.4) yields the desired rotor speeds.

For the 3D position control, the roll and pitch angles are used to control the position of the quadrotor in the $x_W - y_W$ plane, while $\Delta\omega_F$ to control the position along z_W and $\Delta\omega_\psi$ to control the yaw angle. The hover controller is used by the quadrotor to reach a desired position and yaw angle with zero linear and angular velocities. Let $\mathbf{r}_T$ and ψ_T be the trajectory and yaw angle to be tracked, the desired accelerations $\ddot{\mathbf{r}}_i^{des}$ at time step i can be calculated from a proportional-integral-derivative (PID) feedback of the position error $e_i = (\mathbf{r}_{T,i} - \mathbf{r}_i)$ as

$$\ddot{\mathbf{r}}_i^{des} = k_{p,i} e_i + k_{i,i} \int e_i \, \mathrm{d}t + k_{d,i} \dot{e}_i. \tag{2.6}$$

This PID controller is then used for the hover control. By linearizing Eq. (2.1), we get

the simplified relationship between the desired accelerations and the roll and pitch angles as follows,

$$\begin{cases} \ddot{\mathbf{r}}_1^{des} = g(\theta^{des}\cos(\psi_T) + \phi^{des}\sin(\psi_T)), \\ \ddot{\mathbf{r}}_2^{des} = g(\theta^{des}\sin(\psi_T) - \phi^{des}\cos(\psi_T)), \\ \ddot{\mathbf{r}}_3^{des} = \dfrac{8k_F\omega_h}{m}\Delta\omega_F, \end{cases}$$

where k_F is the motor factor so that the vertical force produced be the motor with an angular speed ω is $F = k_F\omega^2$. Then the desired roll and pitch angles for the attitude controller and ω_F can be computed from the desired accelerations as

$$\begin{cases} \phi^{des} = \dfrac{1}{g}(\ddot{\mathbf{r}}_1^{des}\sin(\psi_T) - \ddot{\mathbf{r}}_2^{des}\cos(\psi_T)), \\ \theta^{des} = \dfrac{1}{g}(\ddot{\mathbf{r}}_1^{des}\cos(\psi_T) + \ddot{\mathbf{r}}_2^{des}\sin(\psi_T)), \\ \Delta\omega_F = \dfrac{m}{8k_F\omega_h}\ddot{\mathbf{r}}_3^{des}. \end{cases} \tag{2.7}$$

From Eq. (2.4), Eq. (2.5), and Eq. (2.7), the desired rotor speeds for the hover control of the quadrotor can be calculated.

In this thesis, the 3D position estimates from our onboard vision systems are used as feedback to the position controller after fusing the IMU data with a basic Kalman Filter (Kalman, 1960). Since the IMU can provide roll and pitch estimates with a frequency of 200 Hz, which is much higher than those of the onboard vision systems, we use the estimates provided by the IMU for attitude control. Meanwhile, the yaw estimates provided by the onboard vision system are used for the yaw control.

2.2.3 Trajectory control

Two trajectory control methods have been implemented for our MAV: the setpoint method and the trajectory-following method. A comparison of these two methods will be presented in the experiments in Chapter 4.

Setpoint method

A simple trajectory control method can be achieved by setting a list of 3D points along the predefined flight-path, which will be reached by the quadrotor one after another. In this case, we assume that the quadrotor flies in a hovering state, or in a state trying to reach a hovering state. Then the hover controller described in Sec. 2.2.2 can be used for the position control in order to reach those setpoints. Thus, the quadrotor can be controlled to follow the path. We assume that the MAV has reached a setpoint, if its distance to this point is smaller than a threshold d_s and the yaw angle difference is smaller

than ψ_s, for a period of time t_s, after which we advance the setpoint to the next one on the predefined path.

Trajectory-following method

To allow the quadrotor to follow a predefined flight path more efficiently, we implemented a trajectory-following method based on the path following controller presented in Michael *et al.* (2010). The general idea of this controller is to only consider position errors in the normal direction of the predefined path, while ignoring errors in the tangent direction. Thus, a constant expected forward speed can be achieved. This approach is similar to the one presented in Hoffmann *et al.* (2008), but is extended from 2D to 3D trajectories. The final commanded acceleration is calculated by a PD feedback of the position and velocity errors, together with the consideration of a feed-forward term of the desired acceleration. We simplify this method by dropping the feed-forward term, and set a constant forward speed, v_s, along the tangent direction of the predefined path. The reason for this simplification is that we do not expect our MAV to perform aggressive maneuvers, and thus, a constant expected speed is sufficient for the autonomous navigation tasks of the MAV.

2.3 Camera Model and Calibration

The vision algorithms in this thesis are designed for perspective cameras (Hartley and Zisserman, 2004). The perspective camera model and calibrations are briefly introduced in the following sections.

2.3.1 Perspective camera model

This section introduces the mathematical model of a perspective camera which projects a point in 3D space onto the image plane. The complete chain of transformations for such projections is presented following the frameworks in Bleser (2009) and Zhang (2013).

Normalized pinhole camera model

As illustrated in Fig. 2.5, we can associate a pinhole camera with two different 2D image planes (Forsyth and Ponce, 2002): a normalized image plane, and the physical retina of the camera which will be introduced together with intrinsic camera parameters later.

Under the pinhole camera model, the camera is represented by a 2D image plane and a 3D point called the optical centre or camera centre. The distance from the optical centre to the image plane is called the focal length of the camera, denoted by f, which is specified in terms of metric units. The line from the optical centre perpendicular to the image plane is called the optical axis or principal ray of the camera, and the point where the

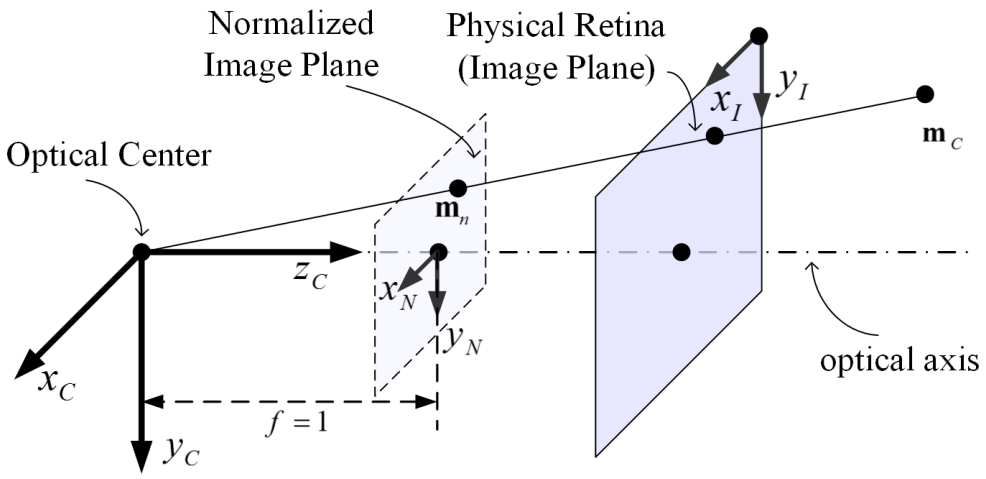


Figure 2.5: Pinhole projection of a 3D point onto the normalized image plane and the physical retina.

optical axis meets the image plane is called the principle point (Hartley and Zisserman, 2004). The camera coordinate system $\mathcal{C}$ is defined such that its origin coincides with the optical centre, and its z axis coincides with the optical axis. A point in 3D space is projected onto the image plane by drawing a ray from the 3D point to the optical centre. Here, under the normalized pinhole model with $f = 1$, the normalized image plane coincides with the normalized image coordinate system $\mathcal{N}$. The projection of a 3D point $\mathbf{m}_c = [x_c, y_c, z_c]^T$ in the camera frame C to a 2D point $\mathbf{m}_n = [x_n, y_n]^T$ in the normalized image frame $\mathcal{N}$ is

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} x_c \\ y_c \end{bmatrix}. \tag{2.8}$$

By introducing homogeneous coordinates, Eq. (2.8) can be written as

$$\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} \propto z_c \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}. \tag{2.9}$$

Radial and tangential distortion

The normalized pinhole model does not consider lens distortions of cameras. However, real cameras, especially low cost ones or those with wide angle lenses, usually exhibit significant lens distortion. In this thesis, the distortion model presented in Heikkila and Silven (1997) is introduced. It accounts for both radial and tangential distortion, and the distorted image coordinates $\mathbf{m}_d = [x_d, y_d]^T$ are expressed as a function of the normalized

image coordinates $\mathbf{m}_n = [x_n, y_n]^T$ as

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \underbrace{(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \begin{bmatrix} x_n \\ y_n \end{bmatrix}}_{\textit{radial distortion}} + \underbrace{\begin{bmatrix} 2\kappa_4 x_n y_n + \kappa_5 \left(r^2 + 2x_n^2\right) \\ \kappa_4 \left(r^2 + 2y_n^2\right) + 2\kappa_5 x_n y_n \end{bmatrix}}_{\textit{tangential distortion}}. \tag{2.10}$$

where $r = \sqrt{x_n^2 + y_n^2}$, and $\kappa_i, i = 1, ..., 5$ are distortion coefficients which are the parameters of the distortion model. More elaborated distortion models can also be found in Brown (1971); Faig (1975); Slama *et al.* (1980); Weng *et al.* (1992). We should note that the distortion function is likely to be dominated by the radial components according to Zhang (1999). In this work, only the first two terms of radial distortion are considered.

We can use Eq. (2.10) to retrieve an undistorted image in the following way. First, we sample in the undistorted image domain to form a set of undistorted coordinates $\mathbf{m}_n^{ij}$, $i \in [1, height]$, $j \in [1, width]$, whose values describe the image with a dimension of $height \times width$ that we want to retrieve. Then we use Eq. (2.10) to compute the corresponding distorted image coordinates $\mathbf{m}_d^{ij}$. Finally, the image value of $\mathbf{m}_d^{ij}$ is interpolated in the distorted image domain, which is the retrieved value for $\mathbf{m}_n^{ij}$. In order to undistort single coordinates, the inverse function of (2.10) is required. An analytical solution does not generally exist in this case. However, we can use Newton's method to iteratively calculate the inverse mapping of (2.10) with the distorted coordinates as an initial guess.

Intrinsic camera parameters

In order to model the projection of a 3D point onto the physical retina of a camera (the image plane as we normally call it) in the pixel coordinate system, as illustrated in Fig. 2.5, the intrinsic camera parameters need to be considered. In the rest of this thesis, we call the pixel coordinate system the image coordinate system $\mathcal{I}$.

The physical retina of a camera is located at distance $f \neq 1$ from the optical center. It may have non-square pixels with dimensions of $\frac{1}{k} \times \frac{1}{l}$, where k and l are expressed in pixel per metric unit (Forsyth and Ponce, 2002). Then $f_x = fk$ and $f_y = fl$ can be used to describe the focal length in x and y directions, respectively. Furthermore, the pixels may even be non-rectangular. In such case, $s = -f_x \cot\alpha$ can be used to describe the skew of pixels. Here, α is the angle between the two pixel axes, which may not be exactly equal to 90°. In general, the origin of the image coordinate system $\mathcal{I}$ is at a corner of the retina (as depicted in Fig. 2.5) instead of the principal point. The coordinates of the principle point in $\mathcal{I}$ are (u_0, v_0). The intrinsic parameters are used to compose the affine transformation matrix K, which describes the mapping from the distorted image

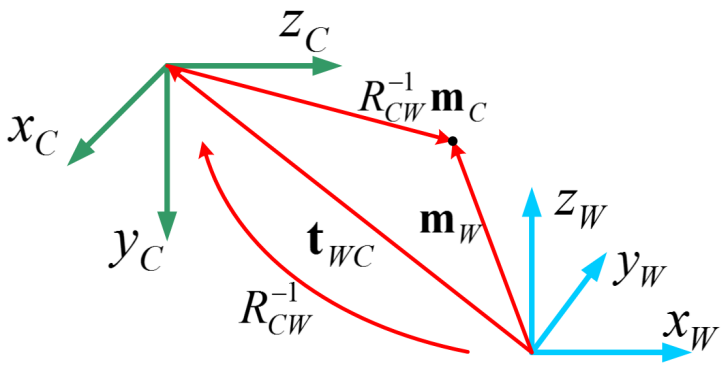


Figure 2.6: Relation between the camera and the world coordinate system, with vectors expressed in the world coordinate system $\mathcal{W}$.

coordinate system $\mathcal{D}$ to the image coordinate system $\mathcal{I}$ as

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{K} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{2.11}$$

Extrinsic camera parameters

In this thesis, points in 3D space usually need to be expressed in terms of a fixed Euclidean coordinate system (e.g. the world coordinate system $\mathcal{W}$) and the camera coordinate system $\mathcal{C}$. The relation between these coordinate systems is given by a rotation R_{CW} and a translation $\mathbf{t}_{WC}$, which are called the extrinsic camera parameters or the camera pose. R_{CW}^{-1} can be interpreted as the orientation of the camera frame and $\mathbf{t}_{WC}$ as the position of the camera centre in the world frame, as illustrated in Fig. 2.6. Then the mapping of a point $\mathbf{m}_w = [x_w, y_w, z_w]^T$ in $\mathcal{W}$ to a point $\mathbf{m}_c = [x_c, y_c, z_c]^T$ in $\mathcal{C}$ is

$$\mathbf{m}_c = R_{CW}(\mathbf{m}_w - \mathbf{t}_{WC}). \tag{2.12}$$

It is often convenient not to make the camera centre explicit, instead, to represent the transformation as $\mathbf{m}_c = R_{CW}\mathbf{m}_w + \mathbf{t}_{CW}$, where $\mathbf{t}_{CW} = -R_{CW}\mathbf{t}_{WC}$. By introducing homogeneous coordinates, Eq. (2.12) can be expressed as

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{CW} & \mathbf{t}_{CW} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}. \tag{2.13}$$

Central perspective camera model

Combining the forward transformations given in Eq. (2.9)–Eq. (2.13), the image coordinates $\mathbf{m}_p$ of a point $\mathbf{m}_w$ expressed in the world coordinate system can be computed as

$$\mathbf{m}_p = \mathcal{P}(\mathbf{m}_w) = (\mathcal{P}_k \circ \mathcal{P}_d \circ \mathcal{P}_n \circ \mathcal{T})(\mathbf{m}_w), \tag{2.14}$$

where $\mathbf{m}_c = \mathcal{T}(\mathbf{m}_w)$ maps $\mathbf{m}_w$ in the world frame $\mathcal{W}$ to the point $\mathbf{m}_c$ in the camera frame $\mathcal{C}$ according to Eq. (2.12), $\mathbf{m}_n = \mathcal{P}_n(\mathbf{m}_c)$ maps $\mathbf{m}_c$ to $\mathbf{m}_n$ in the normalized image frame $\mathcal{N}$ according to Eq. (2.8), $\mathbf{m}_d = \mathcal{P}_d(\mathbf{m}_n)$ corresponds to the distortion function as expressed in Eq. (2.10), and $\mathbf{m}_p = \mathcal{P}_k(\mathbf{m}_d)$ denotes the affine transformation from the distorted image frame $\mathcal{D}$ to the image frame $\mathcal{I}$ as given in Eq. (2.11).

We also define the mapping of a point $\mathbf{m}_c$ in the camera frame to the image coordinates $\mathbf{m}_p$ as

$$\mathbf{m}_p = \mathcal{P}_C(\mathbf{m}_c) = (\mathcal{P}_k \circ \mathcal{P}_d \circ \mathcal{P}_n)(\mathbf{m}_c). \tag{2.15}$$

$\mathcal{P}_C$ will be used in the technical chapters of this thesis.

2.3.2 Camera calibration

The aim of the camera calibration is to obtain the intrinsic parameters and distortion coefficients described in the previous section. The calibration procedure typically requires the collaboration of a calibration object with known geometry information. For complicated auto-calibration methods which relax this requirement, the readers are referred to the work in Maybank and Faugeras (1992); Hartley (1994); Luong and Faugeras (1997). The work in Zhang (1999) has made the calibration procedure rather flexible and simple by requiring images of a planar pattern from a few (at least two) different orientations. The general method proposed in that work can be extended to include more camera parameters to be estimated, as has been done in Bouguet (2001). Fig. 2.7 shows two example images of a planar pattern used for our calibration. The corner points detected in each image can be used to form a set of 2D/3D point correspondences $(\mathbf{m}_{p_{ij}}, \mathbf{m}_{w_{ij}})$, where i is the image index and j is the point index.

The calibration procedure consists of several steps, and also requires the extrinsic camera parameters to be estimated. The planar pattern is always assumed to be fixed in the world coordinate system $\mathcal{W}$, such that the origin of $\mathcal{W}$ coincides with one of its corners, and the $x_W - y_W$ plane is coplanar, i.e. each corner point has coordinates with $z_w = 0$. As initialization step, the intrinsic and extrinsic camera parameters are estimated from the homographies between the planar pattern and its images. The homographies are obtained from Eq. (2.14) while ignoring camera distortions. The initial solution is then refined by solving the nonlinear least squares problem:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j=1}^{m} \| \mathbf{m}_{p_{ij}} - \mathcal{P}(\mathbf{m}_{w_{ij}}, \mathbf{x}) \|^2, \tag{2.16}$$

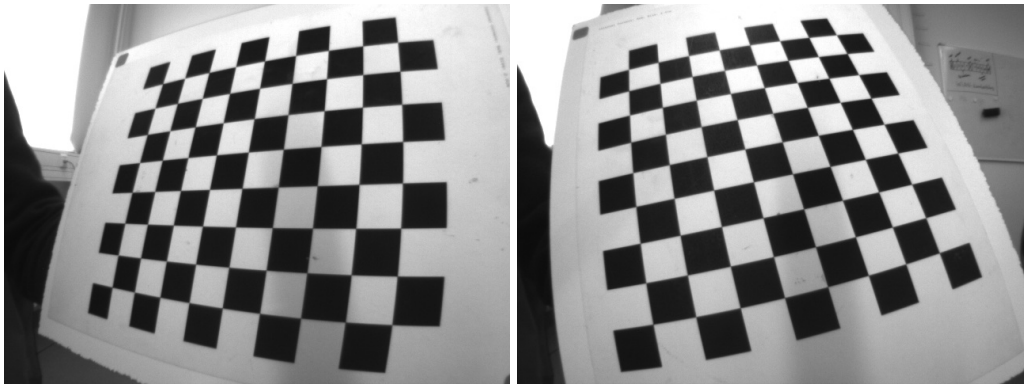


Figure 2.7: Two example images of a planar pattern used for our camera calibration.

where $\mathbf{x}$ comprises the distortion coefficients, the intrinsic and extrinsic parameters, $\hat{\mathbf{x}}$ is the estimate of $\mathbf{x}$, n is the size of the image set, m is the number of corner points on the calibration pattern, and $\mathcal{P}(\mathbf{m}_{w_{ij}}, \mathbf{x})$ corresponds to Eq. (2.14) while adding the parameters that should be estimated.

The toolbox presented in Bouguet (2001) is freely available online, and is used for the camera calibration in this thesis.

2.3.3 Extrinsic calibrations

In this thesis, the extrinsic calibration of the cameras onboard the MAV mainly consists of two calibration tasks: the extrinsic calibration between the MAV body frame and the camera frame, which is also known as hand-eye calibration (Tsai and Lenz, 1988; Horaud and Dornaika, 1995), and the extrinsic calibration among multiple cameras, i.e. calculating their relative poses.

We perform the extrinsic calibrations with the assistance of the external tracking system "Optitrack" by Naturalpoint (2014), which includes 12 infrared cameras. After attaching several markers to an object, the tracking system can provide 6DOF pose estimates of the object with a speed up to 100 *fps*. The deviation of position estimates for a static object is less than a millimetre according to our tests. This tracking system is also used to provide ground truth data of the MAV poses in the experiments throughout this thesis.

MAV-camera calibration

We define T_{MN} to be the homogeneous transformation matrix, which transforms a vector in coordinate system $\mathcal{N}$ to a vector in coordinate system $\mathcal{M}$. It consists of a rotation

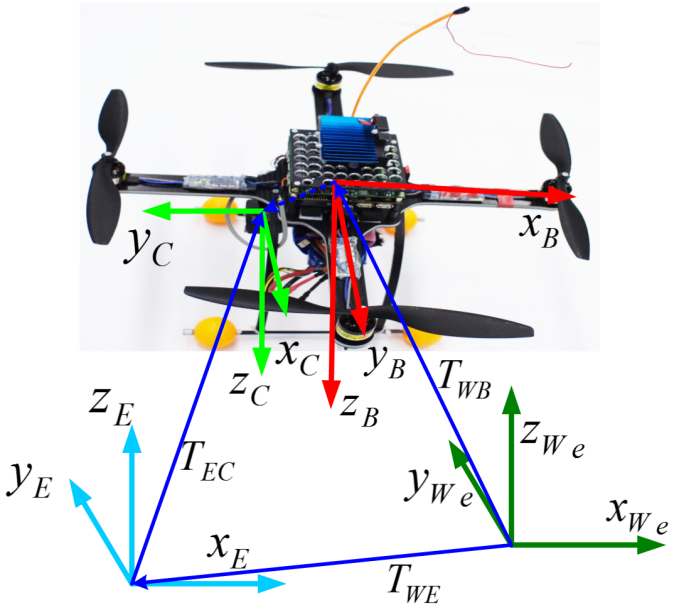


Figure 2.8: The coordinate systems and relative poses in the MAV-camera extrinsic calibration.

R_{MN} and a translation $\mathbf{t}_{MN}$:

$$T_{MN} = \begin{pmatrix} R_{MN} & \mathbf{t}_{MN} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}.$$

The camera frame ($\mathcal{C}$) and MAV body frame ($\mathcal{B}$) are defined as shown in Fig. 2.8. The MAV-camera calibration then finds the homogeneous transformation T_{BC}.

We obtain T_{BC} by estimating the transformations of the corresponding two coordinate systems with respect to an external frame $\mathcal{E}$, which is fixed on the planar pattern as we use in the camera calibration procedure described in the previous section. We attach several markers on both the planar pattern and the quadrotor, then place both of them within the field of view of the external tracking system, which provides measurements in its own world coordinate system $\mathcal{W}_e$. Thus, their 6DOF poses in $\mathcal{W}_e$, T_{WB} and T_{WE}, can be obtained from the measurements of the tracking system. Furthermore, we obtain the transformation between the camera frame $\mathcal{C}$ and the external frame $\mathcal{E}$, T_{EC}, by performing extrinsic-parameter calibration of the camera with respect to the planar pattern using the toolbox in Bouguet (2001). Finally, the desired transformation T_{BC} can be computed as:

$$T_{BC} = (T_{WB}^{-1} T_{WE}) T_{EC}. \tag{2.17}$$

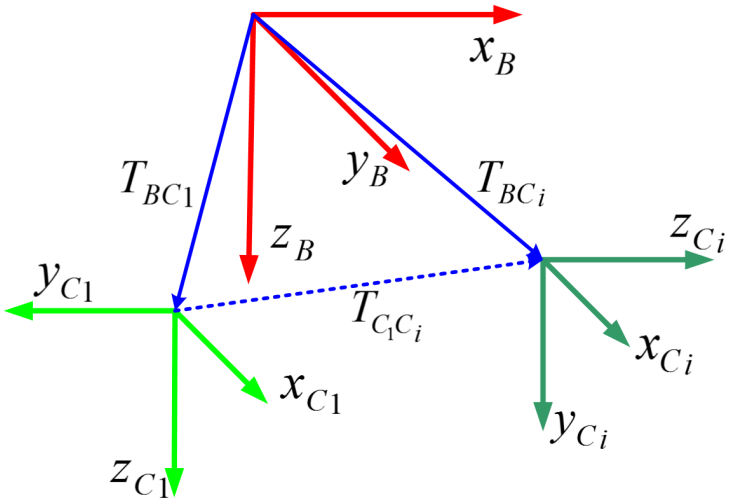


Figure 2.9: The coordinate systems and relative poses in the camera-camera extrinsic calibration.

Camera-camera calibration

The aim of the camera-camera calibration is to find the relative pose, $T_{C_1C_i}$, between camera C_1 and another camera C_i, as illustrated in Fig. 2.9. Since the cameras mounted on our MAV share no overlap in their respective fields of view, typical stereo-camera extrinsic calibration methods, e.g. those presented in Bouguet (2001); Faugeras and Toscani (1986); Luong and Faugeras (1993); Knight and Reid (2000), cannot be used directly. We achieve the calibration by performing the previously described MAV-camera calibration twice to obtain T_{BC_1} and T_{BC_i}. Thus, the relative pose $T_{C_1C_i}$ can be found as

$$T_{C_1C_i} = T_{BC_1}^{-1} T_{BC_i} \tag{2.18}$$

When an external tracking system is unavailable, alternative methods for camera-camera calibration can be used, such as the semi-automatic calibration methods presented in Li *et al.* (2005); Kaess and Dellaert (2010), or the auto-calibration methods presented in Carrera *et al.* (2011); Esquivel *et al.* (2007); Oskiper *et al.* (2007); Angst and Pollefeys (2009).

2.4 Visual SLAM for MAVs

When a mobile robot navigates in an unknown environment, its accurate localization is a prerequisite in order to build a good map of the environment. On the other hand, having an accurate map is essential in order to achieve good localization. These two

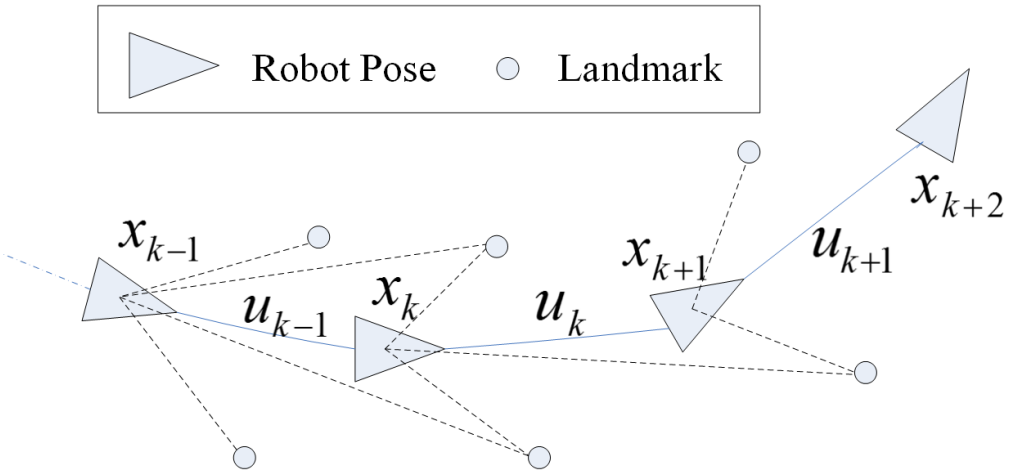


Figure 2.10: The SLAM problem. A SLAM system simultaneously estimate both the robot and the landmark locations, based on observations of landmarks.

problems are addressed as the SLAM problem, which is the process of simultaneously tracking the position of a mobile robot relative to its environment and building a map of the environment, as illustrated in Fig. 2.10.

At a time instance k, given a map achieved at time instance $k-1$, a set of observations to the landmarks in the environment, and a prior estimation of the relative motion of the robot, u_{k-1}, a SLAM system has to simultaneously estimate the robot location x_k and the locations of the landmarks in the environment, which have been observed. To initialize the map, different initialization procedures can be used, depending on the type of sensor modality used for SLAM. Detailed descriptions of the SLAM problem in a probabilistic perspective can be found in Thrun *et al.* (2005); Durrant-Whyte and Bailey (2006); Thrun and Leonard (2008).

SLAM using cameras, i.e. visual SLAM, relies on visual features, e.g. line segments or local features, as landmarks. A popular choice has been adopting local features, such as SIFT (Lowe, 2004), SURF (Bay *et al.*, 2006) and FAST (Rosten and Drummond, 2006), for visual SLAM. Using such visual features facilitates SLAM in unmodified and unknown environments, which is highly desirable for a wide range of applications, including our MAV applications.

2.4.1 The PTAM system

In Klein and Murray (2007), parallel tracking and mapping (PTAM) is proposed for estimating the camera pose in a small-scale unknown scene. In general, it is a monocular visual SLAM process. Here, we briefly introduce the PTAM system which will be used as a foundation of our SLAM systems in this thesis.

The original PTAM implementation can produce a detailed environment map with a

large number of landmarks, which can be used for accurate pose tracking of a monocular camera at a high frequency. In order to achieve real-time operation, the main idea proposed in PTAM is to split tracking and mapping into two separate threads, which can be processed in parallel on a dual-core computer. One thread is responsible for tracking the camera motion relative to the current map. The other thread extends the map that consists of 3D point features organized in keyframes, and refines the map by using bundle adjustment (Triggs *et al.*, 2000).

The tracking thread

In the thread responsible for tracking the camera pose (the tracking thread), the FAST corner detector (Rosten and Drummond, 2006) is applied to each image at four pyramid levels. All map points are projected to the current image frame based on a prior pose estimate, which is predicted by using a simple decaying-velocity motion model. The map points located inside the image after this projection are chosen as candidates which could be used for pose tracking. An 8×8-pixel patch search template is then generated for each map point. To locate each map point in the target pyramid level, a fixed-range search around its predicted position is performed to find the best match for its template. This is done by evaluating the zero-mean sum-of-square-difference (SSD) scores at all FAST corner locations within a circular search region, and by selecting the location with the smallest difference score. If this score is below a preset threshold, the map point is considered to have been found. Then the image coordinates of each map point can be refined by performing an iterative error minimization.

All the successful observations of the map points are utilized in an optimization process to estimate the camera pose update, as will be analyzed in Chapter 5. Depending on the current camera pose, the current keyframe will be decided whether to be added to the map or not.

The mapping thread

The mapping thread integrates new keyframes into the map when requested by the tracking thread, and creates new map points by triangulating FAST corner matches between the new keyframe and its closest neighbors. Local and global bundle adjustment are continuously performed to refine the map for the rest of the time. The operation of the mapping thread is illustrated in Fig. 2.11.

Limitations in MAV applications

In the context of adapting PTAM for autonomous navigation of MAVs, there are three issues we need to keep in mind: First, since the PTAM system features a monocular vision system, it does not provide metric scale measurements. Second, PTAM was originally designed for augmented reality applications in small areas, and thus is not suitable

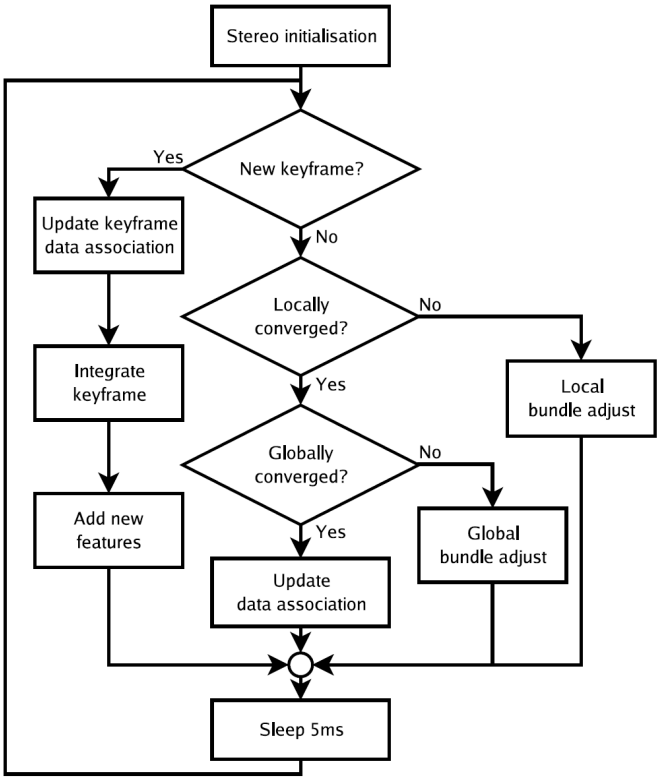


Figure 2.11: The flowchart of the mapping thread of PTAM. Reprinted from Klein and Murray (2007).

for large-scale SLAM. Third, if an MAV flies in complex environments or in aggressive ways that lead to failure in triangulating new map points or tracking the existing points, pose tracking of PTAM will consequently fail. All those three issues will be addressed when we discuss the SLAM systems implemented in this thesis.

2.4.2 Related work

Very early work related to visual SLAM can be found in Harris and Pike (1988). In this work, a 3D visual map is built from feature-points extracted from a sequence of images taken by a moving camera, with the ego-motion of the camera also being estimated. In recent years, real-time visual SLAM has been achieved using both monocular and stereo cameras. Two methodologies have become predominant in visual SLAM (Scaramuzza and Fraundorfer, 2011): filtering methods which fuse information from all past measurements in a probability distribution (Davison *et al.*, 2007; Eade and Drummond, 2007), and keyframe-based methods, like PTAM. Davison and Murray (2002) proposed a visual SLAM system that uses stereo cameras in a probabilistic framework of the Extended Kalman Filter (EKF). This work describes the first application of active vision to real-time sequential map-building in SLAM. Later in Davison (2003), SLAM at camera frame rate is achieved using a monocular camera. Further work in Davison *et al.* (2007) presents the monocular visual SLAM system, MonoSLAM, achieving a real-time and

drift-free performance. The monocular visual SLAM system in Eade and Drummond (2007) builds a graph of locally filtered sub-maps. Updates to the local sub-maps are carried out using non-linear optimization rather than EKF, in order to avoid the inconsistency resulting from imperfect approximation of the observation model. The advantages and disadvantages of filtering and keyframe-based methods are analyzed in the work of Strasdat *et al.* (2010a, 2012), suggesting that, in most modern applications, keyframe optimization gives the most accuracy per unit of computing time.

Most of the current SLAM systems have been limited to small-scale workspaces. Nevertheless, a number of systems have recently been developed for large-scale operations (Cummins and Newman, 2011; Mei *et al.*, 2011). To achieve real-time performance in large scale explorations, the work in Mei *et al.* (2009) combines accurate local visual odometry using stereo cameras with constant-time large-scale mapping. A continuous relative representation is chosen to represent the world. The resulting visual SLAM system obtains precisions down to a few metres over distances of a few kilometres. The monocular visual SLAM system in (Strasdat *et al.*, 2010b) obtains a near real-time performance using a framework similar to PTAM. In the pose-graph optimization (PGO) process at loop closures, the scale drift of the monocular vision system is taken into account. The real-time stereo SLAM system proposed in Lim *et al.* (2011) alternates bundle adjustment in a local window with global segment optimization.

With the availability of some relatively cheap and lightweight RGB-D cameras, like the Microsoft Kinect and the Asus XtionPro, some successful RGB-D SLAM systems have been developed. The KinectFusion system proposed in Newcombe *et al.* (2011) achieves accurate real-time mapping of complex and small-scale indoor scenes in variable lighting conditions. All depth data streamed from a Kinect sensor is fused into a single global implicit surface model of the observed scene for dense reconstruction. Mapping large indoor environments in near-real time has been achieved in Henry *et al.* (2014). The alignment between frames is computed by jointly optimizing over both appearance and shape similarity. Pose optimization is applied to achieve a globally consistent map after loop closures are detected. In Engelhard *et al.* (2011), a similar approach is implemented, while the more efficient SURF feature detector is applied instead of SIFT. To avoid problems caused by incomplete depth measurement within the field of view of the RGB-D camera, the SLAM system presented in Scherer *et al.* (2012) uses depth information only as additional constraints in bundle adjustment to improve the accuracy of the PTAM system.

Autonomous navigation of UAVs/MAVs relying on pose estimates from GPS sensors has been well studied in early researches. Related work usually fuses inertial navigation system (INS) data to aid GPS-sensor data, achieving autonomous navigation in high altitude and long range operations. However, they are not suitable in GPS-denied environments. Recently, much effort has been focused on developing visual SLAM systems to enable autonomous MAVs. Related work using stereo cameras, monocular cameras and RGB-D cameras can be found in the literature. Autonomous mapping and exploration for MAVs based on stereo cameras is presented in Fraundorfer *et al.* (2012). The work

in Schauwecker and Zell (2013) features a vision system for autonomous navigation of MAVs using two pairs of stereo cameras, with stereo triangulation adding constraints to bundle adjustment in PTAM. A stereo setup yields metric scale information of the environment. However, those systems have difficulties in using distant features since they triangulate those feature points based on their short baselines. Stereo visual odometry and SLAM systems may degenerate to the monocular case when the distance to the scene is very much larger than the stereo baseline. In this case, stereo vision becomes ineffective and monocular methods must be used (Scaramuzza and Fraundorfer, 2011).

In Achtelik *et al.* (2011), PTAM is used to provide position estimates for an MAV, while fusing data from an air pressure sensor and accelerometers to estimate the unknown metric scale factor of the monocular vision system. The work in Weiss and Siegwart (2011) presents a visual-inertial data fusion method based on the EKF. It is further implemented in Weiss *et al.* (2012) for autonomous navigation of MAVs using inertial data and visual pose estimates from a modified PTAM system. The scale drift of the monocular PTAM system has been considered in the EKF framework.

A vision-based system combining advantages of both monocular vision and stereo vision is developed in Shen *et al.* (2013b), which uses a low frame-rate secondary camera to extend a high frame-rate forward facing camera that is equipped with a fisheye lens. It can provide robust state estimates for a quadrotor by fusing the onboard inertial data. The resulting vision system mainly relies on monocular vision algorithms, while being able to track metric scale by stereo triangulation. Since the two cameras are configured in a stereo setup, the field of view of the vision system is not expanded. The improved work in Shen *et al.* (2013a) enables a quadrotor to autonomously travel at speeds up to 4 *m/s*, and allows roll and pitch angles exceeding 20 *degrees* in 3D indoor environments.

In Huang *et al.* (2011), autonomous flight of an MAV is enabled by the proposed SLAM system using an RGB-D camera. A visual odometry is used for real-time local state estimation, and integrated with the RGBD-Mapping (described in Henry *et al.* (2014)) to form the SLAM system. An efficient RGB-D SLAM system is described in Scherer and Zell (2013), which enables an MAV to autonomously fly in an unknown environment and create a map of its surroundings. Sparse optical flow is used for feature matching, which is of advantage when motion blur may result in a very limited number of local features being detected.

Chapter 3

Artificial-Landmark-Based Visual Pose Estimation

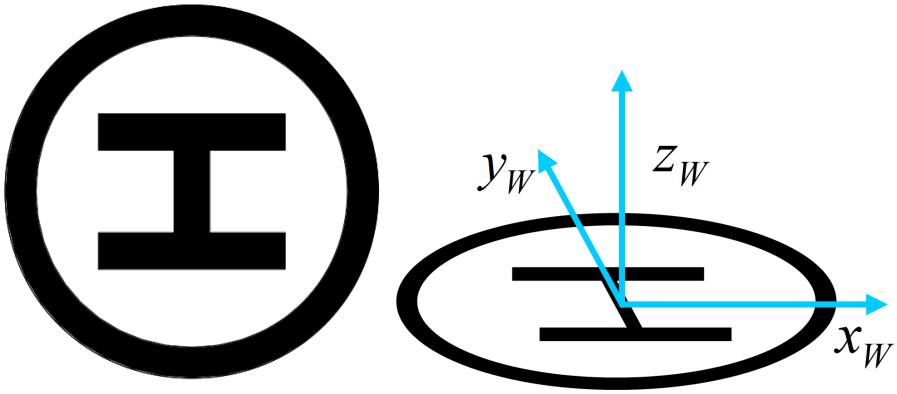
In this chapter, we address the problem of six degrees-of-freedom (6DOF) pose estimation of MAVs using an onboard monocular visual solution. The camera pose can be estimated based on a single image of an artificial landmark by the presented solution. This landmark is chosen to be similar to a regular helicopter landing pad, avoiding a too specified setup. The landmark detection is based on an artificial neural network (ANN), while the pose computation is mainly depending on a computational projective geometry method. The visual pose estimation algorithm developed in this chapter can be applied to enable autonomous takeoff, hovering and landing of an MAV. Furthermore, it will be used for automatic initialization of another visual SLAM system as will be shown in the other technical chapters of this thesis.

Large parts of this work have been pre-published in Yang, S. *et al.* (2012, 2013b).

3.1 Introduction

In MAV applications, visual pose estimation based on artificial landmarks (visual markers), has been a popular choice, since it is computationally very efficient. Related methods can be found in autonomous landing of MAVs on a specific target with previously known geometry information, assuming the target is very distinctive to the background, which can be recognized by a simple image processing method. Such a target can either be fixed on the ground (Saripalli *et al.*, 2002; Merz *et al.*, 2006), or on a moving platform (Saripalli *et al.*, 2003; Meier *et al.*, 2011). In those scenarios, marker-based pose estimation can be more efficient and accurate, compared with visual-feature-based methods (Lowe, 2004; Lepetit and Fua, 2006) or some advanced template-based methods (Hinterstoisser *et al.*, 2011). However, it has often been overlooked to improve the robustness of such marker-based methods in order to be extended to applications in more complex environments, in which the background of the target can be very cluttered.

In this chapter, a robust pose estimation method based on an artificial landmark is presented. We first chose a helicopter landing pad as the visual marker, which consists of a letter "H" surrounded by a circle, as depicted in Fig. 3.1a. After the landing pad is



(a) (b)

Figure 3.1: (a) The landing-pad landmark. (b) The world coordinate system.

robustly detected using an artificial neural network, its geometry information is retrieved by fitting ellipses to its image pattern. Then the five degrees-of-freedom (5DOF) pose of the MAV, its 3D position, roll and pitch angles, is estimated from the image projection of the known circle, using a computational projective geometry method (Kanatani, 1991) with a simple algebraic form. Since MAVs are nearly always equipped with an IMU, its gravity-vector estimates are used as a reference for solving the geometric ambiguity inherited from the computational geometry method. Furthermore, the remaining degree-of-freedom pose, the yaw angle, is estimated using the ellipse fitted to the image contour of the projected letter "H".

In the proposed method, no additional metric measurement sensor is needed for pose estimation. The 3D position and yaw angle estimates are used as input to the nested PID controller described in Sec. 2.2 to control hovering of the MAV above the landing pad, as well as the control of autonomous takeoff and landing.

The remainder of this chapter is organized as follows. Related work on artificial-landmark-based pose estimation methods in MAV applications is reviewed in Sec. 3.2. In Sec. 3.3, the vision algorithm for the landmark recognition is presented. Then the computational projective geometry method for 6DOF pose estimation based on the image of the landmark is presented in Sec. 3.4. The efficiency of the proposed method is demonstrated in the experiments in Sec. 3.5. Finally, we draw the summary of this chapter in the last section.

3.2 Related Work

Previous work on artificial-landmark-based pose estimation in MAV applications normally uses two kinds of visual markers with known geometry information: a group of point markers, and a group of curves.

In Wenzel *et al.* (2010b), four coplanar infrared LED markers are used in a low-cost solution for visual pose tracking of an MAV. A Wii remote infrared camera distributed by Nintendoand (Wikipedia, 2014b) is utilized to capture images of the four markers. In Wenzel *et al.* (2010a), a new geometric method is proposed using a different configuration of such four LED markers. Autonomous takeoff, hovering and landing of an MAV on a moving platform is achieved using the visual pose estimates based on those markers. Their further work in Wenzel *et al.* (2012), an MAV following another MAV is achieved in a follow-the-leader scenario. The relative pose estimates of the follower to the leader are provided by a modified method of that in Wenzel *et al.* (2010a). Since infrared LED markers are difficult to be used in outdoor environments, methods using passive point-markers have been investigated in a number of vision systems. The work in Masselli and Zell (2012) uses four coplanar-configured orange-ball markers for pose estimation of an MAV. The ball detection process is based on a color segmentation method. The MAV pose is estimated by solving the Perspective-3-Point (P3P) problem using a new method, with the fourth marker being used to reject false detections. In Jimenez Lugo *et al.* (2013), the same P3P solver is used to retrieve the relative pose between two quadrotors to solve the leader-following problem. Three passive orange ball markers are used in this work. A comparison of pose estimation methods using infrared LED markers, passive point-markers, and the method proposed in this chapter is presented in Masselli *et al.* (2014). The performance of those methods in both indoor and outdoor environments is evaluated.

Visual markers with various curves have been used for visual pose estimation of UAVs/MAVs. The work in Saripalli *et al.* (2003) achieves autonomous landing of a helicopter on a H-shaped landing pad. An image-moment-based method is used for detecting the landing pad and estimating the orientation of the helicopter. The relative position of the helicopter to the landing pad is estimated using these orientation estimates and precise height estimates provided by a differential-GPS sensor. Merz *et al.* (2006) designed a special landing pad with five circle triplets in different sizes. Three ellipses in the image, i.e. the projections of three circles of this pad, are used for estimating the relative pose in a coarse-to-fine process. The work in Lange *et al.* (2009) achieves autonomous landing and position control of an MAV by estimating the 3D position from a landing pad consisting of several concentric circles, assuming that the MAV is flying exactly parallel to the landing pad. The work in Xu *et al.* (2009) uses a T-shape cooperative object and an infrared camera for pose estimation of UAVs. However, only the yaw angle is computed in this method. More complicated visual markers can be found in Meier *et al.* (2011). This work presents a new self-developed quadrotor system capable of autonomous flight with onboard pose estimation using a vision system and an inertial measurement unit (IMU). The vision system features a marker-based approach with an adapted implementation of ARToolkit+ (Wagner and Schmalstieg, 2007).

Visual markers with circular curves are another alternative for visual pose estimation. The geometry of the image projection of a circle, which is an ellipse in the general case, is well studied in the area of projective geometry (Forsyth *et al.*, 1991; Kanatani and Wu,

1993; Chen *et al.*, 2004). However, this does not mean that the pose estimation methods in the previous work can be directly used in MAV autonomous flight applications, since there is a common geometric ambiguity in those approaches, i.e. two possible solutions exist. Thus, the absolute solution cannot be obtained from one image projection of one circle. The same issue applies to the work in Chen *et al.* (2004) for camera calibration, which uses two coplanar circles in the implementation. Recently, the work in Eberli *et al.* (2011) presents a vision algorithm which is able to estimate the 5DOF pose of an MAV by using two concentric circles with very different radii. However, this algorithm does not work when the camera is in an upright position above the circle. Thus, in this method, the attitude estimates from the IMU are used for the 3D position estimation.

3.3 Artificial-Landmark Recognition

In this section, the vision algorithm for robust and real-time landmark recognition is presented, including the detection of the target landmark and its geometry-information retrieving by accurately fitting ellipses corresponding to the image projection of the landmark.

3.3.1 The artificial landmark and coordinate systems

The landing-pad landmark is printed on A4 paper for the experiments. The radius of the outer and inner boundaries of the circle are 90 *mm* and 75 *mm*, respectively. A larger circle would lead to a larger working distance for the vision system. However, it would also lead to a larger "blind" range above the landmark, in which the camera cannot observe the complete landmark.

The world frame ($\mathcal{W}$) is defined in the way as illustrated in Fig. 3.1b. It is assumed to be the inertial frame. Due to the symmetric nature of the letter "H", we define the x_W axis along the forward direction of the MAV when it takes off.

3.3.2 Vision algorithm for landmark detection

Regular landing pads for helicopters are usually not particularly textured, whereas their background might be visually very cluttered. This is a difficult scenario for feature-based object detection methods, as they will find only few interest points on the landing pad compared to other objects in the background. We note that the landing pad consists of the letter "H" surrounded by a circle which is actually a distinguishing sign. Thus, a straightforward solution to the landmark detection is to treat it as a sign detection problem. It is then solved similarly as proposed in Scherer *et al.* (2011), by binarizing the image, finding connected components, and classifying connected components using an artificial neural network. We also enforce a geometric-relationship constraint to detect the landmark more reliably. Results in different steps are illustrated in Fig. 3.2. This

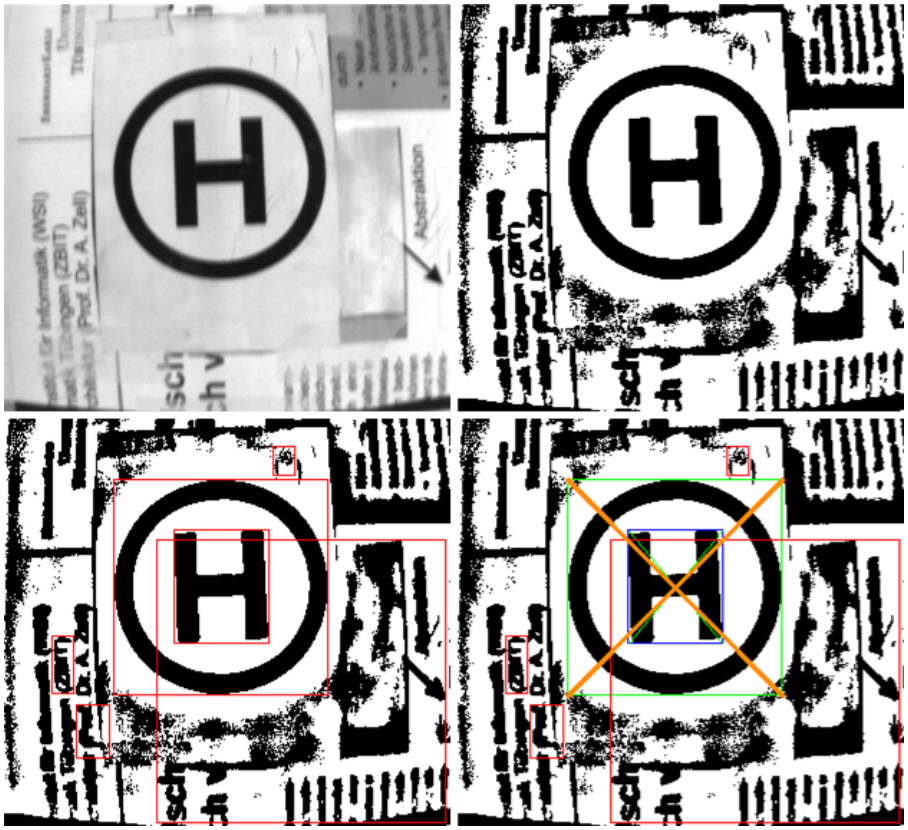


Figure 3.2: Different steps of landing pad detection: the original image (top left), the binary image (top right), connected components labelled with their bounding boxes (bottom left), and the classification result (bottom right).

method allows us to detect the landmark in real-time on computationally constrained hardware.

Binarization

In order to separate image pixels into two groups, each of which represents either objects or the background, adaptive thresholding proposed in Bradley and Roth (2007) is used to binarize each camera image. Here, we assume objects to be black and the background to be white in the binary image.

A basic way to binarize an image is to apply a global threshold to it. However, this will hardly work in cluttered environments, especially with dynamically changing illumination conditions. On the contrary, the adaptive thresholding method proposed in Bradley and Roth (2007) can cope well with illumination changes and even allows different lighting conditions in different parts of the image. It computes the average value $\bar{I}(i,j)$ of a $s \times s$ window of pixels centered around each pixel (i,j). If the value of the pixel (i,j)

is t percent less than $\bar{I}(i,j)$, then the pixel (i,j) is assigned to be black; otherwise, it is assigned to be white.

The average value $\bar{I}(i,j)$ can be efficiently computed by using the integral image, which is computed within a linear time cost as a preprocessing. Each location (i',j') of the integral image stores the sum of intensities of all pixels in the corresponding rectangle, which is defined by the pixel $(0,0)$ as its top left corner and the pixel (i',j') as its bottom right corner. When the surrounding $s \times s$ window of a pixel (i,j) is given by the top left corner (i_1,j_1) and the right bottom corner (i_2,j_2), $\bar{I}(i,j)$ is calculated as

$$\bar{I}(i,j) = \frac{I(i_2,j_2) - I(i_2,j_1-1) - I(i_1-1,j_2) + I(i_1,j_1)}{(i_2-i_1)(j_2-j_1)}.$$

One drawback of the adaptive thresholding method should also be noted: it has major difficulties when binarizing a black/white pattern with a large size, e.g. the circle of the landing pad in Fig. 3.2. When the dynamic window is smaller than the width of the curve of the circle, the image area of the curve will be randomly binarized depending on image noise. However, for the MAV takeoff and landing applications, the dynamic window size can be easily decided by a heuristic, depending on the required working range of the vision system in these applications.

Extracting Connected Components

Given a binary image described above, connected components of all objects in it are extracted based on the run-based two-scan labeling algorithm proposed by He *et al.* (2008) and its modifications presented in Scherer *et al.* (2011). A *run* is a set of contiguous object pixels within a certain row of an image. The original labeling algorithm in He *et al.* (2008) uses runs as basic elements of a connected component for detecting the connectivity, and traverses the image twice to label all connected components.

In the first scan, the original labeling algorithm extracts runs in the image by pixel-wise traversal in the raster scan direction. If a new run is not eight-connected with any run in the row above the current scan row, all pixels in it are assigned a new provisional label; otherwise, those pixels are assigned the same provisional label as one of its eight-connected run. Furthermore, all provisional labels that are assigned to a connected component are combined in a provisional label set, with a *representative label*. Whenever temporary connected components are found to be connected, all corresponding provisional label sets are merged into one set. Thus, after the fist scan, all provisional labels which belong to each connected component are combined in one provisional label set, with a final *representative label*. In the second scan, the labeling algorithm traverses the image again simply to replace each provisional label with its final *representative label*.

We use a similar strategy as in Scherer *et al.* (2011) to modify the algorithm in He *et al.* (2008). First, since completely labeling all pixels in an image is not necessary for the sign detection task in our work, the second scan described above is abandoned. Second,

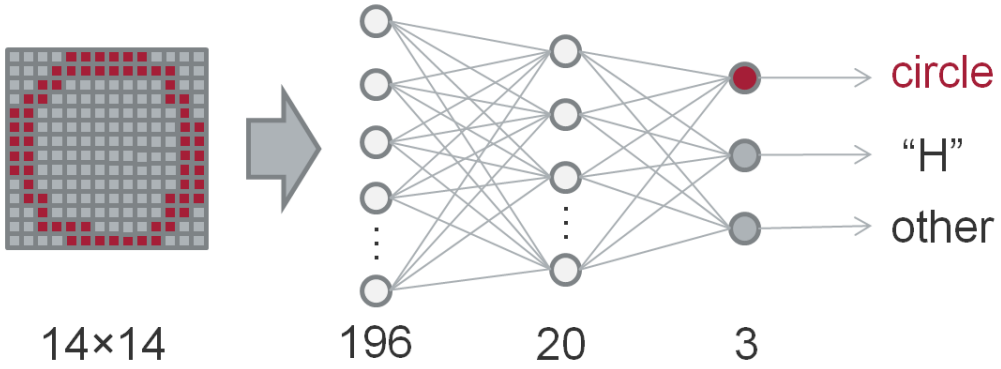


Figure 3.3: Illustration of the classification using the neural network. The size of the image pattern and the neural network are listed in the bottom. From the left to the right: resized image pattern, the input layer, the hidden layer of the neural network, and the output neurons. The classifier requires one output neuron per class. The final classification output is chosen after applying the loss function.

we store the bounding box of each connected component, which corresponds to one provisional label set in the first scan. This allows us to disregard connected components which do not have a proper dimension to be a part of the landmark sign. The proper bounding-box dimension is decided by a heuristic: The bounding box should not be too small to be reliably classified, and the ratio of its height to its width should be in a certain range. In this way, after the first scan of the image, we can efficiently extract a set of connected components which might be parts to the landmark sign, based on their bounding boxes.

Classification of Connected Components

We classify each of the connected components detected in the image using an artificial neural network (ANN) (Bishop, 2006; Dayhoff, 1990). This neural network assigns one of three classes to each connected component: *Circle*, *letter “H”* or *other object*. As the classification result shown in Fig. 3.2, the circles and the letter “H”s are marked with green and blue bounding boxes, respectively; other objects are marked with red ones. We implement this by using the 1-of-3 target coding scheme, which requires one output neuron per class: For each input sample, we expect the output neuron corresponding to its true class to return 1 and the others to return 0 after we apply the loss function, as illustrated in Fig. 3.3.

The structure of the neural network is a multilayer perceptron with 196 input units (one per pixel of patterns resized to 14×14), only one hidden layer consisting of 20 hidden units, and three output units. All units of the hidden layer use the logistic activation function. However, the units of the output layer use the softmax activation function,

Figure 3.4: Samples belonging to different classes for training the neural network, from the top row to the bottom row: circles, letter "H"s and other objects.

so we can interpret the output value of each output unit as the posterior probability of the input patch belonging to its corresponding class. In the loss function, Scherer *et al.* (2011) assign a much larger loss to false positives compared with that to false negatives, due to the concern that false positives and wrong detections cause worse consequences to the robot control than false negatives can do. In our case, we have the same concern. However, after we apply the geometric constraint which will be introduced later, false positives can be significantly suppressed. Thus, we applied a loss function that assigns a loss of 1 to false negatives and a loss of 2 to false positives and wrong detections. The final classification output is then chosen to minimize the expected loss.

We use standard backpropagation to train the network, based on a labeled dataset containing approximately 3,000 samples for each of the two parts of the landmark (circle and letter "H") and 7,000 samples of other objects, taken from various perspectives relative to the landmark. Some examples of the samples can be found in Fig. 3.4. We carry out the training process by using the Stuttgart Neural Network Simulator (SNNS), proposed by Zell *et al.* (1994), in a progressive fashion: We first manually label a small number of samples, which are used to train a preliminary neural network. Then we use this preliminary network to classify all remaining samples in the dataset. Thus, we only need to manually verify the results and correct the labels of those samples which have been classified to a false class. Finally, all the labeled samples in the dataset are used to train the final neural network. After training the network, we can generate efficient C code using the snns2c module of the SNNS, which is one of the reasons that we chose SNNS for this training task.

Enforcing the geometric relationship constraint

Once all connected components are classified by the neural network, we can suppress false positives by enforcing the geometric constraint that each letter "H" which is a part of our landmark has to be surrounded by a circle. Therefore, we disregard all connected components classified as letter "H"s that do not lie within the bounding box of a connected component classified as a circle. The relative sizes and center positions of their

bounding boxes are also considered in this constraint. Let $p_1,\ p_2$ be the false positive rates of the circle and the letter "H" detections, respectively, due to this geometric consistency check, the false positive rate for the final landing pad detection will be lower than $\min(p_1,\ p_2)$, usually by a large amount. In fact, we seldom encountered a single false positive of the landing pad detection during our flight experiments.

If our system is still certain that it has detected the landmark at this point, it extracts the corresponding original image pattern for further processing. In Fig. 3.2, the correct combination of a circle and letter "H" is marked with an orange cross, and the original image pattern inside the green bounding box will be used as the finally detected landing pad.

3.3.3 Retrieving the geometry information

In the general case, the perspective image projection of a circle is an ellipse. To make our vision system reliable in a large range of perspective views, this general case is considered in our work. Thus, accurate ellipse fitting turns out to be a critical issue for the later geometric computation. In this section, we obtain the ellipses corresponding to the inner and outer boundary of the landing pad circle and that of the letter "H". This is achieved by performing edge detection on the gray scale image pattern of the landing pad and fitting ellipses to these detected edges. Lens distortion effects to the ellipse parameters are taken into account during this process.

First, the well known Canny edge detector (Canny, 1986) is applied to the gray scale image pattern. The edge contours retrieved from the detected edges are then used to fit ellipses by implementing a so called direct least square fitting algorithm proposed in Fitzgibbon *et al.* (1999), which is very robust to image noise and very efficient. A comprehensive comparison of this algorithm with some other ellipse fitting algorithms can also be found in Fitzgibbon *et al.* (1999). An implementation of this algorithm exists in OpenCV (Bradski, 2000).

Due to lens distortion, especially when using a wide angle lens, the assumption of a perspective projection no longer applies. To eliminate the effect of this issue, we apply a correction step to the edge contours before applying the ellipse fitting algorithm: We transform the edge contour from the image frame into the undistorted image frame. For this step, the camera model calibrated in Sec. 2.3.2, is adopted. To further improve the efficiency of this transformation, a look-up table mapping distorted image coordinates to undistorted image coordinates is pre-computed at the initialization phase of the vision system.

Even though we can detect two ellipses for the circle of the landing pad, we only use the ellipse corresponding to its outer boundary for further pose estimation. Fitting an ellipse to the projected contour of the letter "H" provides us with the orientation of the landing pad, which will be described in Sec. 3.4.2.

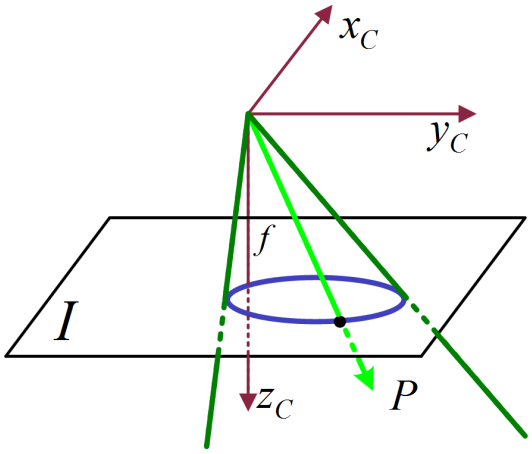


Figure 3.5: A bundle of straight lines passing through the camera optical center and an ellipse on the image plane.

3.4 6DOF Pose Estimation

In this section, we use a single ellipse originated from the projection of a known circle for the 5DOF pose estimation of the MAV within the world frame $\mathcal{W}$, including its 3D position $\mathbf{t}_{WB}$ and the roll and pitch angles (ϕ, θ) of the MAV. ϕ and θ are derived from the normal vector of the plane on which the circle lies. Chen *et al.* (2004) also described this problem for camera calibration, however, with two arbitrary coplanar circles. We briefly introduce it for the pose estimation here, and then use an IMU-aided approach to resolve the ambiguity inherited from this problem. The yaw angle of the MAV (ψ) is estimated as the orientation of the major axis of the ellipse fitted to the projection of the letter "H".

3.4.1 5DOF camera pose estimation

Once the effect of the lens distortion has been corrected, we can consider the vision geometry to obey perspective projection, in which a pinhole camera model applies. An ellipse in the image frame can then be described by the following quadratic equation:

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0, \tag{3.1}$$

or if we define the augmented vector $\mathbf{x} = (x,\ y,\ 1)^T$, we get

$$\mathbf{x}^T \begin{bmatrix} A & B & D \\ B & C & E \\ D & E & F \end{bmatrix} \mathbf{x} = 0. \tag{3.2}$$

Let f be the focal length of the camera, we can define the image plane to be at $z = f$. A bundle of straight lines, passing through the optical center and the ellipse, define an oblique elliptical cone, as illustrated in Fig. 3.5, in the form of

$$P = k(x,\ y,\ f)^T, \tag{3.3}$$

where k is a scale factor describing the distance from the origin to P. Combining Eq. (3.2) and Eq. (3.3), the equation to describe the oblique elliptical cone is:

$$P^T Q P = 0, \tag{3.4}$$

where

$$Q = \begin{bmatrix} A & B & \frac{D}{f} \\ B & C & \frac{E}{f} \\ \frac{D}{f} & \frac{E}{f} & \frac{F}{f^2} \end{bmatrix}, \tag{3.5}$$

which is called a conic in Kanatani and Wu (1993).

We directly derive the 5DOF pose of the circle in the camera frame from the conic Q. The proof of this result is detailed in Kanatani and Wu (1993) and Chen *et al.* (2004). We define r to be the radius of the original circle which is projected as the ellipse, λ_1, λ_2, and λ_3 to be the eigenvalues of Q, and $\mathbf{u}_2$ and $\mathbf{u}_3$ to be the unit eigenvectors for eigenvalues λ_2 and λ_3, respectively. As Q has a signature of (2, 1) (Kanatani and Wu, 1993), without the loss of generality, we can assume that $\lambda_3 < 0 < \lambda_1 \leq \lambda_2$. Then following the world frame definition in Fig. 3.1b, the unit vector of the z_W axis and the origin of the world frame described in the camera frame (denoted by $\mathbf{n}$ and $\mathbf{t}_{CW}$) are given by

$$\mathbf{n} = S_1 \sqrt{\frac{(\lambda_2 - \lambda_1)}{(\lambda_2 - \lambda_3)}} \mathbf{u}_2 + S_2 \sqrt{\frac{(\lambda_1 - \lambda_3)}{(\lambda_2 - \lambda_3)}} \mathbf{u}_3, \tag{3.6}$$

$$\mathbf{t}_{CW} = z_0 \left(S_1 \lambda_3 \sqrt{\frac{(\lambda_2 - \lambda_1)}{(\lambda_2 - \lambda_3)}} \mathbf{u}_2 + S_2 \lambda_2 \sqrt{\frac{(\lambda_1 - \lambda_3)}{(\lambda_2 - \lambda_3)}} \mathbf{u}_3 \right), \tag{3.7}$$

where $z_0 = S_3 \frac{r}{\sqrt{-\lambda_2 \lambda_3}}$, and S_1, S_2 and S_3 are the undetermined signs. Eq. (3.6) and (3.7) give us a clear algebraic formula for the 5DOF pose estimation. As $\mathbf{n}$ faces to the camera, and the center of the circle is in front of the camera in our definition, we can get two constraints for the undetermined signs,

$$\mathbf{n} \cdot (0,\ 0,\ 1)^T < 0, \tag{3.8}$$

$$\mathbf{t}_{CW} \cdot (0,\ 0,\ 1)^T > 0. \tag{3.9}$$

Since only two of the three signs can be determined by Eq. (3.8) and (3.9), there are two possible solutions for $\mathbf{n}$ and $\mathbf{t}_{CW}$. When the camera is in an exactly upright position above the circle, we get $\lambda_1 = \lambda_2$, and only one solution exists. In the general case, let us denote these two solutions to be $\mathbf{n}_1$, $\mathbf{t}_1$ and $\mathbf{n}_2$, $\mathbf{t}_2$. Further disambiguation is needed to obtain the absolute 5DOF pose estimation.

Resolving the Ambiguity

The gravity vector described in the camera frame can be calculated from the roll and pitch angles (ϕ and θ) estimated by the IMU. We use it as a reference to resolve the geometric ambiguity.

Two assumptions are introduced for the disambiguation step: The error of the attitude estimates by the IMU is small, and the landing pad is placed horizontally oriented. Since the attitude estimates by the IMU are used for high frequency attitude control, the first assumption should be met, otherwise an MAV could not fly properly. Fortunately, this assumption does apply to our IMU and most commercial ones. In this case, the error of the gravity vector estimation will be small correspondingly. Also, we usually want an MAV to land on leveled ground. Therefore, it is reasonable to make the second assumption.

Following the second assumption, the gravity vector $\mathbf{g}$ is antiparallel to the z_W axis. While estimating the 5DOF pose, we assume the yaw angle to be $\psi = 0$. Then the rotation matrix from frame B to frame W can be derived as,

$$R_{WB} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix}. \tag{3.10}$$

Let $\mathbf{z}_W$ be the unit vector of the z_W axis, then in frame B, $\mathbf{z}_w$ can be expressed as

$$\mathbf{z}_{BW} = (R_{WB})^{-1} \cdot \mathbf{z}_W.$$

In the camera frame $\mathcal{C}$, we have

$$\mathbf{z}_{CW} = R_{CB} \cdot \mathbf{z}_{BW}. \tag{3.11}$$

$\mathbf{z}_{CW}$ is used as the final reference vector for resolving the ambiguity of the 5DOF pose estimation according to the angles of the vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ to $\mathbf{z}_{CW}$, denoted by θ_1 and θ_2. According to our assumptions, the correct vision measurement should be close to the IMU measurement. So we choose the vector with smaller angle relative to $\mathbf{z}_{CW}$ to be the final estimate of the unit vector $\mathbf{n}$, which means

$$\mathbf{n} = \begin{cases} \mathbf{n}_1 & \text{if } \theta_1 < \theta_2 \\ \mathbf{n}_2 & \text{if } \theta_1 > \theta_2. \end{cases} \tag{3.12}$$

Thus, the last undetermined sign in Eq. (3.6) and (3.7) is now solved, and the 5DOF pose can be derived.

It should be noted that if the error of the gravity vector estimated from the IMU is higher than the angles spanned by the two possible solutions to the true gravity vector, the disambiguation may fall to the false result according to Eq. (3.12). However, as predicted by our assumptions, this should seldom happen. And if this ever results in a

large difference between the previous and current estimate, the current one can simply be disregarded as an outlier.

3.4.2 Resolving the remaining DOF

For yaw angle (ψ_C) estimation, the existing work uses image moments to estimate the orientation of a letter "H", as described in Saripalli *et al.* (2003). In our work, the orientation of the major axis of the ellipse fitted to the letter "H" is used as an approximation of ψ_C. We tested these two methods by using a synthetic image of a letter "H", and rotating it with a step size of one degree. Errors of less than 3 degrees were achieved by using the ellipse fitting method in this case, and less than 1 degree for the image-moment-based method. But when fitting the ellipse to a solid rectangle, even smaller errors were achieved. The errors from both of the two methods would be larger in real applications with noise and perspective projection. We adopt the ellipse fitting approach because of the following advantages. First, additional computational cost can be avoided, and we can unify the 6DOF pose estimation by using the ellipse fitting method. Furthermore, we can use other patterns with rectangular shape to replace the letter "H" as long as we train the neural network with this new pattern, making the vision algorithm more flexible. This approach is demonstrated to be sufficient for controlling the yaw angle of the MAV in our experiments in Sec.3.5.

Due to the symmetric nature of the letter "H", the yaw angle of the MAV has two solutions, with a difference of 180°. As this is inherited from the symmetric configuration of the landing pad, and does not affect the autonomous flight, we simply ignore this issue and assume $-90° < \psi_C < 90°$.

3.4.3 6DOF MAV pose

Until now, the estimated 6DOF pose is describing the pose of the world frame $\mathcal{W}$ expressed in the camera frame $\mathcal{C}$. The 6DOF pose of the MAV in $\mathcal{W}$ is obtained by performing a few basic transforms from $\mathcal{C}$ to $\mathcal{W}$.

Let R_n be the rotation matrix transforming the vector $\mathbf{n}$ in Eq. (3.12) to the z_C axis, which can be calculated by using Rodrigues' formula (Faugeras, 1993). From Eq. (3.6) and (3.7), R_n and ψ_C, we get the 6DOF pose of the camera in the world frame in the forms of

$$R_{WC} = \begin{bmatrix} \cos\psi_C & -\sin\psi_C & 0 \\ \sin\psi_C & \cos\psi_C & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R_n, \tag{3.13}$$

$$\mathbf{t}_{WC} = z_1 \left(\lambda_3 \sqrt{\frac{(\lambda_2 - \lambda_1)}{(\lambda_2 - \lambda_3)}} \mathbf{u}_2 + S\,\lambda_2 \sqrt{\frac{(\lambda_1 - \lambda_3)}{(\lambda_2 - \lambda_3)}} \mathbf{u}_3 \right), \tag{3.14}$$

where $z_1 = S' \frac{r}{\sqrt{-\lambda_2 \lambda_3}}$, and S, S' are determined by Eq. (3.8), (3.9) and (3.12).

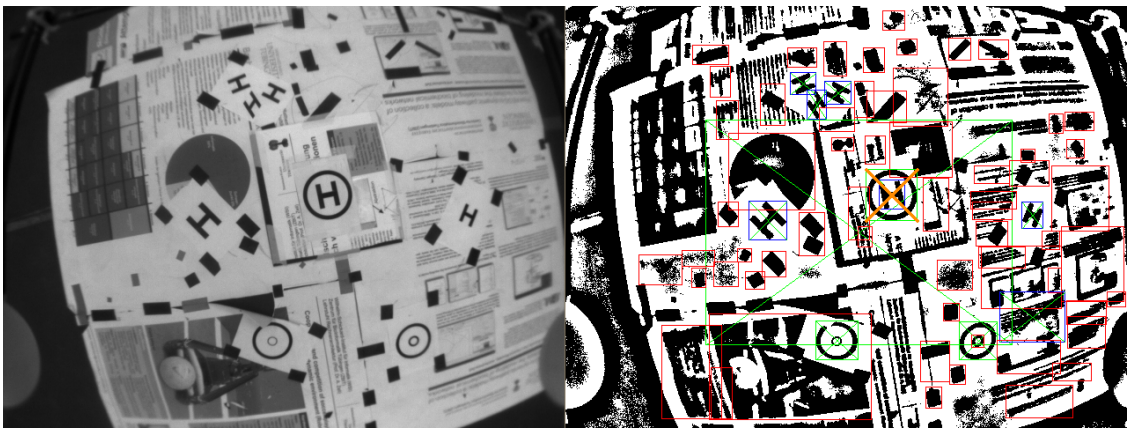


Figure 3.6: Landing pad recognition results when the MAV hover above the landing pad in cluttered environment.

Finally, we get the 6DOF pose of the MAV in the world frame as follows,

$$R_{WB} = (R_{BC} \cdot R_{CW})^{-1}, \tag{3.15}$$

$$\mathbf{t}_{WB} = -R_{WB} \cdot (R_{BC} \cdot \mathbf{t}_{CW}). \tag{3.16}$$

$\mathbf{t}_{WB}$ is the 3D position of the MAV in the world frame, and the rotation matrix R_{WB} gives the three individual roll, pitch and yaw angles of the MAV, which can be calculated by the Euler decomposition of R_{WB} (Diebel, 2006).

3.5 Experiments and Results

In this section, the efficiency of the presented vision system is demonstrated comprehensively by comparing its pose estimates to ground truth data provided by an external tracking system and by using its pose estimates as control inputs of our MAV for its autonomous takeoff, hovering and landing.

3.5.1 Landmark detection and ellipse fitting

Fig. 3.6 shows an example of image processing results for the landmark recognition, with the same color labels as used in Fig. 3.2. Besides the landing pad, we use some other circles and letter “H”s with different orientations and stretched shapes attached to some posters which are rich of texture features. All our latter experiments are done in the same cluttered environment. It shows that different circles and letter “H”s can be efficiently detected even if the perspectives change dramatically. False positives for the

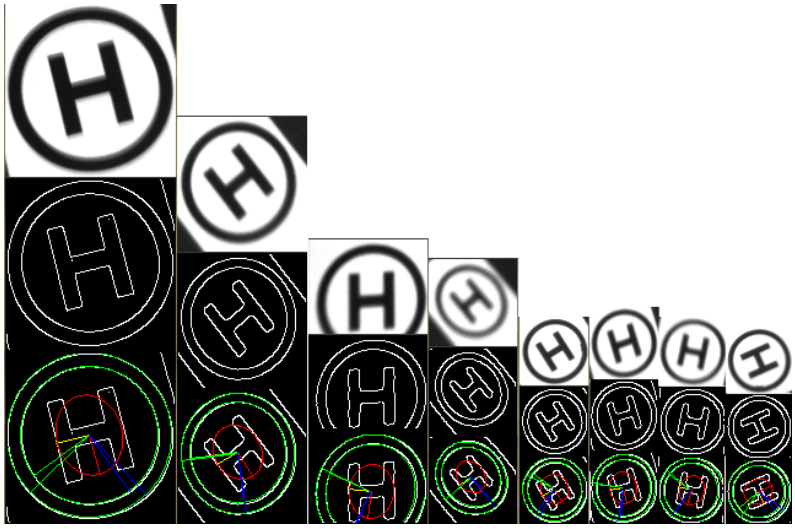


Figure 3.7: Image patches of the landing pad from different perspectives and the corresponding edge detection and ellipse fitting results.

individual circle class or the letter “H” class may appear, but will not be finally classified as a landing pad.

Fig. 3.7 shows the results of fitting ellipses to a few gray scale image patches depicted above, detected from various perspectives. Some of them clearly exhibit the effect of motion blur. The major and minor axis of the ellipses are also plotted. The orientation of the major axis fitted to the letter “H” provides an approximation for the yaw angle of the camera.

3.5.2 6DOF pose estimation results

We compare the 6DOF pose estimates of the onboard vision system with ground truth data from the tracking system, both recorded at a frequency of 60Hz. The 3D position and the attitude of the MAV are compared separately.

Hand-held Case

First, we manually rise and hover the MAV above the landing pad so that a large range of perspective changes of the camera can be tested. As shown in Fig. 3.8, estimates of the onboard vision system are plotted in red, the ground truth data in green. The 5DOF onboard vision pose estimates are well in line with ground truth data, without many obvious outliers, even though the position and attitude of the MAV change in a large range. When the landing pad gets further away from the camera, its image projection will get smaller respectively, and thus image noise will cause larger errors to the pose

Table 3.1: RMSEs in different cases.

	Hand-held	Hovering	Auto
$X_W Y_W$ RMSE (mm)	43.1	38.8	33.7
Z_W RMSE (mm)	7.9	5.7	6.8
3D RMSE (mm)	43.8	39.2	34.4
ϕ RMSE (deg)	1.4	1.3	1.4
θ RMSE (deg)	1.2	1.4	1.5
ψ RMSE (deg)	7.2	4.8	2.7

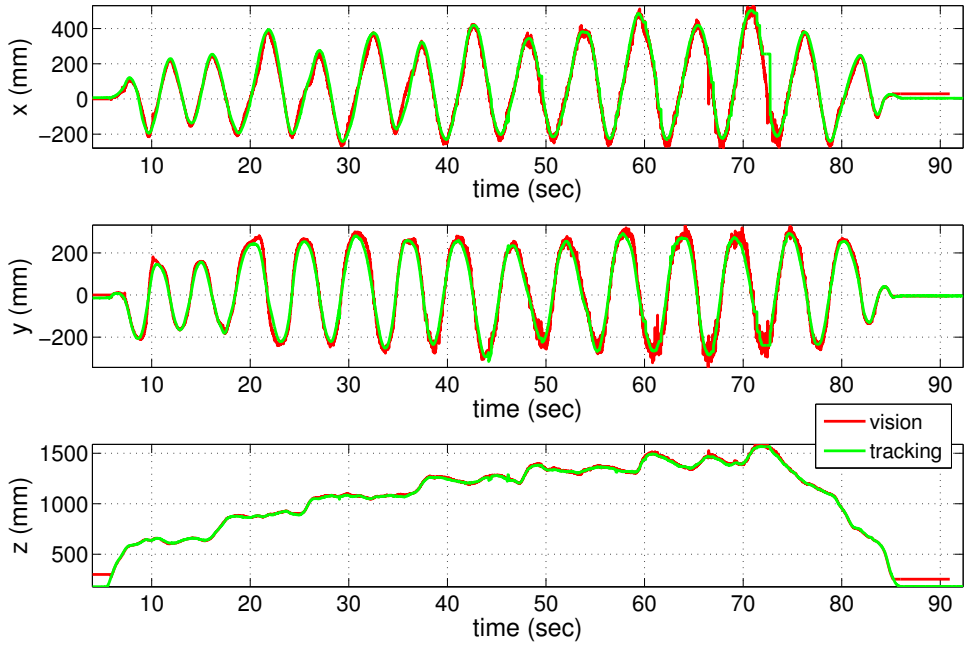
estimates, which can be found in Fig. 3.8 when the MAV was hovering at a height of around 1.5 meters.

Deviations of the yaw angle are larger than those of the roll and pitch angles, especially when the MAV is at poses where the image projections of the letter "H" are warped dramatically. This is because we use the approximation method as described in Sec. 3.4.2, where only the rotation of the letter "H" is considered. Since the IMU, the onboard vision system and the tracking system perform very similarly for the roll and pitch estimation, to clearly demonstrate the performance of the onboard vision system, we omit those estimates provided by the IMU in the figures, even though they are required by both the vision algorithm and attitude control of the MAV.

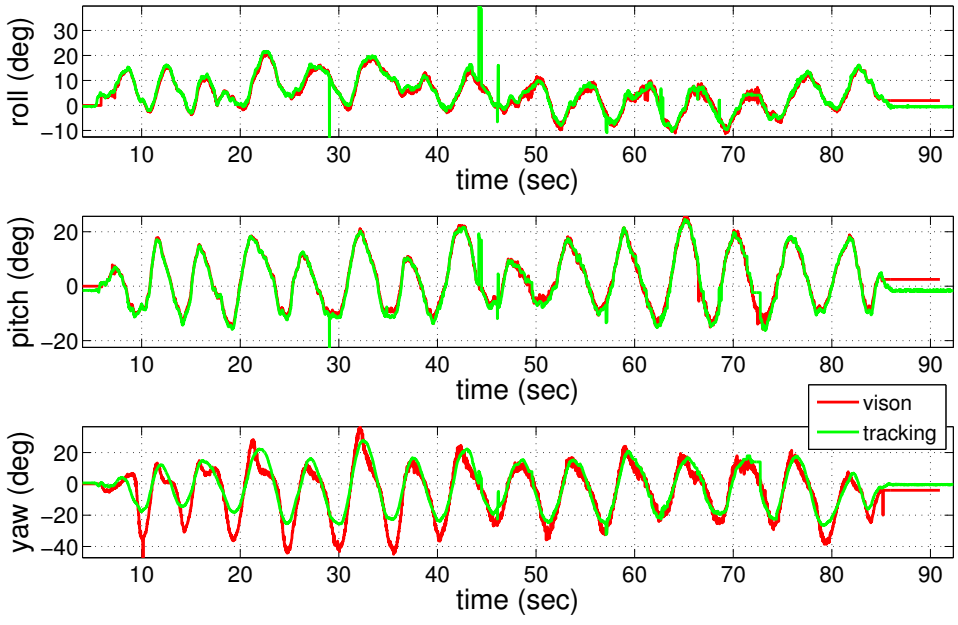
In Fig. 3.8, outliers of the ground truth data are mainly caused by occlusions of the attached markers on the MAV when the MAV is held in certain poses. We initialize the onboard vision pose estimates with the 3D position $[0,\ 0,\ 300]^T mm$ and the identity matrix for the rotation matrix, as the circle is not fully visible when the camera is very close to it, e.g. when the MAV has landed on the landmark. We compute the root-mean-square errors (RMSEs) of the onboard 6DOF pose estimates for the whole trajectory by comparing them with the ground truth data. The raw 3D RMSE in Table 3.1 is for the distance of the onboard vision 3D position estimates to the ground truth data, and $x_W - y_W$ RMSE is for the distance on the $x_W - y_W$ plane.

Hovering Case

Fig. 3.9 shows an autonomous hovering flight with a set point of $[0,\ 0,\ 1000]^T$ (mm) for more than 60 seconds. More motion blur from the vibration of the MAV is introduced during the flight, which will increase the errors of the pose estimates. We still manually added disturbances to the control command to let the MAV hover in a larger area, so that a relatively large range of perspectives could still be tested in this case. Compared with the hand-held case, the performance is not much worse and the RMSEs are even smaller. This is not surprising as in autonomous hovering flight, the three Euler angles are normally very small, and the MAV usually does not reach such poses where large

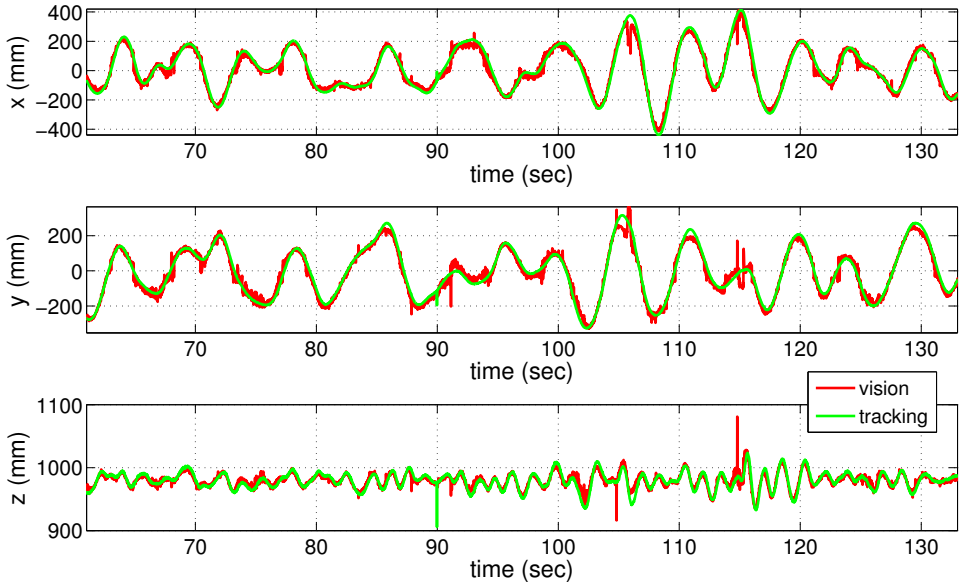


(a)

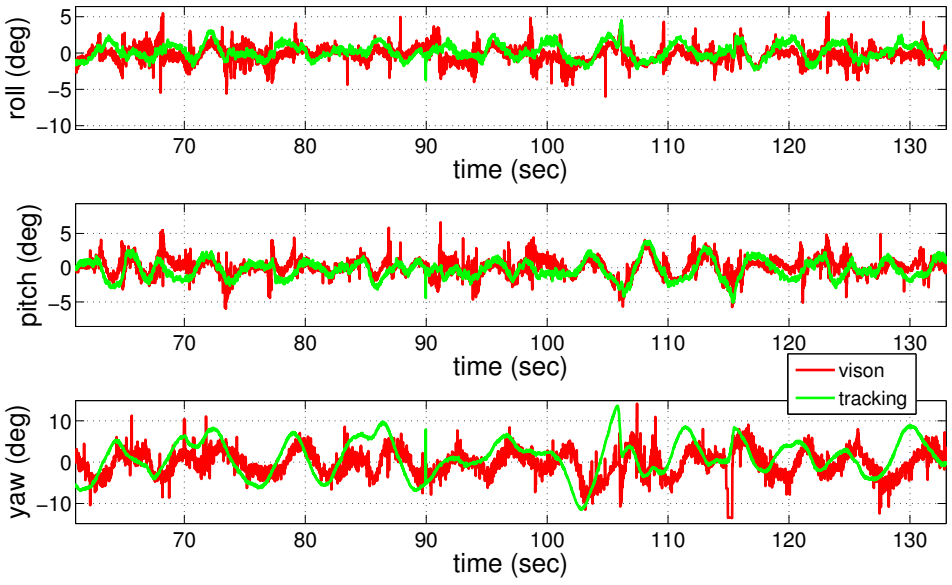


(b)

Figure 3.8: (a) Position and (b) attitude estimates with hand-held MAV.



(a)



(b)

Figure 3.9: (a) Position and (b) attitude estimates during a hovering flight with remotely added disturbances to the controller.

errors may be introduced.

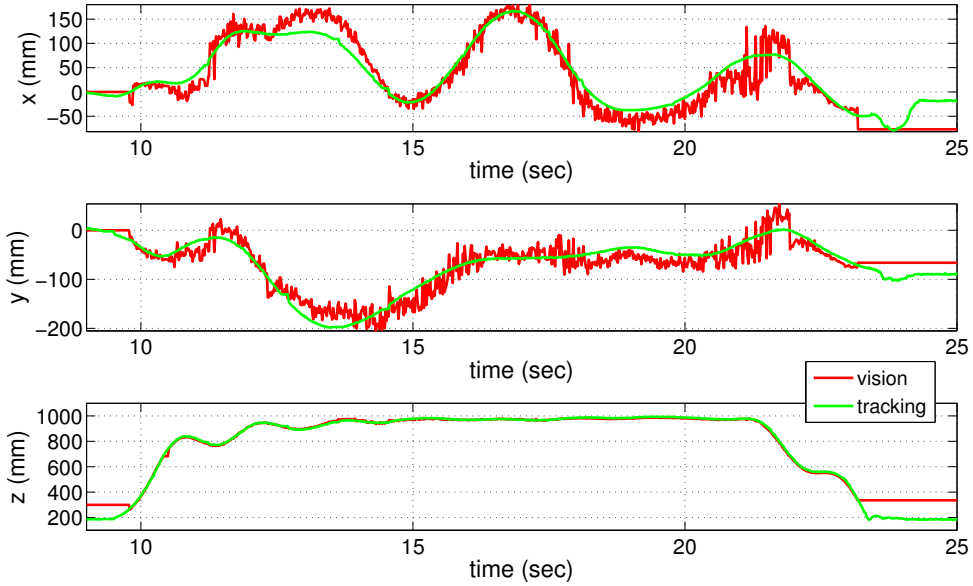
Autonomous Flight

In this experiment, the setpoint method described in Sec. 2.2.3 is used for trajectory control in takeoff and landing of the MAV. The desired trajectory is defined to be along the z_W axis of the world frame for both phases. In the takeoff phase, the target height is simply increased in a constant rate until the MAV reaches the desired hovering height. In the landing phase, the setpoint $\mathbf{p}_{set}$ is initialized as $[0,\ 0,\ h_{ini}]^T$. $\mathbf{p}_{set}$ is decreased by a constant vector $[0,\ 0,\ h_s]^T$ after reaching the current setpoint, until the MAV reaches the point $[0,\ 0,\ h_{land}]^T$, where the MAV can be safely landed by blindly powering down the rotors.

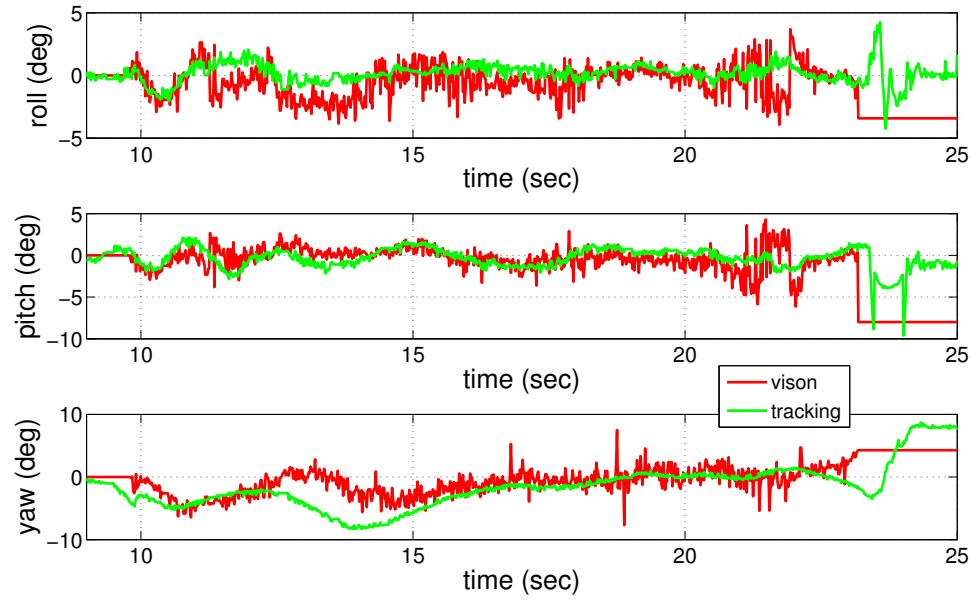
A full trajectory of the MAV during an autonomous taking off, hovering and landing flight is shown in Fig. 3.10. In this flight, it took the MAV about 6 seconds to reach the set point for hovering state. After 5 seconds of hovering around set point $[0,\ 0,\ 1000]^T(mm)$, the MAV started the landing phase, which took about 3 seconds to land on the landing pad. For the start of the takeoff phase, the MAV just ascends with open loop control until it observes the circle of the landing pad. Based on ground truth data of the final states of the MAV, the errors of final landing positions can be found. In our tests of 10 continuous landing flights, the mean errors of the position on x_W, y_W direction and yaw angle are about 24 *mm*, 86 *mm* and 6 *degrees*, respectively. The errors are partially caused by the "blind" range of the camera, during which the camera can not observe the whole landmark circle. This "blind" range also causes the landing of the MAV not soft enough for the mechanism of the MAV landing gear.

Computation Time

The computation time of our vision system during the autonomous flight in Fig. 3.10 is shown in Fig. 3.11. The main part of it comes from the landing pad detection phase, which has an average time cost of less than 10 *ms* and a maximum of about 18 *ms*. Peaks in Fig. 3.11 may be mainly be caused by the performance of the onboard computer, as they did not occur when we tested video logfiles on an off-board PC. While hovering on the set point $[0,\ 0,\ 1000]^T(mm)$, the time cost for geometry computation, including edge detection, ellipse fitting and the 6DOF pose computation, is around 1 ms. When the landing pad gets very close to the camera, its image projection may nearly occupy the whole image, which causes a longer time for geometry computation. When the MAV hovers above the landing pad at a height of about 300 *mm*, the time cost for geometry computation reaches a maximum of about 11 *ms*. The average time cost of the vision algorithm is less than 11 *ms/frame*, making full use of our 60 *fps* camera.



(a)



(b)

Figure 3.10: (a) Position and (b) attitude estimates during an autonomous takeoff, hovering and landing flight.

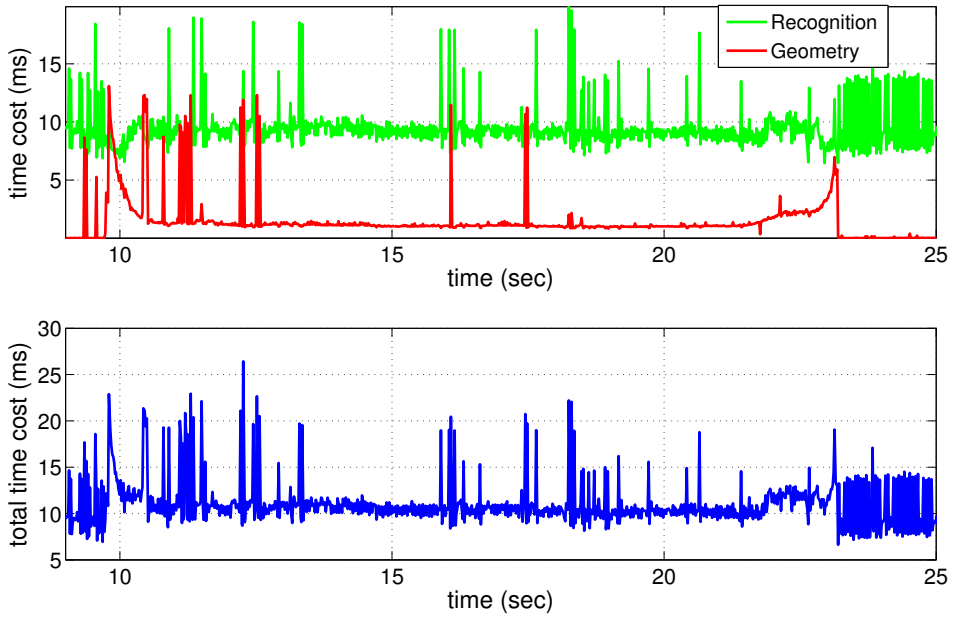
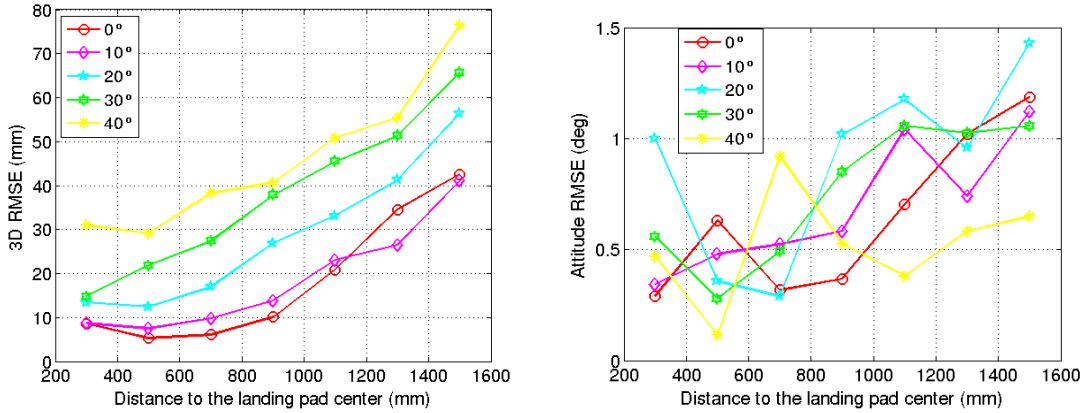


Figure 3.11: Time cost during the autonomous takeoff, hovering, and landing flight.



(a) (b)

Figure 3.12: (a) 3D Position RMSE and (b) attitude RMSE within different distance and attitudes to the landing pad.

5DOF Accuracy Evaluation

We evaluate the 5DOF RMSE of the MAV pose estimated by the onboard vision system at different distances and attitudes to the landmark. The results are shown in Fig. 3.12. We manually fix the MAV above the landmark and record the onboard estimates and ground truth data from the tracking system at each pose for about 10 seconds, i.e. 600 frames, and calculate the RMSEs of these measurements. Since roll and pitch angles have the same effect to pose estimation in our geometric method, we set the pitch angle of the MAV to be approximately zero degree and only change the roll angle. The yaw angle is also set to be approximately zero degree. The optical axis of the camera nearly coincides with the z_W axis, with a small tilt in pitch angle. All onboard pose estimates for this evaluation are obtained within the same system configuration, which means that the systematic deviation with respect to ground truth data is constant throughout this experiment.

Fig. 3.12a shows that the RMSE of the 3D position estimates grows with increasing distance to the landing pad and increasing roll angle. When the roll angle is nearly zero, the two possible solutions mentioned in Sec. 3.4.1 are very close to each other, which may cause our disambiguation method to fail and results in larger deviations. This can explain why the RMSEs for $roll = 0$ may exceed those for $roll = 10$ *degrees*. Fig. 3.12b shows that there is no obvious relationship between the attitude and its RMSE even if it increases to 40 degrees, and its RMSE tends to grow with the increase of the working distance. Attitude estimates of our vision system are overall very accurate, with RMSEs below 1.5 degrees for all tested poses. We do not expect our ground truth data to be much more accurate than this, which could explain why the decrease in accuracy for higher distances and angles in Fig. 3.12b is not so obvious.

3.6 Conclusions

We have presented an onboard vision system that can detect a landing-pad landmark, which consists of a letter "H" surrounded by a circle, from images captured by a monocular camera on an MAV, and determine the 6DOF pose of the MAV relative to the landing pad using a computational projective geometry method.

Our algorithms are computationally efficient enough to process up to 60 frames per second on the onboard computer. We have shown that the whole system produces robust and accurate pose estimates, which were evaluated using an external tracking system. We have used these pose estimates to enable an autonomous MAV to reliably start from, hover above, and land on the landmark, even if the landmark was located within a challenging environment, i.e. in front of a visually cluttered background.

A video demonstrating the autonomous flight of our MAV enabled by the proposed vision system can be found online[1].

[1] `http://www.youtube.com/watch?v=yvyzvttuNsQ`, accessed 12-April-2014

Chapter 4

Visual-SLAM-based Autonomous Landing of MAVs

This chapter presents a monocular visual SLAM solution for MAVs to autonomously search for and land on an arbitrarily textured landing site. The autonomous MAV is provided with only one single reference image of the landing site with an unknown size before initiating this task. The PTAM system is extended to enable autonomous navigation of the MAV in unknown landing areas, in order to search for the landing sites. Furthermore, a multi-scale ORB-feature based method is implemented and integrated into the SLAM framework for landing site detection. A RANSAC-based method is implemented to locate the landing site within the map of the SLAM system, taking advantage of those map points associated with the detected landing site. The efficiency of the presented vision system is demonstrated in autonomous flights of the MAV in indoor and in challenging outdoor environments.

Large parts of this work have been pre-published in Yang, S. *et al.* (2013a, 2014a).

4.1 Introduction

Visual SLAM has brought more flexibility to autonomous navigation of MAVs. It can provide accurate pose estimates of MAVs for the control of autonomous flight in unknown environments (Achtelik *et al.*, 2011). This can be achieved by either stereo visual SLAM, RGB-D SLAM, or monocular visual SLAM. For an MAV with limited payload, monocular visual SLAM can be a better choice, considering that monocular vision is more compact and less computationally intensive than stereo vision, and a monocular camera is lighter in weight than an RGB-D camera. In fact, monocular visual SLAM is especially well-suited for the autonomous landing task of an MAV: The problem of slow scale drift, which is inherent in every purely monocular vision system caused by the unobservability of the metric-scale factor, can hardly cause much effect in the relatively small areas where the MAV is expected to land.

In this chapter we show that the rich information provided by a visual SLAM system can actually benefit both the real-time detection and the localization of a designated landing site. Considering the limited computational power that is typically available

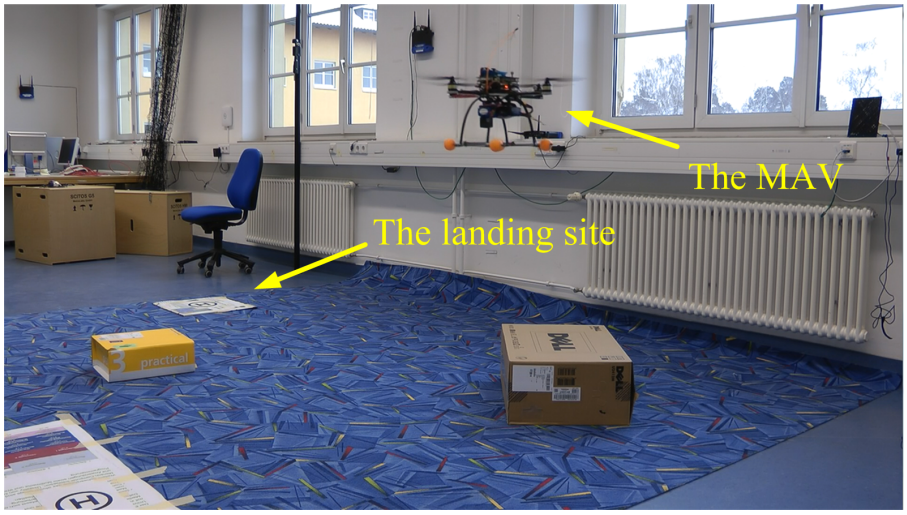


Figure 4.1: Our MAV navigating autonomously to search for a textured landing site.

onboard of an MAV, those processes are normally difficult to be performed in parallel with onboard-vision-based autonomous navigation. We achieve autonomous flight of our MAV by implementing a monocular visual SLAM system with near-constant time cost, while simultaneously detecting an arbitrarily textured landing site based on ORB features (Rublee *et al.*, 2011) and estimating its global position. The resulting monocular vision system enables the MAV to autonomously search for the landing site in unknown environments (as shown in Fig. 4.1), and then land on it once it is found.

The rest of this chapter is organized as follows. After a brief review of the related work on vision systems for autonomous landing of UAVs/MAVs in Sec. 4.2, we introduce the monocular visual SLAM system for autonomous navigation of our MAV in Sec. 4.3. Then the landing site detection and pose estimation algorithm is presented in Sec. 4.4. We evaluate the proposed visual-SLAM-based algorithm for autonomous landing of the MAV in Sec. 4.5, and give further experiment results and discussions on an outdoor experience of our MAV in Sec. 4.6. Finally, we conclude this chapter and provide more discussions in Sec. 4.7.

4.2 Related Work

Early work on visually guided autonomous flight of UAVs/MAVs mainly focuses on the autonomous landing problem, while classical systems combining GPS and Inertial Navigation System (INS) usually being implemented for long range and high altitude missions. Computer vision methods do not depend on external signals. Moreover, they fit especially well to those cases in which precise position control relative to other objects is required, e.g. for the landing tasks of UAVs/MAVs. Thus, they are highly appreciated

for researches towards full autonomy of UAVs/MAVs.

Vision systems for autonomous landing of UAVs/MAVs can mainly be divided into two groups: The first one is for safe landing of an UAV/MAV on a suitable area (Garcia-Pardo *et al.*, 2002; Cesetti *et al.*, 2010), which uses visual methods to detect a place good enough for safe landing. The second group is for landing an UAV/MAV on a predefined artificial landmark (Saripalli *et al.*, 2003; Merz *et al.*, 2006; Lange *et al.*, 2009; Wenzel *et al.*, 2010a), which requires precise pose estimates of the UAV/MAV relative to the specific marker, as have been reviewed in Sec. 3.2. The applications of those systems can be found on large scale helicopters with a high payload to MAVs with a very limited payload. Garcia-Pardo *et al.* (2002) presented a strategy to find a safe landing area by searching the image for a circular area in which all the pixels have a level of contrast below a given threshold. The vision system developed in Cesetti *et al.* (2010) allows a remote user to define a target area as a new waypoint or as a landing area for an UAV, based on a high resolution aerial or satellite image. In that work, a SIFT-based image matching algorithm is implemented to find the natural landmarks, and an optical-flow-based method is used for the detection of a safe landing area.

Although a number of visual-SLAM-based systems have been developed for autonomous flight of MAVs (Achtelik *et al.*, 2011; Weiss *et al.*, 2012; Shen *et al.*, 2013a), none of the those works have addressed the problem of visual SLAM in parallel with landing site detection. Related work on combining SLAM and object recognition in a vision system can be found in Castle *et al.* (2010) and Castle and Murray (2011), which dedicated to the field of augmented reality (AR). In Castle *et al.* (2010), monoSLAM (Davison, 2003) and SIFT features are used to recognize and localize objects within a 3D map built by a wearable camera. Each object is located from a single view using its known size, and then its location is fed back to the EKF framework of the SLAM system. Further in Castle and Murray (2011), a multiple-map and multiple-camera extension to the PTAM algorithm is implemented to replace monoSLAM. The location of an object is determined by triangulation from the locations of SIFT features matched across different keyframes.

To land an MAV on an arbitrary landing site, we implement an ORB-feature-based method for landing site detection, running in parallel with visual SLAM. Furthermore, based on the existing map points, we show that it is possible to robustly estimate the 3D global pose of the detected landing site even if the size of it is unknown, without re-triangulation from the landing site features as Castle and Murray (2011) did. Since the pose estimates of our MAV for position control are provided by a SLAM system, high frequency landing site tracking and pose estimation become unnecessary, while still maintaining the final landing performance. This is of advantage to marker-based methods or conventional template-based methods, which only consider the relative pose of an MAV to a landing site. An example of such a template-based method can be found in the work in Mondragón *et al.* (2010), which estimates the 3D pose of a camera based on the homographies computed for a reference image of a planar pad with a known size.

4.3 Monocular Visual SLAM for Autonomous MAVs

The visual SLAM framework we use for autonomous navigation of our MAV is based on PTAM (Klein and Murray, 2007), as briefly described in Sec. 2.4.1. It provides MAV-pose estimates as control input for the navigation of the MAV. In order to overcome the lack of a scale factor, we implement an automatic initialization method for PTAM based on the work in Chapter 3, which can cope with cluttered environments and provide accurate pose estimates of a camera. Moreover, we modify the mapping thread of PTAM to achieve a near-constant time cost during the navigation of the MAV.

4.3.1 Using PTAM in near-constant time

Bundle adjustment, including global bundle adjustment (full bundle adjustment) and local bundle adjustment (window bundle adjustment), is used for map refinement in PTAM. It is also the most computationally intensive task in PTAM.

Since the global bundle adjustment adjusts poses of all keyframes and positions of all map points, it is so computationally intensive that one may stop the mapping thread from adding enough new keyframes to facilitate successful pose tracking. To enable PTAM to achieve a near-constant time cost, we abandon the global bundle adjustment and only retain its local bundle adjustment, which adjusts the poses of keyframes and positions of map points in a relatively small window.

Meanwhile, we still keep the complete map during the exploration of the SLAM system, rather than using PTAM as a visual odometry like in Schauwecker *et al.* (2012). To keep a complete map is important for the SLAM system in two aspects: First, it allows more accurate pose tracking of the MAV. We do not consider loop closures in the SLAM system in this chapter since we expect the landing site locates in a relatively small area. In this case, discarding old keyframes that are no longer considered for local bundle adjustment from the map might result in considerable pose drift. Such pose drift will also affect the accuracy of locating the landing site. Second, map points having been triangulated in those old keyframes may be useful for the landing site location. All map points, which are potentially visible by the camera, would facilitate the landing site location, as will be presented in Sec. 4.4.

4.3.2 Automatic initialization of the SLAM system

Since there exists a common scale ambiguity inherent in monocular camera systems, PTAM naturally requires additional metric scale information. PTAM was originally intended for augmented reality applications (Klein and Murray, 2007). Thus, an accurate metric scale was not necessary for its original implementation, and only a coarse scale estimate is applied to the triangulation of its initialization phase. We deal with this initialization issue by using an individual visual initialization module as presented in Chapter 3,

Figure 4.2: A scene when the PTAM-based SLAM system is initialized. *Top left*, the original image; *Top right*, the detected artificial landmark, labelled with an orange cross; *Bottom left*, vision features at different pyramid levels of the image marked in different colors (from the bottom to the top level: blue, green, cyan and magenta) after non-maximum suppression; *Bottom right*, initialized map points of the SLAM system, marked in red.

which can robustly estimate the camera pose based on the image projection of a landing-pad landmark. Using this method, we can achieve accurate automatic initialization of PTAM during the takeoff phase of our MAV, without requiring any additional sensors. Before the SLAM system is initialized, the autonomous takeoff of the MAV is enabled by the initialization module. Fig. 4.2 shows an example scene and related results during the initialization of the SLAM system using this method.

Once we obtain a pose estimate of the camera with a height larger than a threshold h_i using the initialization module, this pose estimate and the image associated with it are both sent to a queue in the SLAM system for the initialization. If more than a minimum number of FAST features after non-maximum suppression are detected in all four pyramid levels of this image, we use them to initialize the map of the SLAM system, and send a message to the initialization module to terminate its operation. We obtain the 3D positions of those feature points by assuming that they all lie on the ground plane and by projecting them from their image coordinates to the $z = 0$ plane in the world frame $\mathcal{W}$.

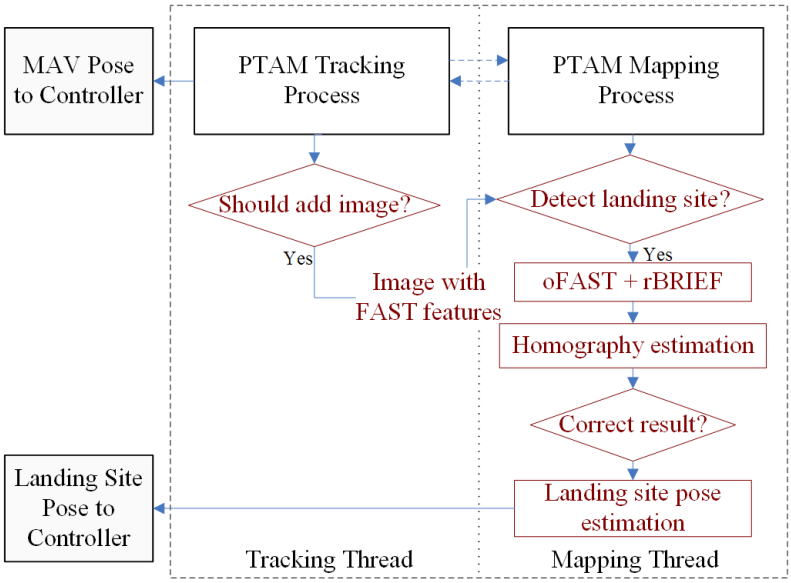


Figure 4.3: Landing site detection and pose estimation integrated in the PTAM-based SLAM system. Operations related to our landing-site detection and localization are marked in red.

In this way, the world frame defined in the SLAM system coincides with $\mathcal{W}$.

4.4 Landing Site Detection and Pose Estimation

To search for an arbitrary landing site during autonomous navigation of our MAV, a feature-matching-based object detection scheme is implemented in this section. Using one preset reference image of the designated landing site, a set of feature matches between the reference image and the currently visible scene can be established. Then the landing site is detected by using a robust RANSAC-based method to estimate the corresponding homography and eliminating false estimates. Because it is possible that some of the map points produced by the SLAM system are associated with the matched features, we can use the 3D position estimates of those map points to estimate the global 3D pose of the landing site. An advantage of our method is that it does not require absolute scale information of the landing site. The above process is integrated in the mapping thread of the SLAM system, as depicted in Fig. 4.3.

4.4.1 Brief overview of the ORB features

Rublee *et al.* (2011) proposed the ORB (Oriented FAST and Rotated BRIEF) features based on the FAST keypoint detector (Rosten and Drummond, 2006) and the BRIEF

descriptor (Calonder *et al.*, 2010), both of which are known for their high computational efficiency.

BRIEF uses a binary string constructed from a set of binary intensity tests as an efficient point feature descriptor. Because BRIEF is not designed to be aware of the orientation of a feature point, it is notably lacking rotation invariance (Rublee *et al.*, 2011), which is, however, important for feature-matching-based object detection. To cope with this issue, Rublee *et al.* (2011) proposed to compute an orientation component for each FAST interest point (oFAST) by using the so-called intensity centroid, which is computed from image moments. BRIEF descriptors for those points are then efficiently rotated according to the orientation component, and thus form the steered BRIEF descriptor. Furthermore, a learning method is developed for choosing a good subset of binary tests, in order to increase the feature variance and reduce correlation among the binary tests, both of which are important for a discriminative feature. The resulting descriptor is named rBRIEF.

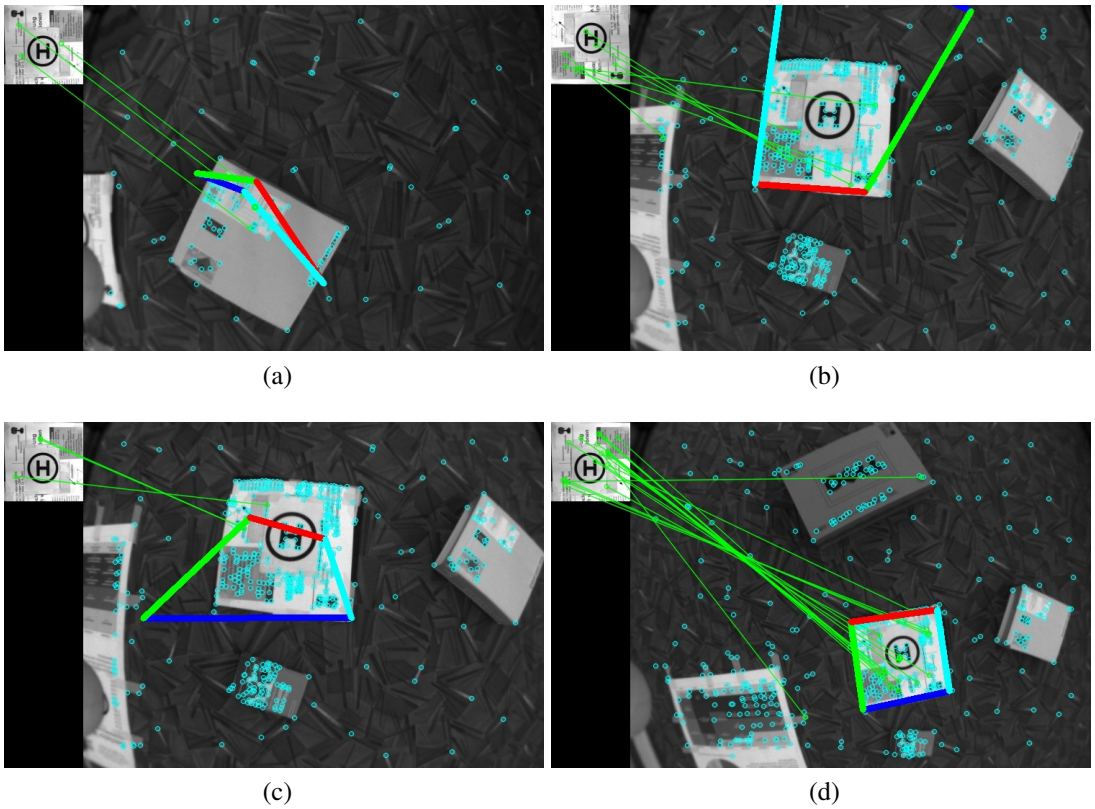
4.4.2 Applying multi-scale ORB to the SLAM framework

We chose ORB as the feature descriptor for our landing site detection because of its low time cost and high discrimination capability for feature matching. ORB achieves scale invariance by applying the FAST detector to a scale-space pyramid of the original image. Note that in the tracking thread of PTAM, FAST points have been detected in four levels of pyramid images of the current scene. Thus, it is straightforward for us to use those points on the multiple image levels for further feature description. We chose such a multi-scale scheme in order to avoid the computation of further pyramid levels, as a compromise between matching performance and time cost. In the mapping thread, we compute orientation components of the FAST points to obtain oFAST features, and use rBRIEF for feature description. We perform both of these operations individually at pyramid level 0 and 1, resulting in two sets of descriptors $\{D_i^c | \, i = 0, 1\}$, each with a size n_i. We discard higher pyramid levels, since at higher levels, a landing site appears too small for us to obtain useful features for matching. For the reference image of the landing site, the number of pyramid levels and the scale factor for producing the pyramid images can vary according to the requirements of scale invariance and available computation time. In our work, we apply a three level pyramid with a scale factor of 1.2 to the reference image, obtaining the reference descriptor sets $\{D_i^r | \, i = 0, 1, 2\}$. A Gaussian blur is applied to each pyramid level before feature detection.

4.4.3 Landing site detection by feature matching

Feature Matching

We use a standard feature matching scheme to obtain a set of good feature matches from $\{D_i^r | \, i = 0, 1, 2\}$ to $\{D_i^c | \, i = 0, 1\}$, for estimating the homography H_{rc} between the reference



(a) (b)

(c) (d)

Figure 4.4: Examples of homography estimation results shown in one pyramid level. After eliminating false estimates, only the one in (d) will be regarded as a correct homography estimate, and a true positive detection of the landing site.

image and the current image of the landing site. For finding all possible matches, we employ a brute-force matcher without cross checking, which is implemented in OpenCV (Bradski, 2000). It finds the k descriptors with the closest normalized Hamming distances in $\{D_i^c | \, i = 0, 1\}$ for each descriptor in $\{D_i^r | \, i = 0, 1, 2\}$. Similar to Lu and Zheng (2010) and Lowe (2004), we consider a match between a reference descriptor and the corresponding descriptor with the closest distance to be valid, if the ratio of the closest to the second closest distance is smaller than a threshold T_r.

Homography Estimation

As the ORB feature is applied at different individual pyramid levels of the current image, we project all matched feature points to the source pyramid level (level 0) to calculate the homography. The homography H_{rc} is estimated by using RANSAC, and then further refined by using the Levenberg-Marquardt method to minimize the image projection

errors. We limit the iterations in RANSAC to a relatively small number, in order to make this process more efficient. However, this may cause a higher false negative rate, i.e. failing to detect the landing site which is actually located in the current scene of the camera. Since our consecutive landing site pose estimation can cope well with a high false negative detection rate, which will be explained in Sec. 4.4.4, we opt for a higher processing performance.

Eliminating False Estimates

The reference image forms a quadrilateral Q_r when transformed to the current image frame with H_{rc}. Some examples of the homography estimates we received can be found in Fig. 4.4. False homography estimates may still occur due to false matches or too few correct matches of features. We dramatically eliminate those false estimates by evaluating some basic properties of quadrilateral Q_r : First, it is required to be a convex polygon (Fisher, 1994). Second, all four vertexes of it should have a reasonable relative distances to their centroid and to each other. This will eliminate estimates like the ones shown in Fig. 4.4b and Fig. 4.4c: Although the reference image can be found in the current image, we reject this homography estimate since we will not achieve a correct pose estimate of the landing site according to it. Third, the number of matched features that are inside of this quadrilateral should be larger than a threshold n_q. We determine whether a point is located inside a polygon by using a crossing-number method (Sunday, 2014).

4.4.4 Locating the landing site within the map

After the landing site has been detected in the current image by using the above method, we locate its 3D position in the world coordinate frame. For this task we take advantage of the environment map produced by the SLAM system, which can consist of a large number of map points. Doing this provides us with much more tolerance to a high false negative detection rate: Even if the landing site is not tracked at camera frame rate, its final position estimate will be hardly affected, since it should retain a static position in the environment. Thus, the mapping thread can flexibly decide the time intervals of adding new image frames for landing site detection. Furthermore, using the map points ensures that only discriminative image features are used for locating the landing site.

We first project all map points to a rectified image frame based on their 3D positions and the calibrated camera model. Again, we use a crossing number method to check whether a projected map point is located within the quadrilateral Q_r or not. Those points inside Q_r form the map points subset $\{p_l\}$.

If the size of $\{p_l\}$ is larger than a threshold n_{lmin}, a RANSAC-based method is applied to the points in $\{p_l\}$ to estimate the dominant plane P_d of the landing site. We perform this step in a similar fashion as in Klein and Murray (2007): Many sets of three points are randomly selected to form a plane hypothesis, while the remaining points are tested for

consensus. The winning hypothesis is further refined using the consensus set, resulting in the detected plane normal $\mathbf{n}_p$. Together with the mean 3D coordinate value of all consensus set points $\mathbf{x}_m$, this normal defines the plane P_d. Once an estimate for P_d is achieved, we use its corresponding measurements $\mathbf{n}_p$ and $\mathbf{x}_m$ as the initial guess for the RANSAC procedure when evaluating the next image frame. Thus, a much smaller threshold for the number of RANSAC iterations can be applied, which further reduces the time cost of the algorithm.

The location of the landing site can be calculated by projecting the quadrilateral Q_r to the plane P_d. We define $\mathbf{x}_{p_i}, i = 0,1,2,3$, as the four vertices of Q_r, which are the image projections of the four corners (P_i) of the landing site with the corresponding world coordinate positions $\mathbf{x}_{w_i}$. After projecting $\mathbf{x}_{p_i}$ to the normalized image frame with rectified lens distortions, we obtain the normalized coordinates $\mathbf{x}_{n_i} = [x_{n_i}, y_{n_i}, 1]^T$. In the camera frame, we then have $\mathbf{x}_{c_i} = s \cdot [x_{n_i}, y_{n_i}, 1]^T$, with s being an undetermined scale factor. Thus, in the world frame we have

$$\mathbf{x}_{w_i} = s \cdot R_{WC} \cdot \mathbf{x}_{n_i} + \mathbf{t}_{WC}, \tag{4.1}$$

with R_{WC} and $\mathbf{t}_{WC}$ being the camera pose in the world frame obtained by the tracking thread of the SLAM system. Since P_i is located on the plane P_d, we have

$$\mathbf{n}_p \cdot (\mathbf{x}_{w_i} - \mathbf{x}_m) = 0. \tag{4.2}$$

From Eq. (4.1) and (4.2), we can calculate $\mathbf{x}_{w_i}$. The landing site location is then obtained as $\mathbf{x}_k = \frac{1}{4} \sum_{i=0}^{3} \mathbf{x}_{w_i}$, where k is the current image frame index.

We further refine the landing site location by integrating m successful estimates of $\mathbf{x}_k$. Estimates with a large difference to $\mathbf{x}_{Lm} = \frac{1}{m} \sum_{k=0}^{m-1} \mathbf{x}_k$ are assumed to be outliers. The mean value of the remaining inlier estimates is then assumed to be the final landing site position estimate $\mathbf{x}_L$.

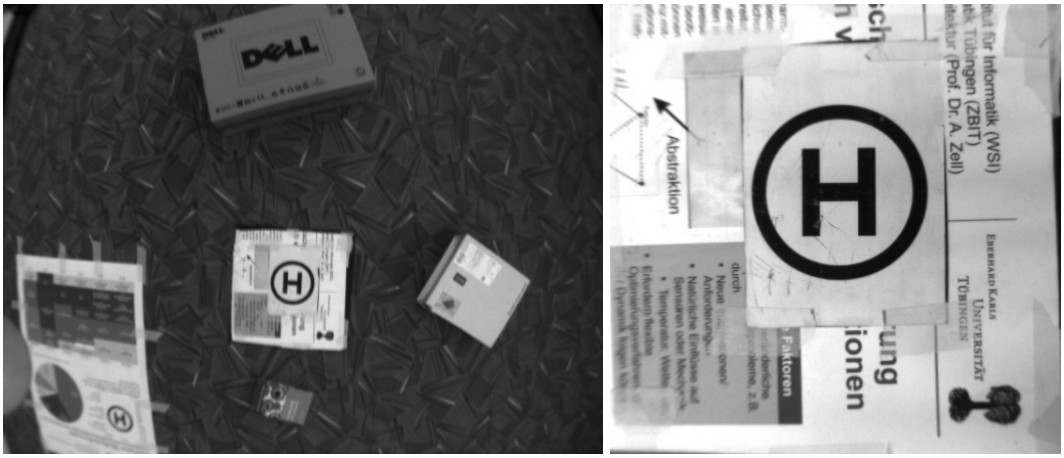
4.5 Experiments and Results

4.5.1 Experimental setup

We implemented our software in several modules (nodes) using the open source Robot Operating System (ROS) (Quigley *et al.*, 2009) on Ubuntu Linux 12.04. ROS provides the infrastructure for logging all interested onboard data and for efficient communication among our different vision modules: one ROS node for the camera driver, one node for the initialization module, and another one for the SLAM system with landing-site detection and pose estimation integrated in it. Images and other onboard sensor data are recorded into ROS-bag files if necessary.

RMSE	poster	book	mail Pack.	PC Pack.
x-y	22	15	34	43
z	1	8	6	2
3D	22	17	35	43

Table 4.1: RMSEs (*mm*) of position estimates for different landing sites during a manual flight.



(a) (b)

(c) (d) (e)

Figure 4.5: (a) A scene from the MAV, (b), (c), (d) and (e) are the reference images of the poster landing pad (size 500×500, height 4.5), the book (size 246×175, height 33), the mail package (size 380×335, height 140) and the computer package (size 650×435, height 235), respectively. All size and height are measured in *millimeters*.

4.5.2 Landing site position estimation results

We evaluate the landing site detection and position estimation results by processing a video logfile taken during a manual flight of our MAV above different objects with planar surfaces which are used to represent different landing sites: a poster pad, a book, a mail package, and a computer package. Each of them has different texture features. Moreover, they are different in size and height. We control the MAV to take off from another pad nearby those objects, such that our SLAM algorithm can be initialized by this pad as described in Sec. 4.3.2. Fig. 4.5a shows an image taken by the camera on the MAV during this flight. Reference images of those landing sites are captured by manually holding the MAV above them in different illumination conditions, as shown in Fig. 4.5.

We process the same recorded video sequence four times, selecting a different reference image for landing site detection each time. The identical MAV trajectory estimated by the visual SLAM algorithm is shown in Fig. 4.6e and 4.6f. The origin of the world frame is attached to the center of the artificial landmark from which our SLAM system is initialized. The poster pad provides a total number of 481 ORB features in all three pyramid levels, the book 69, the mail package 153 and the computer package 97 features. Despite their differences we mentioned above, they can be correctly detected and located. The RMSEs of their 3D position estimates are listed in Tab. 4.1.

Since position distributions of the detected landing sites are similar, we only present the results for the book and the computer package, which have the overall smallest and largest RMSEs, respectively. Fig. 4.6a and 4.6b show the distribution of the position estimation results for the book, and Fig. 4.6c and 4.6d show that of the computer package. The position estimates are projected to the $x_W - y_W$ and $x_W - z_W$ planes of the world frame. The systematic errors to the position estimates of different landing site are in the order of few centimeters, which is acceptable for autonomous landing of MAVs. The few estimates with relatively large errors do not affect the autonomous landing since they can be excluded after a refinement process as described in Sec. 4.4.4. In Fig. 4.6e, we mark the height of the detected poster pad with black crosses, if it is detected at the corresponding time. Similarly, in Fig. 4.6f, we mark the MAV yaw angle estimates when the poster pad is detected. They show that the poster pad is detected when the MAV is at different positions and yaw angles.

4.5.3 Autonomous navigation and landing results

Using the setpoint method

In this experiment, we use the poster pad shown in Fig. 4.5b as the target landing site. Our MAV autonomously navigates using the setpoint method described in Sec. 2.2.3 for trajectory control, in order to search for the landing site and finally land on it. Once an initial position of the landing site $\mathbf{x}_{lini}$ is estimated, we change the setpoint to be above this position, keeping our searching height h_s. After the final refined location of the

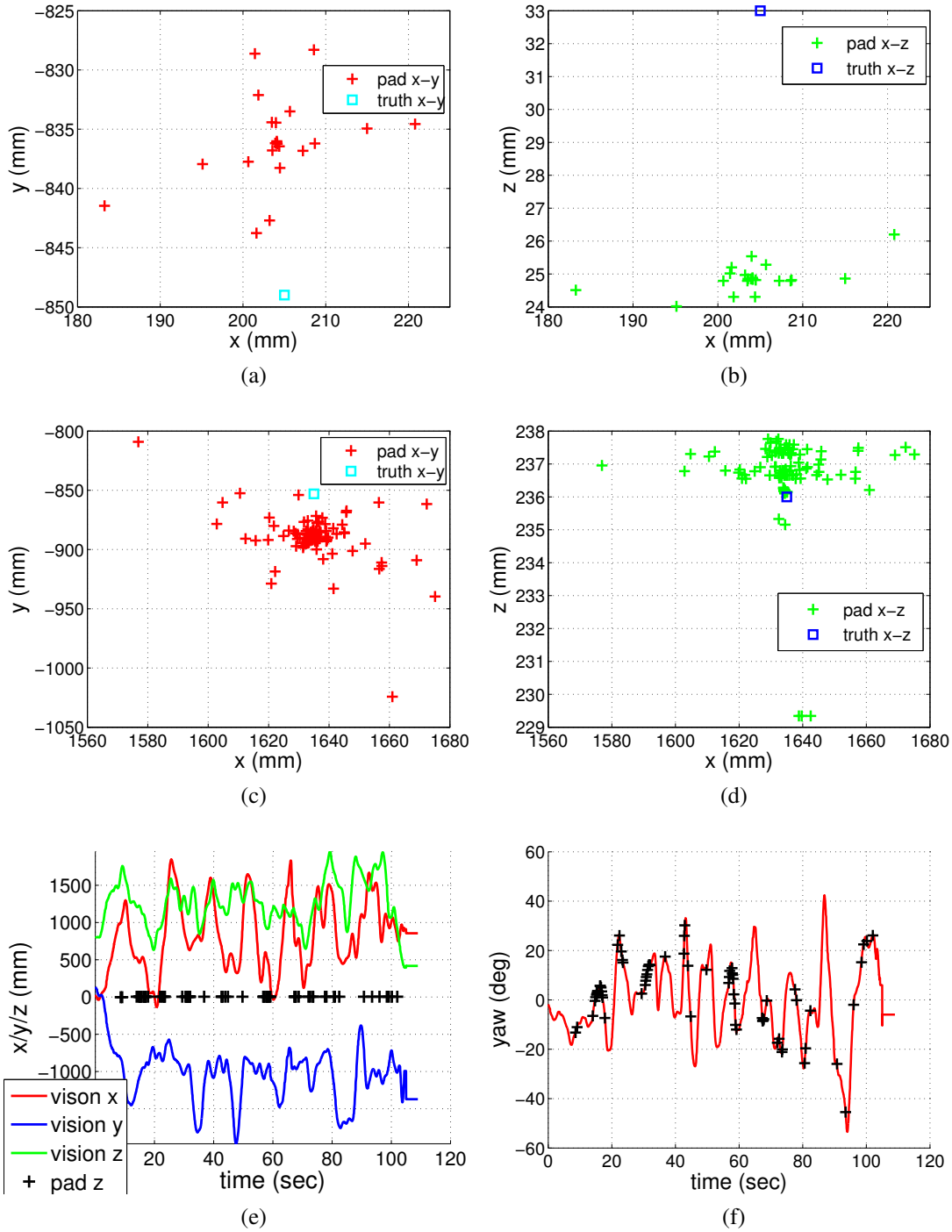


Figure 4.6: (a), (b) Position estimates for the book, on $x_W - y_W$ and $x_W - z_W$ plane respectively, and (c), (d) for the PC package. (e) Trajectory of the MAV, and (f) the corresponding yaw angle estimates (a cross is marked if the landing site is detected at the corresponding time).

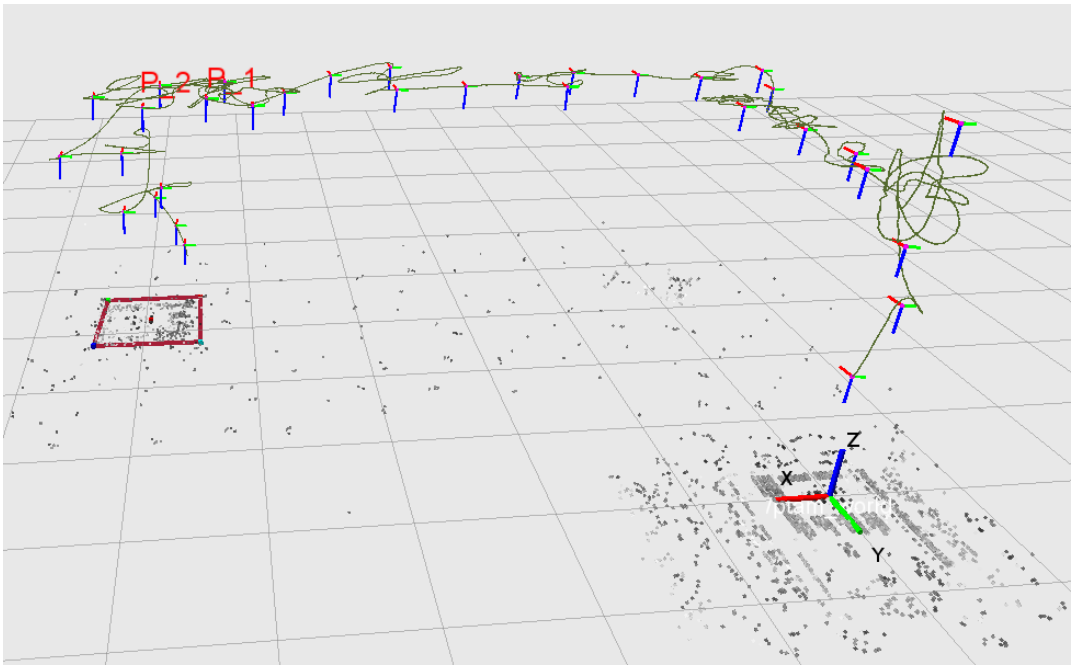


Figure 4.7: The built map and MAV trajectory during a searching and landing task using the setpoint method for trajectory control.

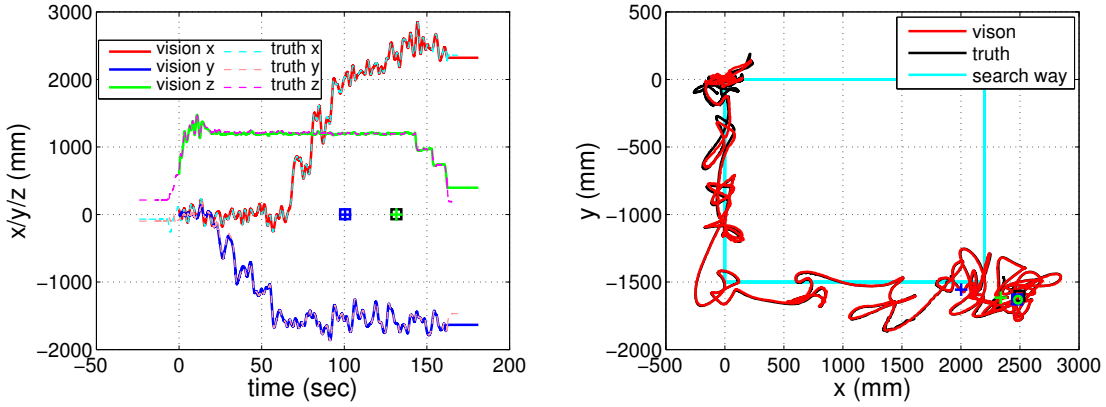
landing site $\mathbf{x}_L = [x_l, y_l, z_l]^T$ is determined, we define the end of the searching path to be $[x_l, y_l, h_s]^T$. Finally, the desired height of the setpoint is decreased until the MAV reaches a predefined landing height h_l where it can steadily shut down its motors to finish the landing process.

The trajectory of this searching and landing task, as estimated by our onboard SLAM system, is shown in Fig. 4.7. The map points are triangulated and refined when new keyframes are added. The world frame is indicated as the RGB axes (corresponding to $x_W - y_W - z_W$ axes) that lie on the ground grids in Fig. 4.7. The pose of each keyframe has been depicted as small RGB axes.

The predefined searching path should depend on the expected complexity of the landing area. Here, a simple rectangular searching path is defined, as depicted in Fig. 4.8b with a cyan rectangle. We choose four setpoints evenly distributed on each edge of the rectangle. The MAV navigates along this searching path after takeoff and initialization of the visual SLAM system. At each corner of the rectangle, the MAV is commanded to hover for 3 seconds to maintain better stability. The parameters for the setpoint method (see Sec. 2.2.3) are listed in Tab. 4.2. The landing site is detected when the MAV is at the position $P_1 = [2.001, -1.556, 1.197]^T(m)$, relative to the starting position. When the landing site is detected at the MAV position $P_2 = [2.337, -1.616, 1.201]^T(m)$, the detection process stops, and the landing site pose is further refined using all the previous

Parameter	h_s	h_l	d_s	ψ_s	t_s
Value	1.2m	0.3m	0.2m	15deg	0.2s

Table 4.2: Parameter setup for the trajectory control using the setpoint method.



(a) (b)

Figure 4.8: (a) MAV position estimates from our vision system on x_W, y_W, and z_W axes. (b) MAV trajectory projected to the $x_W - y_W$ plane. The initial and final position estimates of the landing site and the associated MAV poses are also marked.

obtained pose estimates. The final refined landing site position is visualized as a bold colored quadrilateral in Fig. 4.7. Fig. 4.8 is the resulting trajectory on different axes of the world frame. It shows that the MAV needs to hover for a certain period of time to satisfy the constraint of reaching each setpoint, and then moves to the next one.

Fig. 4.8 also shows that the above MAV trajectory fits well with the ground truth data, which proves the accuracy of both the SLAM system and its initialization module. Tab. 5.1 lists the RMSE of the on-board MAV-pose estimates when compared with the ground truth data. Position P_1 is marked with a blue cross in Fig. 4.8b , P_2 with a green cross. The initial position estimate of the landing site on the $x_W - y_W$ plane is marked with a blue square, and the final refined estimate with a green circle. Both position estimates are very close to the ground truth data, which is marked with a black square. The blue and green crosses in Fig. 4.8a show the initial and final height estimate, compared to the ground truth height marked with squares. With the landing site size being $500 \times 500(mm)$, the initial and final position estimation error is $[-19, -26, -6]^T(mm)$ and $[-11, -27, -5]^T(mm)$, respectively.

Table 4.3: MAV pose estimation RMSEs of the whole trajectory, with position errors in *millimeters* and attitude errors in *degrees*

	x	y	z	3D	roll	pitch	yaw
RMSE	8.6	13.6	14.3	21.6	1.04	0.85	1.49

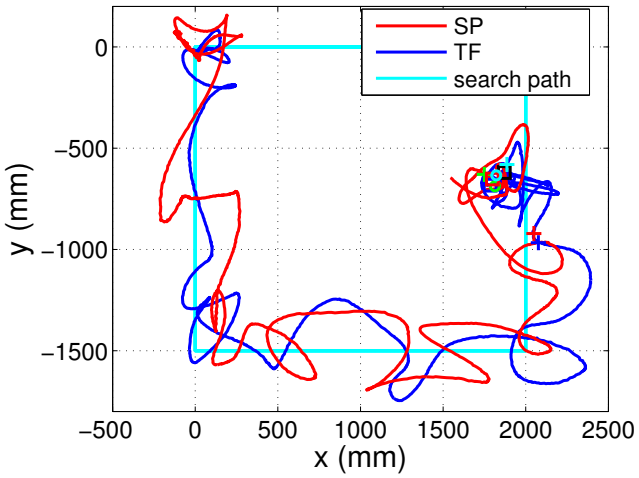


Figure 4.9: MAV trajectories projected to the $x_W - y_W$ plane, when using the two different methods for navigation.

Using trajectory-following method

In this experiment, we compare the efficiency of the trajectory-following method (TF, described in Sec. 2.2.3) with the setpoint method (SP) for trajectory control of our MAV. The poster pad is again used.

The trajectories of the MAV when using both methods are shown in Fig. 4.9. The PID parameters of the low level position and attitude controllers are the same in both cases. Trajectory controller parameters different from those in the previous section are shown in Tab. 4.4. Only when we use the method of trajectory following, we can explicitly set the MAV forward speed. However, we should note that it does not directly control the MAV speed. Instead, the forward speed will be fed to the position controller of the MAV.

When we calculate the RMSEs of the actual flight trajectory with respect to the predefined searching path on $x_W - y_W$ plane, position errors along the tangent direction of the searching path are ignored. The resulting RMSEs during the period of searching for the landing pad are listed in Tab. 4.5. It shows the two methods result in similar precision for trajectory control under the parameter setup in Tab. 4.4, with the setpoint method performing a bit better. However, the trajectory-following method performs extremely well on yaw angle control. More importantly, this method is much more efficient regarding the flight time, which is 32 seconds to obtain the initial pose of the landing pad. On the contrary, the setpoint method spends 52.8 seconds to achieve this, even though we

Parameter	h_l	t_s	v_s
Value	0.4m	0.1s	0.35m/s

Table 4.4: Parameter setup for trajectory control using the two different methods.

	$x-y$	z	yaw
RMSE1	134.8mm	18.3mm	3.6deg
RMSE2	163.5mm	15.3mm	0.03deg

Table 4.5: MAV trajectory control RMSEs, with RMSE1 for the set-point method, and RMSE2 for the trajectory-following method.

have set t_s smaller than that in Sec. 4.5.3 to speed up the searching process. To improve the trajectory control precision, which will result in less efficiency in forward speed on the other hand, we suggest to distribute more setpoints along the predefined path and a relatively larger t_s in the setpoint method, while setting a smaller forward speed in the trajectory-following method.

When using the setpoint method, the detected landing pad positions and the corresponding MAV positions are marked in Fig. 4.9 in the same way as in Fig. 4.8b. For the results of using the trajectory-following method, the initially and finally detected landing pad positions are marked with a red square and cyan circle, and the corresponding MAV positions are marked with red and cyan crosses, respectively.

4.6 Outdoor Experience

4.6.1 Outdoor experiment

In this experiment, we want to examine the robustness and extensibility of our vision system when working in outdoor environments. The experiment was carried out on a sunny midday with moderate wind, and the scenario is challenging in three aspects: First, the MAV flies above a meadow, as shown in Fig. 4.10. Thus, the first challenging aspect is that the environment is filled with highly self-similar visual features from grass, which will be discussed in Sec. 4.6.2. Second, there is wind strong enough to bring disturbance to the pose controller of our MAV, which makes it a good scenario for testing the robustness of the controller. Third, another challenge is introduced by the grass and other plants below the MAV: When the MAV flies above them in low altitude, their physical shape will be obviously changed by the wind produced by MAV propellers. The effect introduces more noise to our visual SLAM system.

The landing site we defined is a piece of package paper with a size of $0.95m \times 0.95m$. The reference image as shown in Fig. 4.11 was taken on another day by the same camera on our MAV. It provides a total number of 59 ORB features in the three pyramid levels. The trajectory-following method is used for trajectory control. Parameters for trajectory control are set in the same way as in Sec. 4.5.3, except that we define d_s to be $0.3m$. However, we define a longer searching path as $5m \times 5m$ with a height of $3.5m$.

Figure 4.10: The scenario for the outdoor experiment.

Figure 4.11: The reference image of the landing site in the outdoor environment.

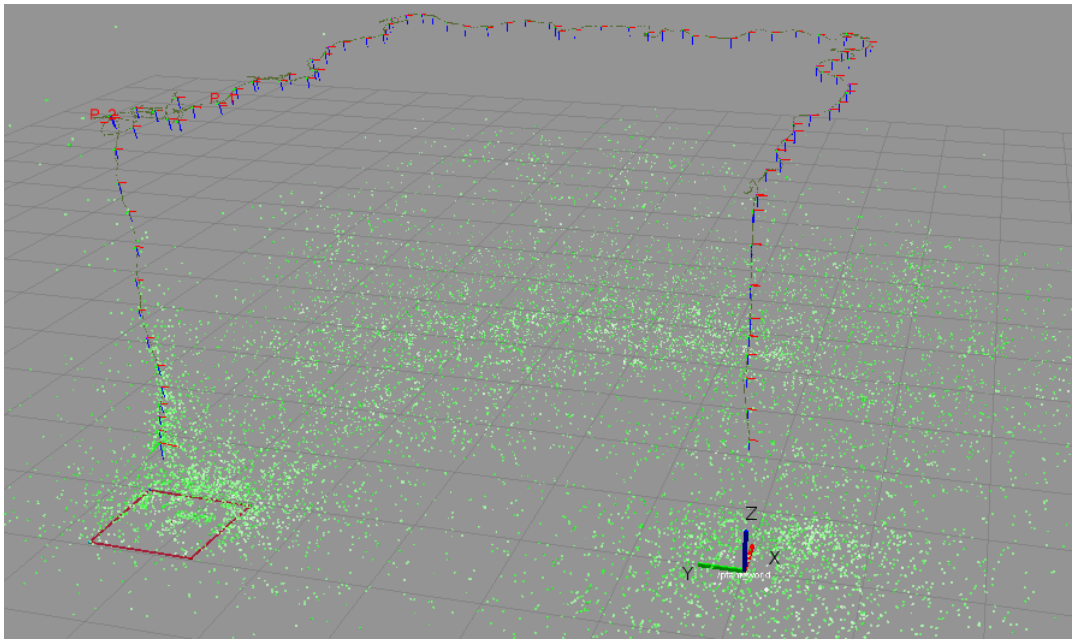


Figure 4.12: Visualization of the built map and MAV trajectory during the outdoor searching and landing task. Map points are marked in green.

	$x-y$	z	yaw
RMSE	192.0mm	62.4mm	0.04deg

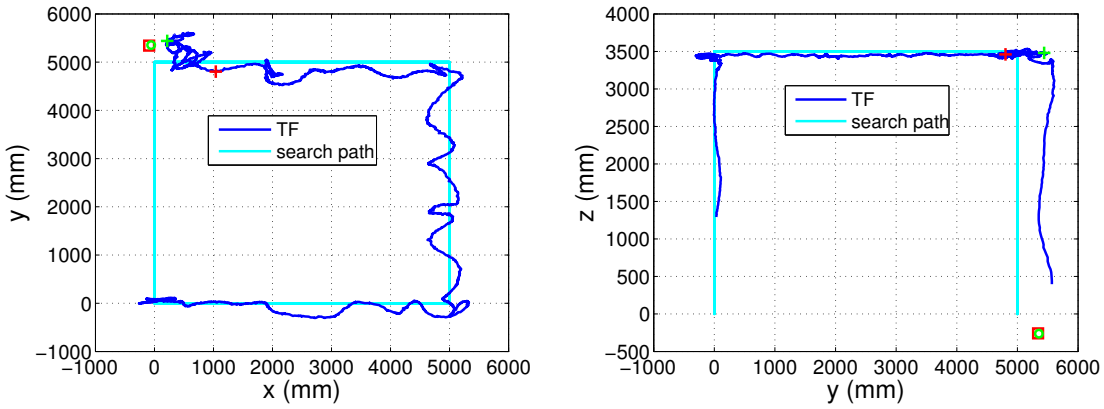
Table 4.6: MAV trajectory control RMSEs for the outdoor flight.

Fig. 4.12 and Fig. 4.13 show the built map and MAV trajectory of an outdoor flight. Tab. 4.6 shows the RMSEs of the actual flight trajectory with respect to the predefined searching path. We find that the flight performance is not much worse than that in the indoor experiments despite the challenges we mentioned earlier. One reason for such results is that the MAV now flies much higher than in the indoor experiments, which leads to much less ground effect on the MAV.

Fig. 4.14 shows a scene when the SLAM system is initialized in this outdoor experiment. Although this time the assumption of all features lying on the same plane is no longer strictly true, it will not bring obvious effects to the pose tracking process, and those inaccurate map points produced during initialization will be later adjusted by local bundle adjustment.

4.6.2 Discussions

We mentioned three challenging issues in Sec. 4.6.1, which make it difficult for our MAV to autonomously navigate in outdoor complex environments. Those challenges may finally cause pose tracking failures. Among them, the self-similarities of visual features



(a) (b)

Figure 4.13: (a) MAV trajectory projected to $x_W - y_W$ plane, and (b) to the $y_W - z_W$ plane. The initial and final position estimates of the landing site and the associated MAV poses are also marked.

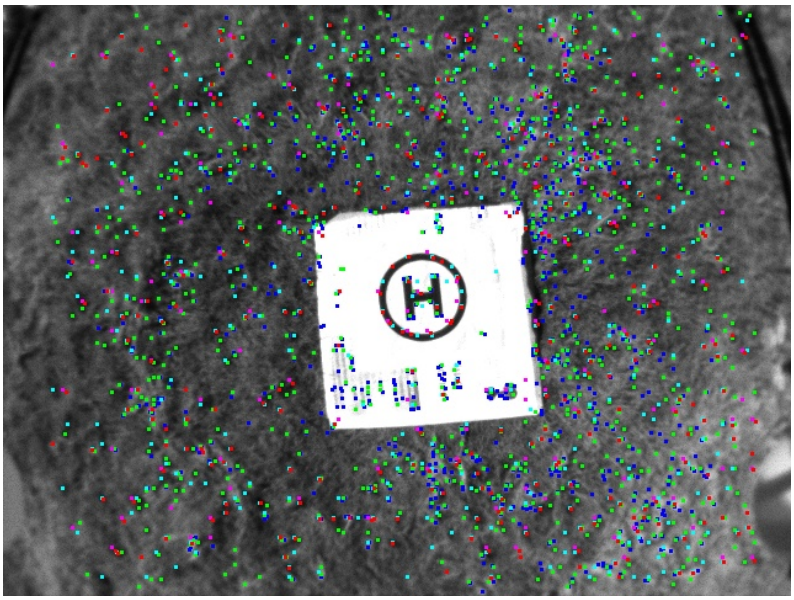


Figure 4.14: Visual features at different image levels marked in different colors (from the bottom to the top level: blue, green, cyan and magenta), when the SLAM system is initialized in the outdoor environment.

are related to vision systems in general. Fig. 4.15a shows a view of intermediate results from our visual SLAM system, which gives an example depicting self-similarities. Such self-similarities may make the feature matching difficult, especially when features are in high density, resulting in false map points to be triangulated. As can be found in Fig. 4.15b, most of those map points with relatively large altitude are false ones. More robust feature matching strategies need to be investigated to cope with this issue. Another way to achieve better tracking robustness can be augmenting the monocular SLAM system with more cameras looking at different directions. Thus more diverse vision features can be matched for localization. This method will be investigated in later chapters. Illumination changes in outdoor environments are another challenge for vision systems, which may easily cause cameras to be overexposed or underexposed, and result in tracking failures. We did our outdoor experiment in sunny daytime, since the illumination does not change much in this case. Thus, to finally go outdoor, we have to develop vision systems employing techniques which can efficiently handle illumination changes, e.g. auto-adjusting camera parameters as proposed in Lu *et al.* (2010).

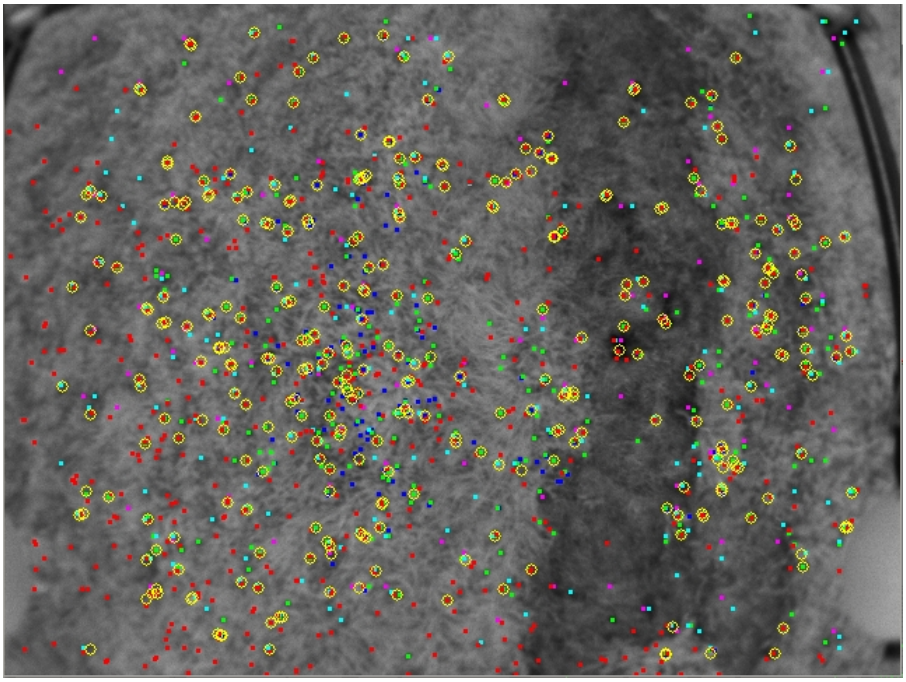
4.7 Conclusions and Discussions

In this chapter we have presented a monocular vision system which enables an MAV to navigate autonomously in unknown landing areas, and search for the landing site on which it is designated to land. Our visual SLAM system can provide accurate pose estimates for the control of the MAV. We have solved the landing site detection problem by integrating a multi-scale ORB feature matching scheme into the mapping thread of the SLAM framework. We have further utilized the map points produced by the SLAM system to accurately estimate the 3D position of the landing site, using a RANSAC-based method. No absolute scale information of the landing site is needed for its pose estimation.

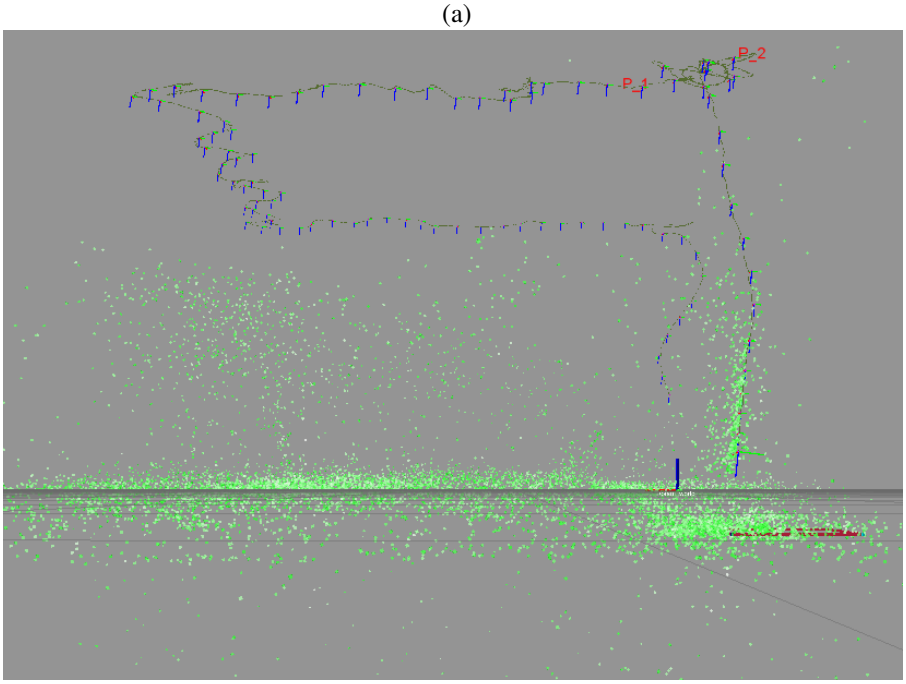
By evaluating the position estimation results of different landing sites, we have shown that our method is flexible and accurate enough for the proposed task of searching for and landing on an arbitrarily textured landing site. Finally, we have demonstrated our claims by the autonomous navigation and landing flights of our MAV in indoor and in outdoor environments. The successful outdoor flight in the challenging scenario proves that our visual system can be extended to applications in complex outdoor-environment. Video demonstration for the work presented in this chapter can be found online[1].

For an autonomous landing phase at the end of a long-term mission of an UAV/-MAV, we propose to fuse IMU data to get its accurate short-term relative pose estimates, which can provide a metric scale constraint to initialize the monocular visual SLAM system. Thus, autonomous searching for and landing on an arbitrary landing site could be achieved with a similar strategy as proposed in this chapter.

[1] `http://www.youtube.com/channel/UCQd6_G6qyvGHUmz7NUelDZQ/videos`, accessed 12-April-2014



(a)



(b)

Figure 4.15: (a) An intermediate result of the SLAM system when the MAV flies above a meadow. Those blue, green and cyan points are features at different image levels, and the red points are built map points, while those successfully matched map points are marked with yellow circles. (b) A side view of the final built map in the outdoor scenario.

Chapter 5

Multi-Camera Visual SLAM for Autonomous MAVs

In this chapter, we extend a monocular visual SLAM system to utilize two cameras with non-overlapping fields of view. The resulting visual SLAM system is used to enable autonomous navigation of an MAV in unknown environments. The methodology behind this system can easily be extended to multi-camera configurations, if the onboard computational capability allows this. We analyze the iterative optimizations for pose tracking and map refinement of the SLAM system in multi-camera cases. This ensures the soundness and accuracy of each optimization update. Our multi-camera visual solution is more resistant to tracking failure than conventional monocular visual SLAM systems, especially when MAVs fly in complex environments. It also brings more flexibility to configuring multiple cameras used onboard of MAVs. We demonstrate its efficiency with both autonomous flight and manual flight of an MAV. The results are evaluated by comparing with ground truth data provided by an external tracking system.

Large parts of this work have been pre-published in Yang, S. *et al.* (2014c).

5.1 Introduction

Monocular vision systems with conventional lenses normally have rather limited fields of view. This is one of their disadvantages when being used for MAV navigation applications, since a larger field of view (FOV) can provide better environmental awareness. In the context of visual SLAM on MAVs, a larger FOV of the vision system can be more resistant to tracking failure. The FOV can be enlarged by using a lens with a wider viewing angle (even fish-eye lens), at the cost of suffering from larger lens distortion and loss of environmental details, due to a smaller angular camera resolution. This also applies to catadioptric omnidirectional vision systems (Lu *et al.*, 2011). Another type of omnidirectional vision systems combines multiple cameras into one vision system, maintaining a single-viewpoint projection model. However, these cameras need to be very precisely configured within relatively heavy mechanical systems in order to preserve this model, and thus are not flexible enough for MAV applications.

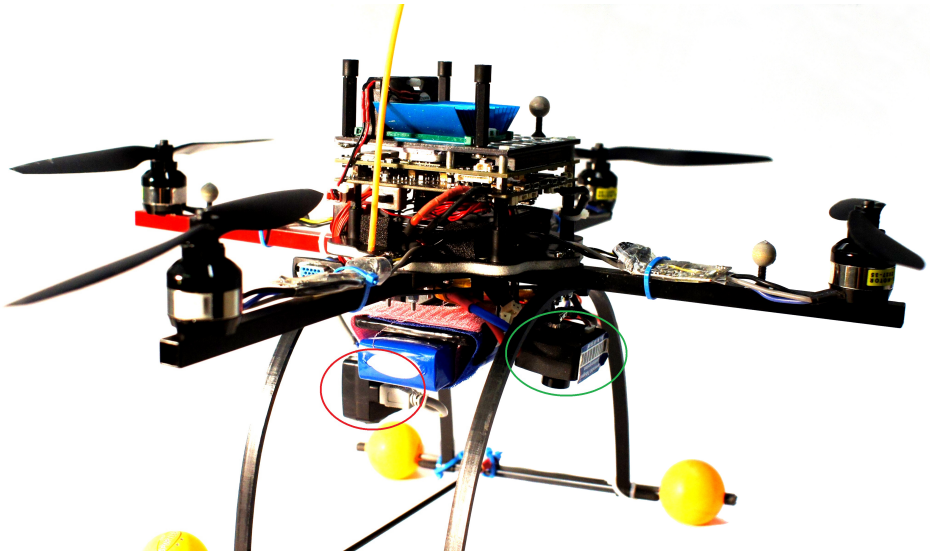


Figure 5.1: Our MAV platform, with two cameras facing two different directions: downward (green ellipse) and forward (red ellipse).

In this chapter, we focus on achieving autonomous navigation of MAVs by extending PTAM to utilize image features from multiple cameras. We expand the FOV of our MAV vision system by using two cameras mounted looking in two different directions (forward and downward) to capture more critical views, as shown in Fig. 5.1. The choice of the count of cameras is the result of a compromise between tracking robustness and onboard computational capability.

Our method allows a SLAM system to integrate images captured from various useful perspectives without requiring the cameras to be mounted in a specific way in order to keep a single-viewpoint model. This makes the configuration of cameras very flexible. On the other hand, since multiple cameras no longer preserve this model, using features from multiple cameras in SLAM is not trivial: How the features are involved in iterative optimizations needs to be carefully analyzed. Based on that analysis, we are able to integrate those image features into a single visual SLAM system. This enables our MAV to achieve more robust pose tracking and to build a map that consists of more interesting regions of the environment.

The remainder of this chapter is organized as follows. Related work on pose estimation using multiple cameras is reviewed in Sec. 5.2. In Sec. 5.3, the analysis on optimizations in multi-camera SLAM is presented. Further implementation details about the proposed visual SLAM system are presented in Sec. 5.4. The performance of the proposed SLAM system is evaluated in the experiments in Sec. 5.5. Finally, in the last section, we provide a summary and discussion of this chapter.

5.2 Related Work

In order to use multiple cameras for pose estimation, the extrinsic parameters of those cameras need to be calibrated. Here we describe the previous work solving this calibration problem for multiple cameras with non-overlapping fields of view. Carrera *et al.* (2011) proposed a SLAM-based automatic calibration scheme for multiple cameras. The scheme uses global bundle adjustment to optimize the alignment of maps built by different visual SLAM instances, each processing images from one corresponding camera. The proposed solution computes the relative 3D poses among cameras up to scale. An advantage of this method is that it does not require any external infrastructure or calibration pattern. An extensive review on auto-calibration methods for such camera systems can also be found in this work.

Previous work on developing multi-camera systems can mainly be found in applications of surveillance and object tracking (Collins *et al.*, 2000; Kettnaker and Zabih, 1999; Rosales and Sclaroff, 1999; Krumm *et al.*, 2000). More relevant related work appears in the context of structure from motion (SFM). The work in Pless (2003) presents a theoretical treatment of multi-camera systems in SFM deriving the generalized epipolar constraint. The work in Frahm *et al.* (2004) proposes a virtual camera as a representation of a multi-camera system for pose estimation. A structure-from-motion scheme is achieved using multiple cameras in this work.

A number of multi-camera systems for pose estimation of mobile robots can be found in the literature. In Ragab (2008), pose estimation of a mobile robot is solved by placing two back-to-back stereo pairs on the robot using the Extended Kalman Filter (EKF). The work in Lee *et al.* (2013a) adopts a generalized camera model described in Pless (2003) for a multi-camera system, to estimate the ego-motion of a self-driving car using a 2-Point RANSAC algorithm. This system allows point correspondences among different cameras. In Lee *et al.* (2013b), pose-graph loop-closure constraints are computed. The relative pose between two loop-closing pose-graph vertices is obtained from the epipolar geometry of the multi-camera system. Kaess and Dellaert (2006) presented a visual SLAM system with a multi-camera rig using Harris corner detectors (Harris and Stephens, 1988). In their further work in Kaess and Dellaert (2010), a Bayesian approach to data association is presented, taking into account moving features, which can be observed by cameras under robot motion. The work in Solà *et al.* (2008) provides solutions to two different problems in multi-camera visual SLAM: automatic self-calibration of a stereo rig while performing SLAM and cooperative monocular SLAM.

Another work most similar to our work in this chapter is that in Harmat *et al.* (2012), which uses PTAM with multiple cameras mounted on a buoyant spherical airship. It employs a ground-facing stereo camera pair which can provide metric scale, together with another camera mounted pointing to the opposite direction using a wide-angle lens. There are three advantages of our work compared to it. First, we provide a solid mathematical analysis on how measurements from different cameras can be integrated in each optimization process of PTAM. The analysis guarantees soundness and accuracy of

the optimizations for the pose update and bundle adjustment using measurements from multiple cameras. Second, we make use of the fact that multiple cameras are typically mounted rigidly, and force camera poses to obey their rigid extrinsic calibration in bundle adjustment, as will be shown in Sec. 5.3.4. This ensures a consistent map to be built in a multi-camera SLAM system. Furthermore, the resulting pose tracking accuracy in Harmat *et al.* (2012) was evaluated only in a manual flight experiment, and the position errors are reported to be higher than our results, although stereo cameras were used there. We also demonstrate that our SLAM system can enable autonomous navigation of an MAV.

5.3 Multi-Camera Visual SLAM

We implemented our SLAM system based on the PTAM system as in Chapter 4. The reason for this choice is that PTAM provides an efficient tracking module and it is able to generate an accurate map with a large number of map points from the environment. Furthermore, PTAM uses iterative optimizations for both pose tracking and map refinement. Thus, we can extend it to incorporating multi-camera image features by solving those optimization problems in multi-camera cases. In this section, we provide the mathematical analysis on how measurements from different cameras can be integrated in one SLAM system.

5.3.1 Multi-camera projection and pose tracking

Using the same calibrated camera projection model described in Sec. 2.3, the image projection of the j^{th} map point to the i^{th} camera is

$$\mathbf{u}_{ji} = \mathcal{P}_{C_i}\left(E_{c_iw}\mathbf{p}_j\right), \tag{5.1}$$

where $\mathcal{P}_{C_i}$ is the projection function of the i^{th} camera considering lens distortion as described in Eq. (2.15), $\mathbf{p}_j$ are the world coordinates of the j^{th} map point p_j, and E_{c_iw} is a member of the Lie group SE(3), which represents the i^{th} camera pose in the world coordinate system, containing a rotation and a translation component.

Since our goal is to use the SLAM system to track the pose of the MAV, without losing generality, we compute the pose update of one specific camera, which we call the first camera, C_1, based on the measurements from all cameras. The pose of other cameras can be updated by assuming a constant transformation relative to C_1. Thus, with a calibrated transformation between the first camera and the MAV body coordinate system, the MAV pose can be updated, as illustrated in Fig. 5.2. Following this idea, pose updates of all cameras can be expressed with one single six-element vector μ using the exponential map:

$$E'_{c_iw} = E_{i1} \cdot e^{\mu} \cdot E_{c_1w}, \tag{5.2}$$

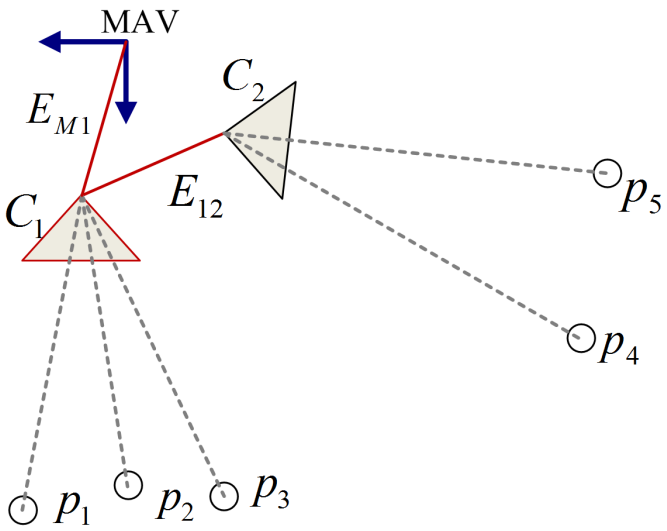


Figure 5.2: A 2D illustration of MAV and camera pose updates in a dual-camera case. Although points $p_{j\in\{1,2,...,5\}}$ are measured in either camera C_1 or C_2, these measurements are used to optimize the pose of camera C_1 only. The camera C_2 pose and the MAV pose are updated by assuming rigid connections to camera C_1.

where μ is an element of the Lie algebra $se(3)$, and E_{i1} is the pose of C_1 in the i^{th} camera coordinate system. For C_1 itself, E_{i1} is simply the identity matrix. The pose tracking (and a part of mapping) problem of the SLAM system now mainly consists of how to obtain an optimized μ as a pose update for camera C_1 by minimizing a certain objective function. The advantage of the parametrization of the camera pose updates using the six-element vector μ is that it allows a closed-form differentiation of Eq. (5.2).

5.3.2 Optimizations in the multi-camera SLAM

Both the camera pose update and map refinement (using bundle adjustment) in PTAM are based on iteratively minimizing a robust objective function of the reprojection errors of sets of image measurements S_i, which are observed map points in each camera (or keyframe) i. It needs to be analyzed how to correctly use features from different cameras in such optimization processes in order to preserve the soundness and accuracy. In an n-camera (or n-keyframe) system, we need to minimize the function:

$$\sum_{i=1}^{n}\sum_{j\in S_i}\mathrm{Obj}\left(\frac{|\,\mathbf{e}_{ji}\,|}{\sigma_{ji}},\sigma_T\right), \tag{5.3}$$

where Obj is the Tukey biweight objective function (Huber, 2011), $|\,\mathbf{e}_{ji}\,|$ is the reprojection error of point j measured in camera (or keyframe) i, σ_{ji} is the estimated measure-

ment noise of point j, and σ_T is a median-based robust standard-deviation estimate of the distribution of all reprojection errors (Zhang, 1997). $\mathbf{e}_{ji}$ is defined as the difference between the image reprojection of map point j and its actual image measurement:

$$\mathbf{e}_{ji} = \mathbf{u}_{ji} - \hat{\mathbf{u}}_{ji}. \tag{5.4}$$

The minimization problem can be solved by multiple iterations of reweighted nonlinear least squares (Wright and Nocedal, 1999). One fundamental requirement to do this efficiently is to differentiate $\mathbf{e}_{ji}$ (i.e. to obtain the Jacobians of $\mathbf{e}_{ji}$) with respect to those parameters that need to be estimated, at each iteration step. In pose tracking, the differentiation of $\mathbf{e}_{ji}$ with respect to the estimated camera pose update needs to be computed. In bundle adjustment, the differentiation of $\mathbf{e}_{ji}$ with respect to the map point j position change is also required. The work in Blanco (2010) provides a good tutorial to the mathematical background of related differentiations. We will focus on the differentiations of $\mathbf{e}_{ji}$ in multi-camera systems in the following two sections.

5.3.3 Pose update with multiple cameras

For the pose update of an n-camera system, the optimization problem is to find the optimal pose update μ for camera C_1:

$$\mu_1 = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j \in S_i} \operatorname{Obj}\left(\frac{|\, \mathbf{e}_{ji} \,|}{\sigma_{ji}}, \sigma_T \right). \tag{5.5}$$

Following the discussion in Sec. 5.3.2, we analyze the differentiations required for solving this optimization problem. For a map point j measured by the first camera C_1, we can compute the Jacobian matrix of $\mathbf{e}_{ji}$ with respect to the estimated C_1 pose update μ using the chain rule as

$$\mathbf{J}_{1\mu} = \frac{\partial \mathcal{P}_{C_1}\left(e^{\mu} E_{c_1 w} \mathbf{p}_j\right)}{\partial \mu} = \left. \frac{\partial \mathcal{P}_{C_1}(\mathbf{c})}{\partial \mathbf{c}} \right|_{\mathbf{c}=E_{c_1 w}\mathbf{p}_j} \cdot \frac{\partial \left(e^{\mu} E_{c_1 w} \mathbf{p}_j\right)}{\partial \mu}. \tag{5.6}$$

The first term of the above matrix product is the Jacobian of the camera projection function in Eq. (5.1), and the last term is:

$$\frac{\partial \left(e^{\mu} E_{c_1 w} \mathbf{p}_j\right)}{\partial \mu} = \begin{pmatrix} \mathbf{I}_3 & -[E_{c_1 w} \mathbf{p}_j]_{\times} \end{pmatrix}. \tag{5.7}$$

However, for map points measured by other cameras, with Eq. (5.2), the differentiation becomes:

$$\mathbf{J}_{i\mu} = \frac{\partial \mathcal{P}_{C_i}\left(E_{i1}e^{\mu}E_{c_1w}\mathbf{p}_j\right)}{\partial \mu} = \left.\frac{\partial \mathcal{P}_{C_i}(\mathbf{c})}{\partial \mathbf{c}}\right|_{\mathbf{c}=E_{c_iw}\mathbf{p}_j} \cdot \frac{\partial\left(E_{i1}e^{\mu}E_{c_1w}\mathbf{p}_j\right)}{\partial \mu}. \tag{5.8}$$

Its difference to Eq. (5.6) lies in the last term of this equation:

$$\frac{\partial\left(E_{i1}e^{\mu}E_{c_1w}\mathbf{p}_j\right)}{\partial \mu} = Rot(E_{i1}) \cdot \begin{pmatrix}\mathbf{I}_3 & -[E_{c_1w}\mathbf{p}_j]_\times\end{pmatrix}, \tag{5.9}$$

where $Rot(E_{i1})$ represents the rotation component of E_{i1}.

5.3.4 Bundle adjustment with multiple cameras

Bundle adjustment in PTAM means solving the following minimization problem:

$$\{\{\mu_2 \dots \mu_N\}, \{p_1 \dots p_M\}\} = \underset{\{\{\mu\},\{p\}\}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} \mathrm{Obj}\left(\frac{|\,\mathbf{e}_{ji}\,|}{\sigma_{ji}}, \sigma_T\right), \tag{5.10}$$

where N is the number of keyframes and M is the number of observed map points that need to be updated. The first keyframe is normally assumed to be fixed and to function as the reference frame.

In a multi-camera system, we assume that the relative poses among the group of new keyframes obtained at the same time t by different synchronized cameras are constant, since the cameras are mounted rigidly. Thus, in bundle adjustment, we can use image measurements from all cameras to compute the optimal pose updates of the keyframe set $\mathbf{K}_1$ which are obtained by the first camera C_1. The poses of other rigidly connected keyframes are computed based on the updated poses of their associated keyframes in $\mathbf{K}_1$. This multi-camera bundle adjustment strategy is illustrated in Fig. 5.3. Therefore, a consistent map can be built by using multiple cameras. Here, the consistency is in the sense of rigid connectivity among sub-maps generated by multiple cameras.

In the above case, to solve bundle adjustment, we differentiate $\mathbf{e}_{ji}$ with respect to the corresponding keyframe (in $\mathbf{K}_1$) pose update μ_i, which can be computed in the same way as in Eq. (5.6) or Eq. (5.8), depending on the camera identity of the point j, i.e. by which camera this point has been measured. The Jacobian of $\mathbf{e}_{ji}$ with respect to the estimated point j pose can be expressed in a consistent way, regardless of the camera used to measure this point:

$$\mathbf{J}_{\mathbf{p}_j} = \frac{\partial \mathcal{P}_{C_i}\left(E_{c_iw}\mathbf{p}_j\right)}{\partial \mathbf{p}_j} = \left.\frac{\partial \mathcal{P}_{C_i}(\mathbf{c})}{\partial \mathbf{c}}\right|_{\mathbf{c}=E_{c_iw}\mathbf{p}_j} \cdot \frac{\partial\left(E_{c_iw}\mathbf{p}_j\right)}{\partial \mathbf{p}_j}. \tag{5.11}$$

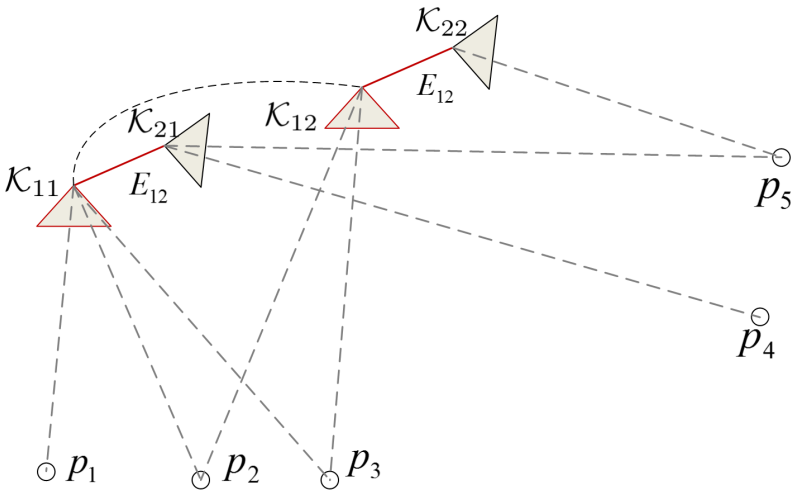


Figure 5.3: A 2D illustration of bundle adjustment in a dual-camera case with two keyframes from each camera ($\mathcal{K}_{11}, \mathcal{K}_{12}$ from camera C_1 and $\mathcal{K}_{21}, \mathcal{K}_{22}$ from camera C_2). In this bundle adjustment problem, the measurements to points $p_{i\in\{1,2,\ldots,5\}}$ are used to optimize the poses of keyframes $\mathcal{K}_{11}, \mathcal{K}_{12}$ and the positions of all these points. The poses of keyframe $\mathcal{K}_{21}$ and $\mathcal{K}_{22}$ are updated by assuming their rigid connections to $\mathcal{K}_{11}$ and $\mathcal{K}_{12}$, respectively.

The last term simply becomes:

$$\frac{\partial\left(E_{c_iw}\mathbf{p}_j\right)}{\partial\mathbf{p}_j} = Rot\left(E_{c_iw}\right). \tag{5.12}$$

5.4 Implementation

Our two cameras have no overlap in their fields of view. This configuration achieves a maximal total FOV of the vision system. In the software of the SLAM system, we modify both the pose tracking thread and the mapping thread of the original PTAM system. We also re-organize the map in order to efficiently manage it in our dual-camera case. Details are presented in the following sections.

5.4.1 Organizing the map

Since we do not expect our MAV to do aggressive maneuvers with extreme roll or pitch rotation, the two cameras mounted on our MAV as shown in Fig. 5.1 will hardly obtain images with similar perspectives to the environment. Thus, we can assume that the two cameras share no common feature point from the environment. Then the global map of the SLAM system can be treated as two sub-maps, each of which corresponds to one

camera. We divide the map by giving a label to each keyframe and map point, indicating from which camera they have been obtained.

The above organization method can make map operations in both pose tracking and mapping more efficient. For example, it is required to search for all potentially visible map points in each pose tracking process. When performing this search in an image taken by camera C_1, now only the map points measured by C_1 need to be checked whether they are potentially visible or not. In the mapping process, neighbouring keyframes can be searched in a sub-map only, instead of the whole global map. This is done whenever we need to decide whether a new keyframe should be added, or to choose an existing keyframe for triangulating new map points.

Despite our choice of a sup-map organization method, we still maintain a single global map. It is a trivial issue in PTAM to assume features can be matched among multiple cameras, since both map point triangulation and bundle adjustment are designed to handle multiple observations of a feature point. Thus, if multiple cameras can share common perspectives to the environment, e.g. they are mounted looking in the forward direction or the sides of the MAV, we can easily adjust the organization of the map to allow feature matching among multiple cameras. In this case, each sub-map should include all the keyframes and map points generated by all the cameras that share common perspectives.

5.4.2 Pose tracking

Map points in the two sub-maps are reprojected to their corresponding source camera to decide whether they are potentially visible. Successful matches between image features and those potentially visible points serve as image measurements which are used in iterative optimizations for the pose update of the camera C_1. Then the optimal pose of the camera C_2 is computed by using Eq. (5.2).

In the original PTAM, if map point j is measured in image pyramid level s ($s \in \{0,1,2,3\}$), the measurement noise is estimated to be $\sigma_j = 2^s$. In our dual-camera pose tracking optimizations, we assume that the measurement noises of the two cameras follow the same distribution, i.e. for any camera $i \in \{1,2\}$ measuring point j, we have $\sigma_{ji} = 2^s$. This also applies to the optimizations in bundle adjustment. Measurement noise of each camera in a multi-camera system using significant different cameras or lenses should not be considered to follow the same distribution. Instead, the distributions can be estimated according to the actual performance of each sensor, as proposed in Scherer *et al.* (2012).

5.4.3 Mapping

New keyframes $\mathcal{K}_{1n}$ and $\mathcal{K}_{2n}$ from both dual cameras are added to the global map, when the geometric distance of $\mathcal{K}_{1n}$ (or $\mathcal{K}_{2n}$) to its closest neighbor obtained by the same camera is larger than a threshold. This means, if we obtain a keyframe from any of the dual cameras that should be added to the map, the keyframe obtained by the other camera

at the same time will also be added. Thus, in the global map, each keyframe obtained by the camera C_1 is always associated with another keyframe obtained by C_2, and vice versa. Here, the geometric distance measures the sum of weighted translation distance and angular difference, as has been done in the original PTAM.

Additionally, we attempted to implement a scheme that allows individual keyframes from only one of the dual cameras to be added to the global map: a keyframe from a camera is added to the map, only if its geometric distance to its closest neighbor exceeds a threshold. However, this scheme does not obviously reduce the total number of keyframes in the map. On the contrary, it requires a complex logic in order to achieve correct associations between the two keyframe sets which are obtained by the two cameras.

To achieve real-time performance of the SLAM system during its exploration, we only retain the local bundle adjustment and abandon global bundle adjustment in the mapping thread. The local bundle adjustment process involves a subset of map points and keyframes in the global map which are generated by both cameras. It computes the pose updates for keyframes and the 3D position updates for map points, which are added to the keyframe set $\mathbf{K}_a$ and the point set $\mathbf{p}_a$, respectively. As explained in Sec. 5.3.4, only the keyframes from the camera C_1 will be added to $\mathbf{K}_a$. Keyframes which are obtained by the camera C_2 and associated (rigidly connected) to $\mathbf{K}_a$ form the set $\mathbf{K}_c$, whose poses can be computed by using the optimized poses of $\mathbf{K}_a$. $\mathbf{p}_a$ consists of all the points which are measured in $\mathbf{K}_a$ or $\mathbf{K}_c$. A further fixed keyframe set $\mathbf{K}_f$ contains any keyframe in which a measurement of any point in $\mathbf{p}_a$ has been made. Then the minimization of the local bundle adjustment becomes

$$\{\{\mu_{i\in\mathbf{K}_a}\},\{p_{j\in\mathbf{p}_a}\}\} = \underset{\{\{\mu\},\{p\}\}}{\operatorname{argmin}} \sum_{i\in\mathbf{K}_a\cup\mathbf{K}_c\cup\mathbf{K}_f} \sum_{j\in\mathbf{p}_a\cap S_i} \operatorname{Obj}\left(\frac{|\,\mathbf{e}_{ji}\,|}{\sigma_{ji}},\sigma_T\right), \tag{5.13}$$

which is solved by using the Levenberg-Marquardt method (Hartley and Zisserman, 2004) as in the original PTAM. The Jacobians of $\mathbf{e}_{ij}$ are solved as described in Sec. 5.3.4. We use a similar strategy as in PTAM to define the keyframe set $\mathbf{K}_a$: It consists of n_k keyframes obtained by the camera C_1, including the newest keyframe and the other $n_k - 1$ ones nearest to it. Normally, we set $n_k = 5$.

5.4.4 Automatic Initialization

Metric scale ambiguity generally exists in monocular camera systems. Our dual-camera system has the same issue since the cameras have no overlap in their respective fields of view, and thus no stereo triangulation can be used to recover the metric scale factor of the map built by the system. We solve this issue by initializing the metric map of our SLAM system similarly as we did in Chapter 4. We use the initialization module presented in Chapter 3 to robustly estimate the pose of the downward-looking camera C_1 during the takeoff phase of our MAV. When the MAV height is larger than a given threshold, the

first camera pose and the associated image are sent to the SLAM system. 3D positions of the feature points in the image are obtained by assuming they lie on the ground plane, which does not need to be strictly true, as demonstrated in the outdoor experiment in Chapter 4. The sub-map corresponding to the first camera C_1 is initialized with those feature points. The sub-map corresponding to the second camera C_2 is initialized after two keyframes from this camera are obtained, when 3D feature points can be triangulated with the known keyframe poses.

5.5 Experiments

5.5.1 Experimental setup

Our software system is implemented in several modules, as we did in Chapter 4. The only difference is that we use two cameras here, and their drivers are in two individual ROS nodes. We synchronize them both at the hardware and the software level. At the hardware level, we use a master-slave scheme to synchronize different cameras. Camera C_1 performs as a master, with an image frequency of $f_m = 30\ fps$. Image acquisitions of camera C_2 are triggered by the signal sent by C_1 at each time it starts to acquire an image. At the software level, we enforce a rule that the time-stamp difference of the two newest images from the two cameras should be smaller than $1/(2 \cdot f_m)\ second$. An image obtained by the slave camera C_2 which breaks this rule will be dropped to avoid unexpected errors during image transportation between camera drivers and the SLAM system. If this happens, the new image from camera C_1 will be used for pose tracking only, and not for the further mapping process.

5.5.2 Enabling autonomous navigation

In this first experiment, we demonstrate the efficiency of our SLAM system to enable autonomous navigation of MAVs. Furthermore, we evaluate its accuracy by comparing its pose tracking results to the ground truth data provided by the external tracking system.

A picture of the experimental environment is shown in Fig. 5.4, in which we also sketch the world coordinate system in visual SLAM and depict the desired path of the MAV. There is a large white area on the desired path, in which no visual feature can be obtained by the downward-looking camera. The MAV autonomously navigates along a predefined rectangular path (plotted in cyan in Fig. 5.5b) in a counter-clockwise direction with a commanded forward speed of $v_s = 0.4m/s$, taking off and finally landing above the origin of the world coordinate system. The takeoff phase is controlled by using pose estimates fed from the initialization module until the SLAM system is initialized. We set the MAV to turn 90 degrees at the first corner, which makes the forward-looking camera unable to triangulate new features and track its pose if it is the only camera in the

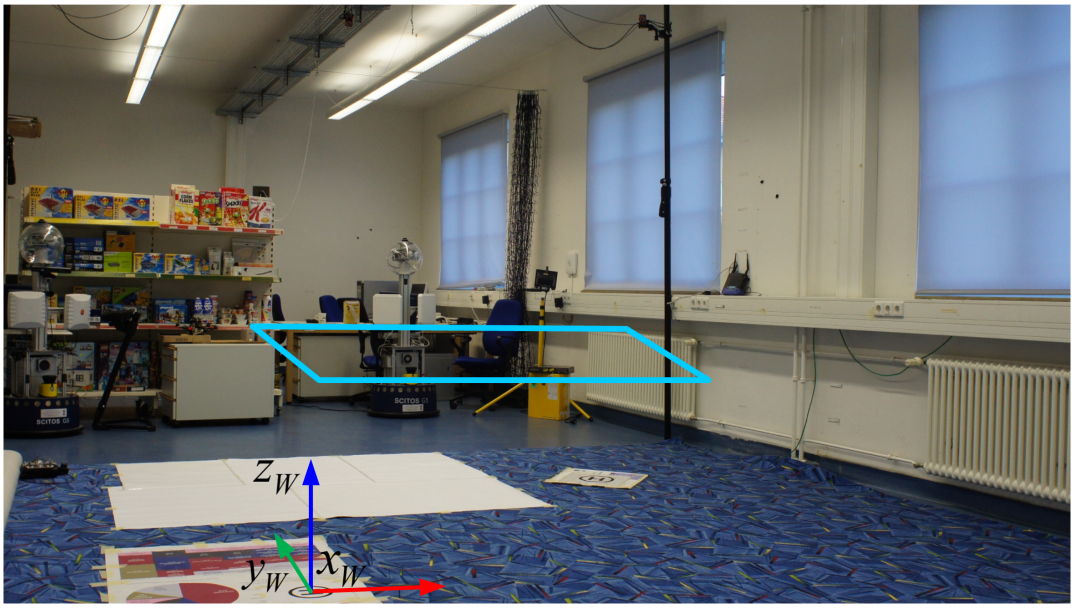
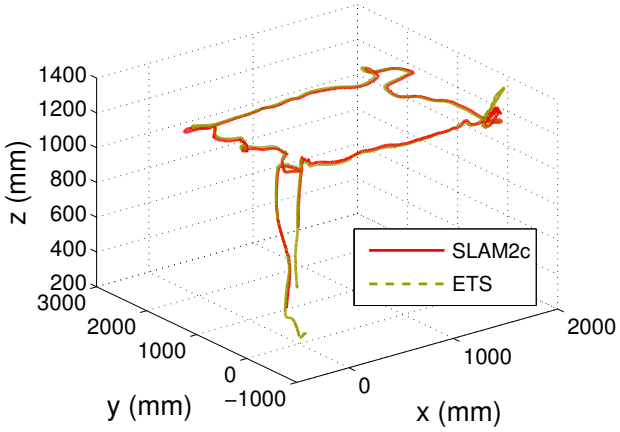


Figure 5.4: A scene of our robot lab where we carry out the experiments. The world coordinate system and the desired flight path of the MAV are indicated inside.

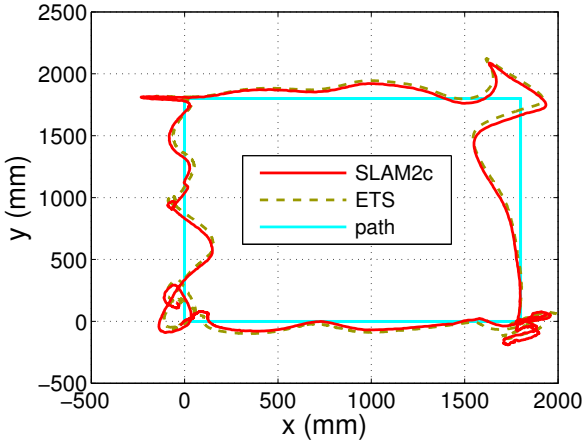
SLAM system. We use the trajectory-following method (described in Sec. 2.2.3) for the trajectory control of our MAV in autonomous flight.

The resulting MAV trajectory during an autonomous flight can be found in Fig. 5.5. The MAV trajectory estimated by our onboard SLAM system (SLAM2c) using two cameras fits well with the ground truth data from the external tracking system (ETS). The SLAM2c attitude estimates are actually less noisy than that of the ETS data. The RMSEs of the pose estimates of SLAM2c data with respect to the ETS data are listed in Tab. 5.1 (the row of Auto). Three sources of noise which contribute to the errors should be noted. First, slow scale drift still exists in our SLAM system, since this system does not have additional sensor data or stereo triangulation to provide metric scale measurements. Second, the extrinsic camera calibration errors can also affect the pose tracking and mapping accuracy. A last minor factor comes from the ground truth data itself, since it is difficult to set the tracking system coordinate frame to perfectly coincide with the world frame.

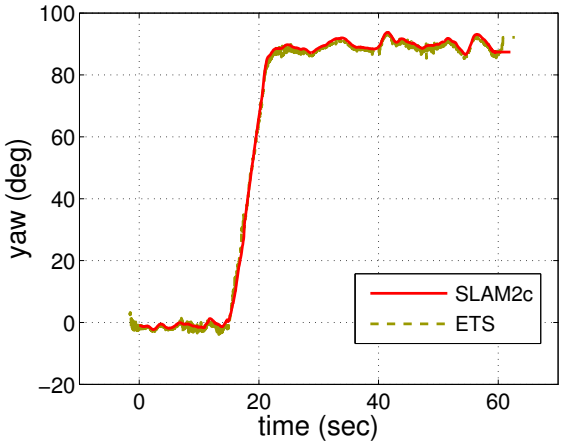
In Fig. 5.5b, we can find fluctuations in the resulting trajectory at the designated path corners. The reasons are that we set the MAV to hover 5 *seconds* at each corner, and we have not implemented a sophisticated and precise pose controller, which is out of the scope of this thesis. If the desired pose of the MAV propagates forward when the MAV is still trying to hover back to a corner, the trajectory may form a fluctuation like the one at the top right corner of Fig. 5.5b.



(a)



(b)



(c)

Figure 5.5: The MAV poses estimated by our visual SLAM system (SLAM2c) and the external tracking system (ETS) during the autonomous navigation. (a) The trajectory on x_W, y_W, and z_W axes, and (b) projected to the $x_W - y_W$ plane. (c) The yaw angle of the MAV.

Table 5.1: MAV pose RMSEs of the whole trajectories in autonomous flight (Auto) and manual flight (Manual), with position errors in *millimeters* and attitude errors in *degrees*.

RMSEs	x	y	z	3D	roll	pitch	yaw
Auto	23.5	37.2	15.9	46.8	0.82	0.81	1.04
Manual	23.4	43.2	12.5	50.7	1.31	1.08	1.06

Table 5.2: MAV pose RMSEs during the part of manual flight when the MAV pose can be tracked by the downward-looking camera alone, with position errors in *millimeters* and attitude errors in *degrees*.

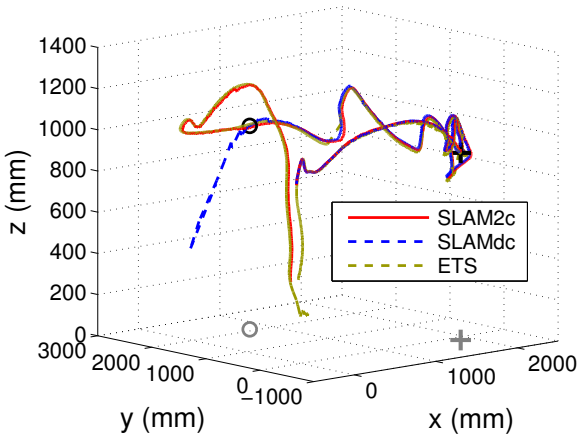
RMSEs	x	y	z	3D	roll	pitch	yaw
SLAM2c	25.5	35.4	13.1	45.6	1.02	0.94	0.91
SLAMdc	38.2	29.7	19.1	52.0	1.30	1.27	1.37

5.5.3 Further evaluation through manual flight

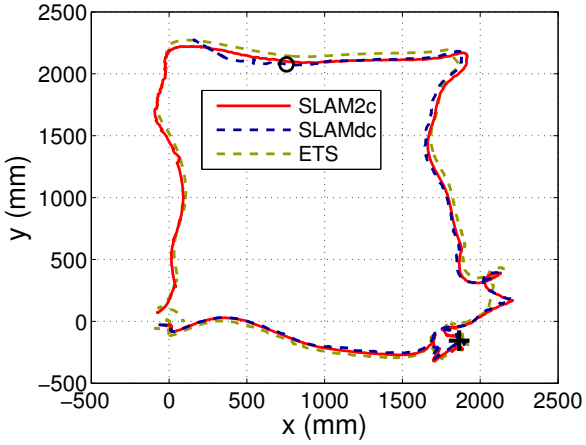
We process an image logfile off-board to perform further evaluation of our SLAM system. The onboard computational capability does not allow us to take image logfiles during autonomous navigation. Thus, we manually control the MAV to follow a similar path as in Sec. 5.5.2, and take a logfile containing images from both cameras and other useful onboard sensor data by utilizing the original ROS functions for recording ROS-bag files.

The MAV trajectory during this manual flight is shown in Fig. 5.6. When using the proposed SLAM system with two cameras, the MAV pose can be tracked well throughout the flight, and also fits well with the ground truth data. The corresponding RMSEs are listed in Tab. 5.1 (the row of Manual). Two minor parts of the ground truth data are missing due to the flight outside the working area of the external tracking system, which is found to be two periods of straight dashed lines in Fig. 5.6c around the time of 17 *seconds* and 29 *seconds*. Fig. 5.7 shows two views of the final map built by our SLAM system including the dual-camera trajectory.

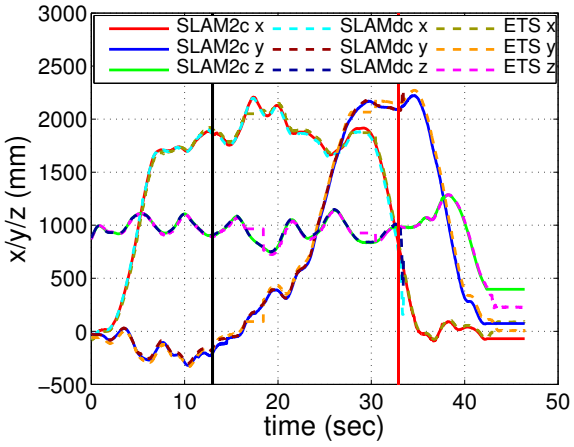
However, if we use PTAM with the downward-looking camera alone, pose tracking will fail when the MAV flies above the white area (see the SLAMdc case in Fig. 5.6). The MAV position where pose tracking fails in this case is marked with black circles in Fig. 5.6a and Fig. 5.6b. We mark the time when it fails with a vertical red line in Fig. 5.6c, in which MAV positions on each axis are shown. During the part of flight before tracking failure happens, the RMSEs of pose tracking using the proposed SLAM



(a)

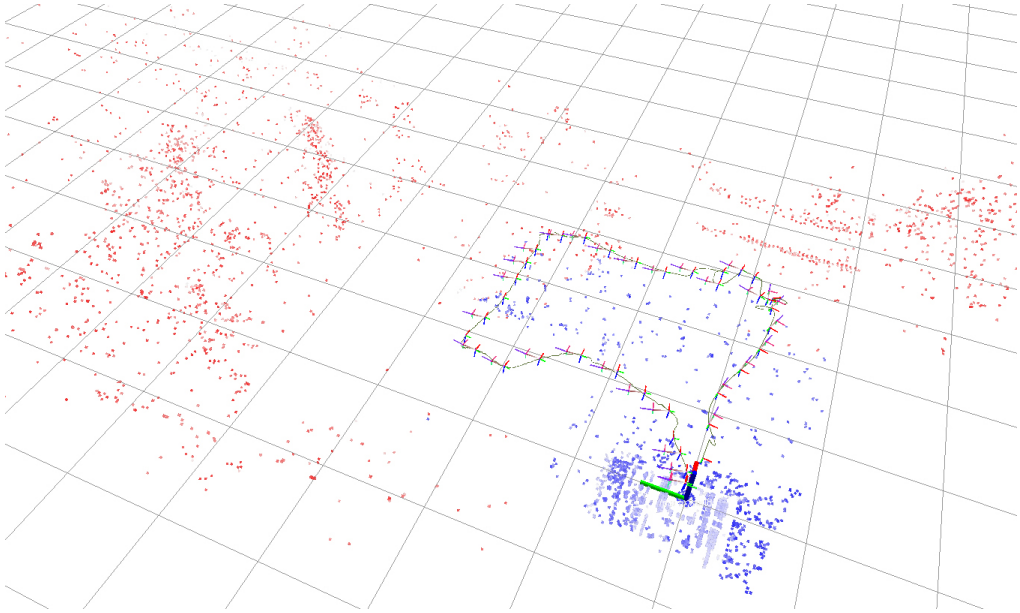


(b)

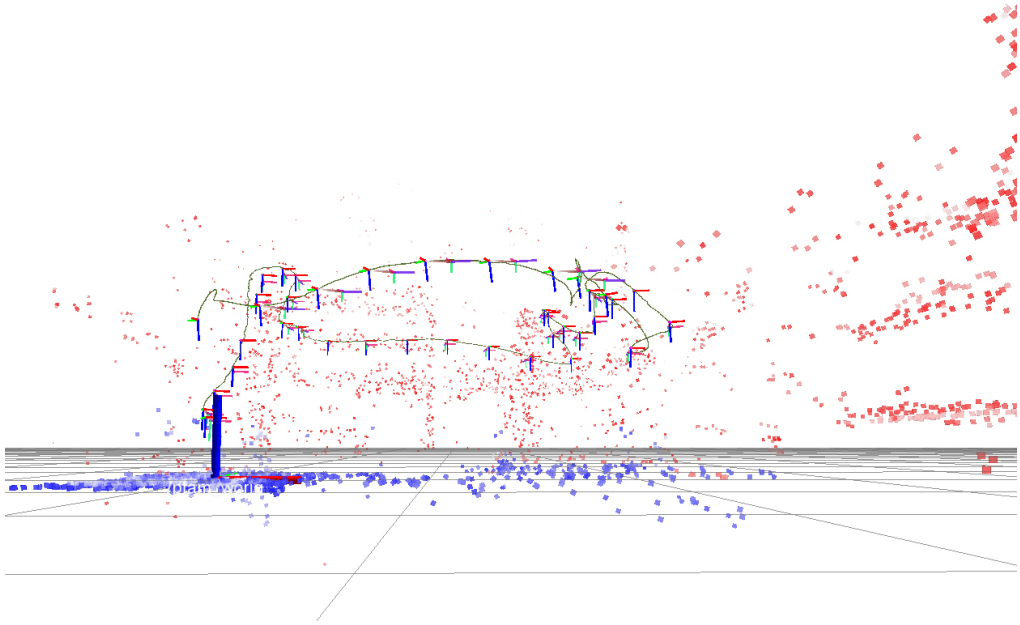


(c)

Figure 5.6: The MAV poses estimated by the proposed SLAM system (SLAM2c), PTAM with only the downward-looking camera (SLAMdc), and the external tracking system (ETS) for the manual flight logfile. (a) The trajectory on x_W,y_W, and z_W axes, (b) projected to the $x_W - y_W$ plane, and (c) with respect to the flight time.



(a)



(b)

Figure 5.7: Built map and the forward-looking camera trajectory during the manual flight, in two different perspectives. Map points measured by the downward-looking camera are marked in blue, those measured by the forward-looking camera in red.

system (SLAM2c) are a bit smaller than those of using only the downward-looking camera (SLAMdc), as can be found in Tab. 5.2. Similarly, if we use the SLAM system with only the forward-looking camera after the initialization of the system, pose tracking will fail again when the MAV turns 90 *degrees* during hovering. Like in the first failure case, we mark the failure with black crosses in Fig. 5.6a and Fig. 5.6b, and a vertical black line in Fig. 5.6c.

5.6 Conclusions and Discussions

In this chapter, We have presented a visual SLAM system which can utilize feature measurements from dual cameras. The mathematical analysis on how those measurements should be integrated in the optimization processes of the SLAM system has been provided. We have demonstrated the efficiency of the system by enabling an MAV with two cameras to navigate autonomously along a predefined path. The experiments with a logfile taken from a manual flight prove that our proposed system is more resistant to tracking failure in complex environments. We have also shown the accuracy of our system by comparing the pose tracking results with the ground truth data provided by the external tracking system. A video demonstration of this work can be found online[1].

In the future, the effect of extrinsic camera calibration errors to the pose tracking and mapping accuracy could be analyzed. Furthermore, it would be interesting to investigate the fact that more parameters could be tracked in the optimizations of the SLAM system, if there is overlap in the fields of view of the multiple cameras: Based on more Jacobian analysis, it is possible to integrate the online changes of extrinsic camera parameters into the optimization problem. Such online changes may happen when the cameras are not rigidly connected, e.g. when they are mounted on different wings of a fixed-wing aerial vehicle.

[1] `http://www.youtube.com/channel/UCQd6_G6qyvGHUmz7NUelDZQ/videos`, accessed 12-April-2014

Chapter 6

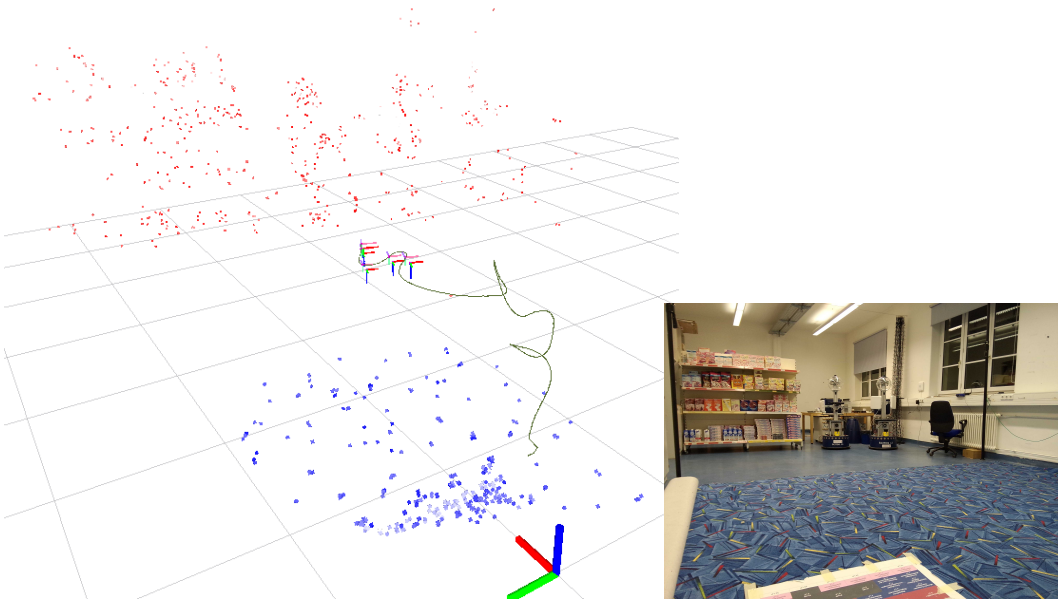
Towards Constant-Time Multi-Camera Visual SLAM for MAVs

In this chapter we present a robust visual SLAM system consisting of a constant-time visual odometry and an efficient back-end with loop-closure detection and pose-graph optimization. Robustness of the visual odometry is achieved by utilizing dual cameras looking in different directions with no overlap in their respective fields of view, as proposed in Chapter 5. The back-end of the SLAM system maintains a keyframe-based global map, which is also used for loop-closure detection. An adaptive-window pose-graph optimization method is proposed to refine keyframe poses of the global map and thus correct pose drift that is inherent in the visual odometry. The position of each map point is then refined implicitly due to its relative representation to its source keyframe. We demonstrate the efficiency of the proposed visual SLAM algorithm for applications onboard of MAVs in experiments with both autonomous and manual flights. The pose tracking results are compared with the ground truth data provided by the external tracking system.

Large parts of this work have been pre-published in Yang, S. *et al.* (2014b).

6.1 Introduction

In Chapter 5, we have presented a visual SLAM system which integrates feature measurements from multiple cameras. Multiple cameras looking in different directions can provide more reliable image features for pose tracking, compared with one monocular camera. Therefore, we achieve more robust pose tracking of an MAV being more resistant to tracking failures. This achievement is especially useful when MAVs fly in complex environments. We keep the complete map during explorations of the visual SLAM system. Thus the system can utilize all previous visible measurements for localization and mapping, and provide more accurate pose estimates and mapping results when working in small-scale environments. However, this SLAM system does not scale well for operations in large-scale environments, in which it will hardly be able to build a consistent map when flying around with loops, i.e. re-visiting certain places during an exploration.



(a) (b)

Figure 6.1: (a) A local map built by our dual-camera visual odometry during an exploration, with $n_L = 4$ (see Sec. 6.4.1). Map points from the forward-looking camera are marked in red color, and those from the downward-looking camera in blue. The trajectory of the forward-looking camera is plotted in green. (b) A scene of the actual lab environment where this experiment was performed, in a similar perspective.

In this chapter, we modify our previous multi-camera visual SLAM system to operate as a robust visual odometry with constant-time cost during large-scale explorations. Our final implementation utilizes two cameras pointing forward and downward, respectively, as we have done in Chapter 5. An example map built by the visual odometry in our dual-camera setting is shown in Fig. 6.1a. Moreover, we propose an efficient visual SLAM back-end for loop-closure detection and correcting pose drift that is inherent in the visual odometry by using pose-graph optimization (PGO).

The back-end of our visual SLAM system maintains a global map organized in keyframes, each of which is associated with some map points represented using positions relative to it. In the global map, we keep map points in a relative representation, the keyframes in absolute representation. Thus, map points can be implicitly updated by PGO operations, which optimize the poses of keyframes in the global map. Strasdat *et al.* (2011) proposed a double-window graph structure for optimizing the global map. Bundle adjustment within a small inner window and pose-graph optimization within a larger window are integrated within one optimization problem. In our work, we decouple bundle adjustment and PGO into a visual odometry front-end and a separate back-end, so that accurate pose tracking can be achieved in constant time, without losing the benefit of PGO which

can correct pose drift of the visual odometry in long-term explorations and thus ensures a consistent global map.

The remainder of this chapter is organized as follows. We review the related work on map representation and loop closure detection in visual SLAM in Sec. 6.2. Then we present our visual SLAM back-end for managing the global map, loop-closure detection and pose-graph optimization, in Sec. 6.3. We provide further details of the implementation of our SLAM system in Sec. 6.4. In Sec. 6.5, we validate our SLAM system by using it onboard of an MAV. Finally, in the last section, we conclude the work of this chapter and discuss possible future work.

6.2 Related Work

In visual SLAM, different ways to represent the environment map have been proposed. The map of the MonoSLAM system (Davison *et al.*, 2007) adopts a probabilistic feature-based map. This map consists of the current estimates of the camera state and all feature points with uncertainty measurements, which are updated by the Extended Kalman Filter (EKF). In PTAM, the map consists of a collection of map points and keyframes. Recently, the work on visual SLAM systems using keyframe-based methods has proposed to organize the map with relative representation to improve the efficiency. In Mei *et al.* (2011), robot positions and the map are represented in a continuous relative representation (CRR) framework, which allows relative bundle adjustment for map refinement and real-time loop closure. The work in Lim *et al.* (2012) presents a hybrid metric-topological map for large scale online environmental mapping. The map is represented as a graph of the keyframes and the relative poses between keyframes. This work strictly enforces the metric property of the local sub-maps, which are optimized by using bundle adjustment and assumed to be rigid segments in the global segment optimization.

Recently, a number of efficient loop-closure detection methods have been proposed using visual vocabulary (Sivic and Zisserman, 2003). In those methods, local features are extracted to represent the appearance information of an image, and the loop-closure detection is solved using a place-recognition scheme. The visual vocabulary model treats an image as a bag of words (BOWs) much like a text document. In this model, each word corresponds to a region in the space of invariant feature descriptors (Cummins and Newman, 2011). A comparison of the visual-vocabulary-based approach to map-to-map (Clemente *et al.*, 2007) and image-to-map (Williams *et al.*, 2008) approaches for loop-closure detection in monocular SLAM can be found in the work of Williams *et al.* (2009).

In Cummins and Newman (2011), images are represented with a bag of words whose co-visibility probability is learned offline using a Cho-Liu tree. In Cadena *et al.* (2012), loop closures are detected based on the BOW method using SURF features. The loop closing verification is carried out using a method based on conditional random fields. The work in Gálvez-López and Tardós (2012) features a hierarchical BOW method. It

uses a vocabulary tree that discretizes a binary descriptor space. This vocabulary tree can efficiently speed up the retrieval of similar images and the verification of geometrical consistency for loop-closure detection. By using BRIEF descriptors (Calonder *et al.*, 2010) which are binary and require very little time to be computed, a much faster conversion of the BOWs can be achieved than using SIFT or SURF descriptors. Rather than building the visual vocabulary based on a prior knowledge of the environment, Nicosevici and Garcia (2012) proposed a method for loop-closure detection using visual vocabularies built online. To investigate the effect of quantity and quality of visual information to place recognition, Milford (2013) presented comprehensive experiments with different datasets using SeqSLAM (Milford and Wyeth, 2012).

6.3 Back-End of the SLAM System

Our multi-camera visual odometry only maintains a local map for pose tracking. We further implement a back-end for the SLAM system to manage and refine a global map, which is built based on the local map of the visual odometry during exploration. The back-end mainly performs loop-closure detection and pose-graph optimization to refine the global map.

6.3.1 The global map representation

We use a keyframe-based global map representation as proposed in Scherer *et al.* (2014). The representation is similar to the one used in the original PTAM: Each keyframe consists of a four-level image pyramid with FAST corners computed in each level, while each map point storing a reference to its source keyframe in which it is measured. However, the work in Scherer *et al.* (2014) adopts a relative representation for the position of each map point and uses local feature descriptors for feature matching.

Instead of storing an absolute position of each map point in the world frame as in PTAM, we store the position of each map point relative to its source keyframe. In this way, we can achieve implicit updates of the absolute position of a map point after the pose of its source keyframe being updated in pose-graph optimization as will be described in Sec. 6.3.3.

We also compute BRIEF descriptors (Calonder *et al.*, 2010) for feature points in all pyramid levels of each keyframe. They are used for wide-baseline matching between keyframes and appearance-based loop-closure detection in the back-end. Compared with some other local feature descriptors, e.g. SIFT (Lowe, 2004), SURF (Bay *et al.*, 2006) and ORB (Rublee *et al.*, 2011), the BRIEF descriptor is not scale invariant nor rotation invariant. However, it is more efficient and is already sufficient for our MAV-application scenario. First, since we compute it in four pyramid levels of a keyframe, scale invariance can be achieved to a certain extent already without considering further pyramid levels. This is similar to our strategy for landing site detection in Sec. 4.4.2. Second, as

will be discussed in Sec. 6.4.1, only keyframes and map points from the forward-looking camera will be used to build the global map. Then, as our MAV flies in a nearly hovering attitude, we do not expect the image measurements of a map point to be significant rotated among different keyframes. Thus, rotation invariance of feature descriptors is not necessary for feature matching. In general, BRIEF descriptors offer us a good compromise between distinctiveness and computation time. On the other hand, ORB features can be a reasonable alternative to BRIEF if rotation invariance is required.

6.3.2 Loop-closure detection

loop-closure detection provides additional pose constraints (edges) to the pose graph of the SLAM system as will be described in Sec. 6.3.3. It is of vital importance for correcting pose drift and building a consistent map during loopy explorations in large-scale environments. We consider the following two types of loop closures:

Appearance-based loop closure

we use appearance-based (image-to-image) method for large-scale loop-closure detection among keyframes. More specifically, we adopt local features (BRIEF features) to represent the image appearance information of a keyframe. We use the open-source implementation of the bag-of-words method developed in Gálvez-López and Tardós (2012) for the loop-closure detection task. In the off-line process, we train the vocabulary tree required by this method using BRIEF descriptors of FAST keypoints extracted from a large number of images taken from various environments.

After a large loop is detected between the current keyframe $\mathcal{K}_A$ and a previous keyframe $\mathcal{K}_B$, we match feature corners in $\mathcal{K}_A$ to map points measured in $\mathcal{K}_B$ to retrieve 2D-3D correspondences. If there are enough 2D-3D correspondences, we estimate the relative pose E_{AB} between these two keyframes to achieve keyframe registration. This can be done efficiently using a P3P (Gao *et al.*, 2003) plus RANSAC method and further refining the result by robust optimization which minimizes 2D reprojection errors of all inlier correspondences. Then the edge $\mathcal{E}_{AB}$ is added to the pose graph. We do not compute the otherwise relative pose E_{BA}, since we expect that the keyframe $\mathcal{K}_A$ would have significant pose drift relative to $\mathcal{K}_B$, and thus positions of map points measured in $\mathcal{K}_B$ should be trusted.

Local loop closure

A potential local loop closure can be detected by trying to register the current keyframe $\mathcal{K}_A$ to its best neighbor $\mathcal{K}_B$ within a certain geometric distance range, and estimating their relative pose in a way similar to the method we used after an appearance-based loop closure is detected. Here, the best neighboring keyframe is decided by its co-visibility of common features with $\mathcal{K}_A$.

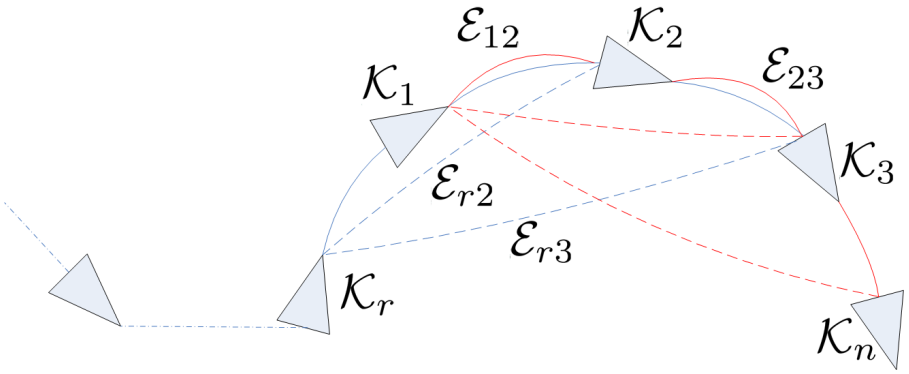


Figure 6.2: Edges added to the pose graph after a new keyframe $\mathcal{K}_n$ is added, in the case of four keyframes involved in bundle adjustment. Edge $\mathcal{E}_{r2}$ and $\mathcal{E}_{r3}$ will also be added to the graph before $\mathcal{K}_r$ is removed from the local map. Edge $\mathcal{E}_{12}$ and $\mathcal{E}_{23}$ will be replaced by new constraints (in red) after a new bundle adjustment operation.

In this case, we compute both E_{AB} (from 2D-3D correspondences matching feature corners in $\mathcal{K}_A$ to map points measured in $\mathcal{K}_B$) and E_{BA} (from 2D-3D correspondences matching feature corners in $\mathcal{K}_B$ to map points measured in $\mathcal{K}_A$). We expect E_{AB} and E_{BA} to agree with each other if a true local loop closure has been detected, as proposed in Scherer *et al.* (2014).

6.3.3 Adaptive-window pose-graph optimization

We apply pose-graph optimization (PGO) at each time when a new keyframe is added to the global map. During this process, we adaptively define a window (sub-graph) of the whole global graph to be adjusted, depending on whether new edges are added from loop closures.

The graph structure

For the purpose of PGO, the graph structure definition of our SLAM system is straightforward: It consists of a set of keyframe-pose vertices ($\mathcal{V}_i$) and relative edges ($\mathcal{E}_{ij}$) describing the relative pose-pose constraints (E_{ij}) among those vertices. Each vertex $\mathcal{V}_i$ stores an absolute pose E_i of its corresponding keyframe $\mathcal{K}_i$ in the world frame of the SLAM system. The edges consist of constraints obtained from the bundle adjustment in the visual odometry and the two types of loop closures described in Sec. 6.3.2.

We notice that each bundle adjustment operation in the visual odometry produces pose constraints among all n_L keyframes in the local map. When a new keyframe $\mathcal{K}_n$ is added to the local map of the visual odometry, the oldest keyframe $\mathcal{K}_r$ will be removed before the next bundle adjustment operation. As a result, poses of the remaining $n_L - 1$

keyframes will be re-adjusted during the next bundle adjustment step without considering the pose constraints to $\mathcal{K}_r$. This means that pose constraints between $\mathcal{K}_r$ and the other $n_L - 1$ nodes may actually contain useful information and should be considered in PGO. Thus, we not only add pose constraints among consecutive keyframes to the pose graph, but also add those between $\mathcal{K}_r$ and all other keyframes within the current local map, as depicted in Fig. 6.2.

Defining the sub-graph for optimization

In PGO, we always consider a window of the whole pose graph $\mathcal{G}$ to be adjusted. We perform uniform-cost search for choosing sub-graph vertices, beginning with the latest added vertex. The cost is measured by the geometric distances among vertices. The sub-graph to be adjusted (denoted as $\mathcal{G}_s$) consists of all those sub-graph vertices and constraints among them. We define the size of the sub-graph vertices in the following adaptive way depending on whether a loop closure has been detected: During regular exploration of the SLAM system without loop closure detected, only a relatively small window (with no more than n_s vertices) of the whole pose graph is to be adjusted. In this case, PGO is still useful due to the constraints produced by the bundle adjustment, as we discussed in the last section. When a loop closure between two keyframes is detected, an edge between their corresponding vertices will be added to the pose graph. Then we expand the graph window $\mathcal{G}_s$ to contain a large number of vertices, until it has included all vertices in the detected loop or a maximal number (n_m) of vertices.

Pose-graph optimization

Given an n-vertices sub-graph $\mathcal{G}_s$ we previously defined, the pose-graph optimization is to minimize the following cost function:

$$F(\mathbf{E}) = \sum_{\mathcal{E}_{\mathcal{G}_s}} \Delta E_{ij}^T \Omega_{ij} \Delta E_{ij}, \tag{6.1}$$

with respect to all vertex poses $\mathbf{E} = (E_1, E_2, ..., E_n)$, where $\mathcal{E}_{\mathcal{G}_s}$ contains all edges in $\mathcal{G}_s$, $\Delta E_{ij} := log(E_{ij} \cdot E_j^{-1} \cdot E_i)$ is the relative pose error in the tangent space of SE(3) and Ω_{ij} is the information matrix of the pose constraint E_{ij}. We estimate Ω_{ij} coarsely as a diagonal matrix in a way proposed in Strasdat *et al.* (2011):

$$\Omega_{ij} = \omega_{ij} \begin{bmatrix} \lambda_r^2 I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & \lambda_t^2 I_{3\times3} \end{bmatrix}, \tag{6.2}$$

with the rotation component λ_r being a constant and the translation component λ_t being proportional to the parallax of E_{ij} which is measured by normalizing the translation $\mathbf{t}_{ij}$ using the average scene depth. Here, $\mathbf{t}_{ij}$ is the translation element of E_{ij}. We consider

the pose constraints produced in the bundle adjustment and loop closures are similar in accuracy. Thus, we set ω_{ij} to be a constant $\omega_{ij} = 1$.

6.4 Implementation

Our visual SLAM system mainly consists of three threads: two threads for the visual odometry (the tracking thread and the mapping thread as in PTAM), and a third thread working for the back-end (back-end thread). The mapping thread of the visual odometry manages a local map and handles most of the interactions with the back-end. The global map and the pose graph are managed by the back-end thread. The flowcharts of the mapping thread and the back-end thread are shown in Fig. 6.3. In this section, we present further implementation details about interactions between the visual odometry and the back-end.

6.4.1 The visual odometry

General approach

In order to achieve constant-time cost in large-scale explorations, we further reduce the complexity of our previous multi-camera visual SLAM system presented in Chapter 5, by fixing the size of keyframes from each camera to be a constant number n_L in the local map. We remove the oldest keyframe when a new keyframe is added to the map. Bundle adjustment is performed within all $m \cdot n_L$ keyframes in an m-camera case. This changes our multi-camera SLAM system to be an efficient constant-time visual odometry.

Motion-model update

In the tracking thread, we assume a slowly decaying-velocity model of the cameras. In each tracking process, we estimate a prior pose E_p of the camera C_1 as the initial guess of the pose estimation, based on the camera velocity $\mathbf{v}_c$ and the camera pose E_l in the last tracking process. Once the local map is updated by the mapping thread according to the updated global map, we update the prior estimation E_p to avoid tracking failure. This is performed by simply retaining the velocity $\mathbf{v}_c$ unchanged, and updating E_p by assuming a constant relative pose E_{ip} to its neighbouring keyframe $\mathcal{K}_i$. Thus, we have the new prior pose $E'_p = E_p E_i^{-1} E'_i$, where E_i and E'_i are the poses of K_i before and after the local map update, respectively.

Removing old keyframes from the local map

If the number of keyframes from camera C_i in the local map exceeds n_L after a new keyframe is added, we remove the oldest keyframe $\mathcal{K}_r$ from the local map. Before that, we first update data associations of the map points measured in $\mathcal{K}_r$.

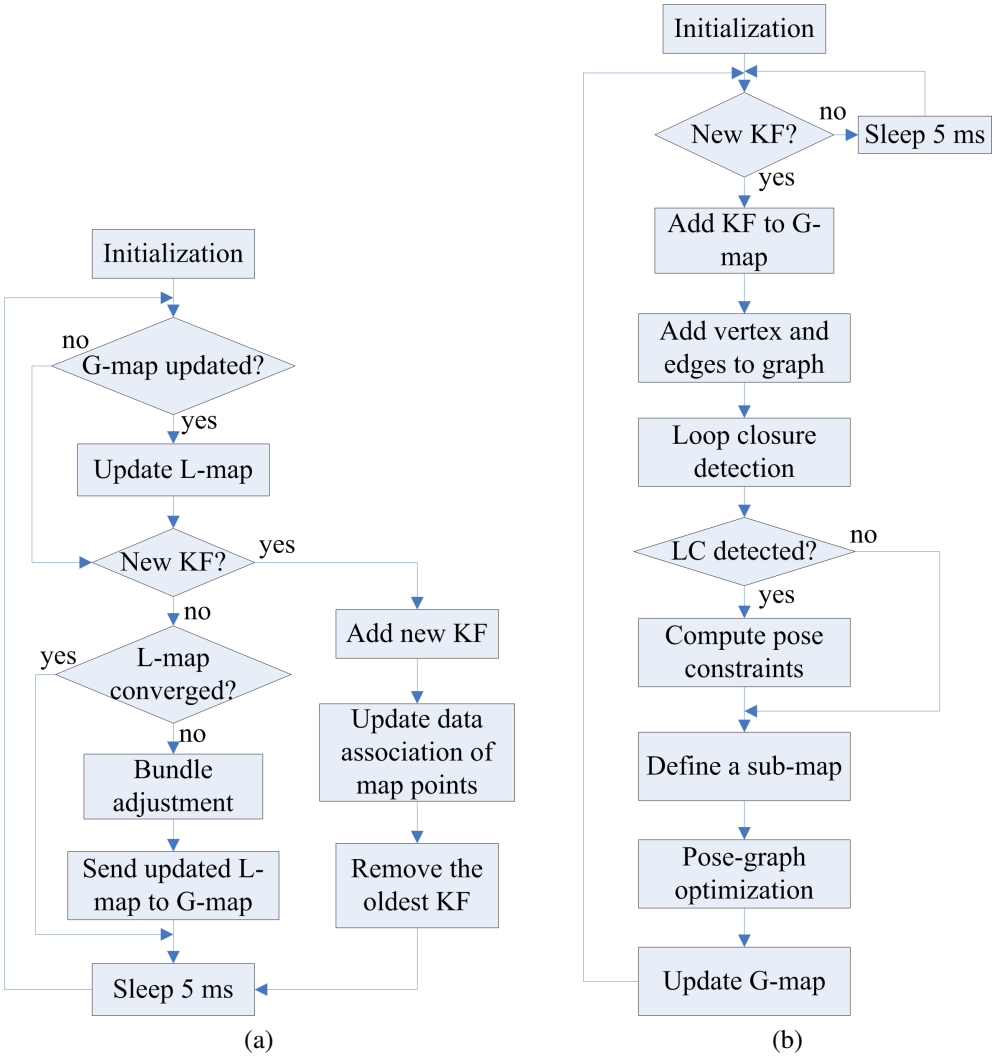


Figure 6.3: (a) the mapping thread of the visual odometry, and (b) the back-end thread of the SLAM system. Abbreviations: KF (keyframe), L-map (local map), G-map (global map), and LC (loop closure).

Each keyframe $\mathcal{K}_i$ is associated with a set of map points p_j. We call $\mathcal{K}_i$ the source keyframe of p_j, since after it obtains a measurement to p_j, it is then used to triangulate p_j together with one of its neighboring keyframes which also measures p_j. When $\mathcal{K}_i$ should be removed, we only remove those map points which are not measured by any other keyframe in the local map. Other map points could be important for later pose tracking. If p_j is measured by another keyframe $\mathcal{K}_m$, we transfer its source keyframe identity to $\mathcal{K}_m$, and update its related data association with the measurement in $\mathcal{K}_m$, which will be used in pose tracking. Meanwhile, we mark p_j indicating that it has been sent to the global map already, in order to avoid re-sending it with $\mathcal{K}_m$ in the future.

Updating the local map

After each pose-graph optimization process, we update the local map according to the global map, including updating keyframe poses and the positions of their measured map points.

Here, we distinguish keyframe $\mathcal{K}_{1j}, j \in \{1,2,...,n_L\}$ from the downward-looking camera C_1 and keyframe $\mathcal{K}_{2j}, j \in \{1,2,...,n_L\}$ from the forward-looking camera C_2. We assume that the pose of $\mathcal{K}_{2j}$ is identical to the pose of its corresponding keyframe in the global map, while the $\mathcal{K}_{1j}$ pose is updated by assuming a calibrated rigid transform to $\mathcal{K}_{2j}$. Since the map points in the local map are stored with absolute coordinates, their positions need to be updated individually. We do this by assuming unchanged relative translations to their current source keyframe.

6.4.2 The back-end

Overview

The back-end thread runs in an endless loop as illustrated in Fig. 6.3b. It is activated by the mapping thread whenever a new keyframe is added to the waiting queue of the global map. After a pose-graph optimization operation is done, the global map is updated according to the pose graph: The keyframe poses are updated by directly copying the corresponding vertex poses of the graph, leaving the associated map points only implicitly updated. The implementation of our PGO is based on the open source library g^2o described in Kummerle *et al.* (2011).

Adding local map to the back-end

In order to construct the global map, we add the local sub-map (including keyframes and map points) built by one specific camera to the back-end. BRIEF descriptors of feature points in the keyframes are computed in the back-end. We set the specific camera to be the forward-looking camera C_2, for three reasons. First, since the dual cameras are rigidly connected, pose constraints of one camera are sufficient for pose-graph optimization. Second, we mainly use our MAV in low-altitude applications. Thus, we expect

Table 6.1: MAV pose tracking RMSEs in the autonomous flight (Auto) and the manual flight (Manual) experiments using the proposed visual SLAM system, and using only the visual odometry (VO), with position errors in *millimeters* and attitude errors in *degrees*.

RMSEs	x	y	z	3D	roll	pitch	yaw
Auto	103.0	117.0	82.7	176.5	1.70	1.45	0.80
Manual	50.6	89.9	114.9	150.4	2.06	1.34	1.92
VO	69.2	168.2	106.7	206.9	2.37	1.35	2.28

more interesting views taken from the forward-looking camera, which will be used for loop-closure detection. Third, this is again a compromise for real-time performance: Keyframes from the downward-looking camera might later be included in the global map when the onboard computation capability significantly improves.

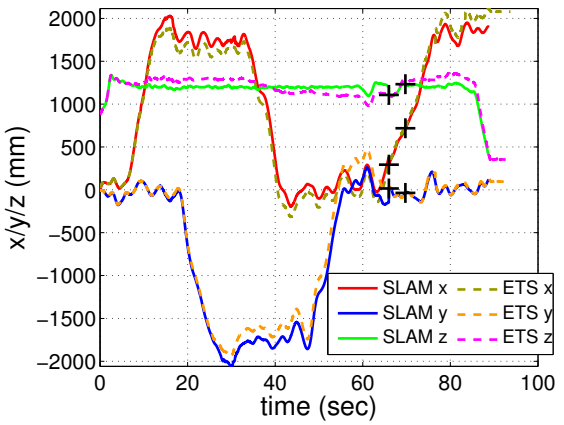
6.5 Experiments

In this section, we evaluate our visual SLAM system in an indoor environment (our robotics laboratory), as shown in Fig. 6.1b. The external tracking system available here provides accurate measures of the pose tracking errors of our onboard SLAM system.

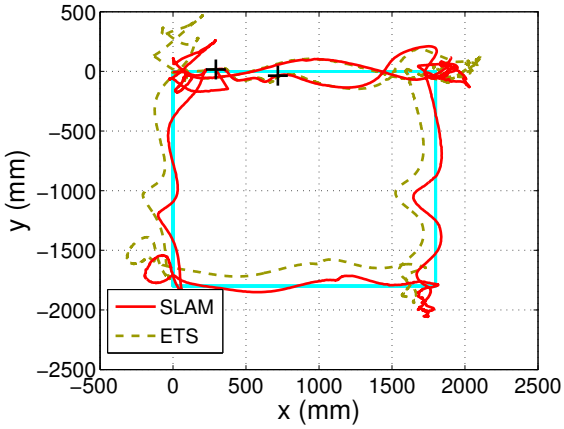
6.5.1 Enabling autonomous navigation

In this experiment, we demonstrate the efficiency of our SLAM system to enable autonomous navigation of our MAV. The MAV autonomously navigates along a predefined rectangular path with a height of 1.2 *m* (plotted in cyan in Fig. 6.4b) in a clockwise direction, taking off above the origin of the world frame and landing on the top-right corner. It turns 90 *degrees* at each corner in order to head to the forward direction.

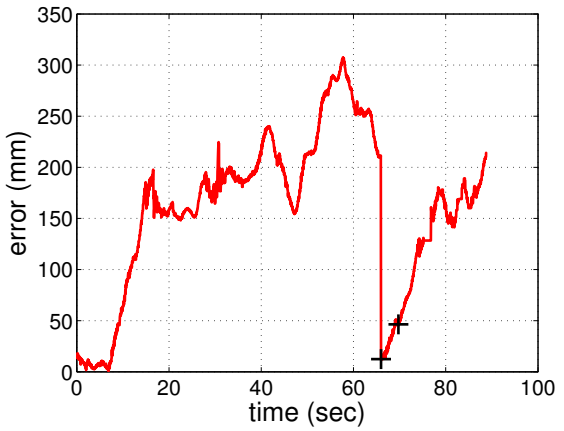
Fig. 6.4 shows the pose tracking results of the SLAM system (SLAM) compared with the ground truth data provided by the external tracking system (ETS). The visual odometry slowly drifts during explorations. However, the local-map update process will correct the drift after a loop closure is detected and the pose-graph optimization is performed. We can clearly recognize the effect of such corrections in Fig. 6.4: The black crosses are marked to the pose tracking results after local-map updates. The first loop-closure provides an obvious correction to the pose tracking of the visual odometry. Fig. 6.4c shows the 3D translation errors of the pose tracking results compared with the ground truth data during the flight. Those very short horizontal lines in Fig. 6.4c result from missing ground truth data during those periods of time, when the MAV is not flying in the effective field of view of the external tracking system. The RMSEs of the 6DOF pose



(a)



(b)



(c)

Figure 6.4: MAV pose-estimation results during the autonomous flight, compared with ground truth data: (a) MAV position estimation on x_W, y_W, and z_W axes, (b) on $x_W - y_W$ plane, and (c) the translation error. Pose corrections after loop closures been detected and PGO been processed, are marked with black crosses.

estimates of the SLAM system to the ground truth data during the whole flight are listed in Tab. 6.1 (in the row of Auto).

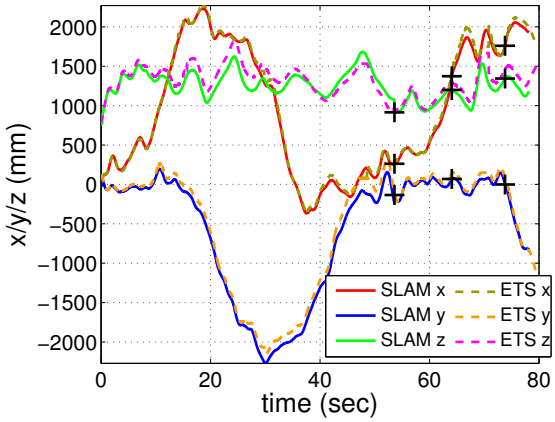
Pose estimates of a visual odometry are subjected to drift during explorations. In our case, two metric scale related issues which contribute to the pose drift should be noted: First, although dual cameras are used in our visual odometry, they have no overlap in their respective fields of view. Thus, the visual odometry cannot make stereo triangulation to track the metric scale of the environment, which results in scale drift in the pose estimation. Second, the accuracy of our automatic initialization module mentioned in Sec. 6.4.1 could be affected by the vibration of the MAV. The initialization errors in the metric scale and the attitude will be accumulated in the whole flight trajectory of the MAV. Another factor which could affect the pose tracking accuracy of the visual odometry is the errors in extrinsic calibration of the dual cameras.

6.5.2 Further evaluations with manual flight data

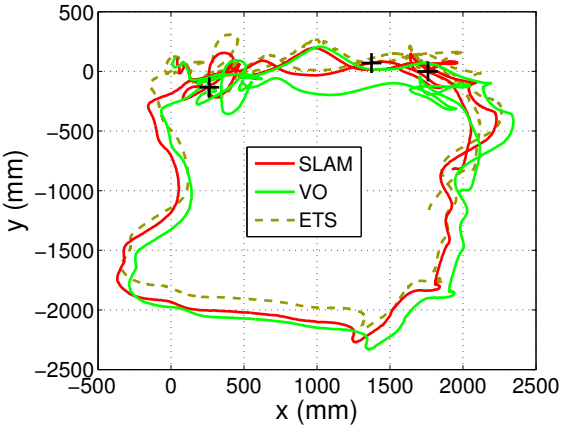
We manually control the quadrotor to fly in a similar way as we have done in Sec. 6.5.1, and record all necessary onboard data in a ROS-bag file. We process this logfile in post-processing on the onboard computer, to gain more insights into the performances of both the visual odometry and the back-end of the SLAM system. The results of this experiment are shown in Fig. 6.5 and Fig. 6.6.

Fig. 6.5a shows the pose tracking results of the SLAM system on x_W, y_W, and z_W axes of the world frame. Fig. 6.5b provides a top view of the MAV trajectory on $x_W - y_W$ plane. Here, we have processed the ROS-bag file twice using our SLAM system: firstly, using the full SLAM system (SLAM), and secondly, using only the visual odometry (VO) without the back-end. The results of the two processes are compared with pose estimates from the external tracking system (ETS). Fig. 6.5c shows the position estimation errors of these two processes. Without the back-end, the visual odometry would result in larger pose drift. Furthermore, loop-closure detection of the back-end can obviously benefit the pose tracking with drift corrections. RMSEs of the pose tracking results of the two processes are listed in Tab. 6.1 (in the row of Manual and the row of VO, respectively).

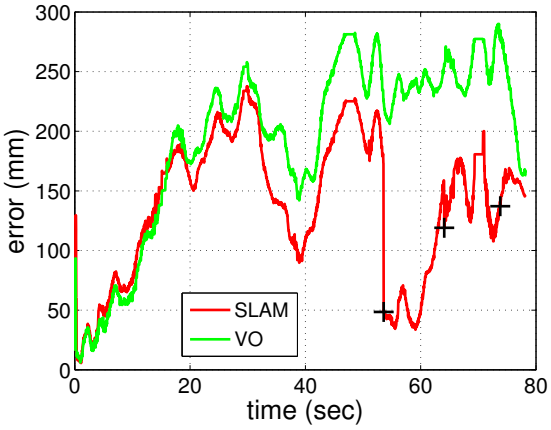
Fig. 6.6a and 6.6c provide more details in the performance of the visual odometry when running our full SLAM system. Time costs of the tracking thread (Tracking) and the mapping thread (LMapping) of the visual odometry are shown in Fig. 6.6a, which are largely affected by the number of map points in the local map, as shown in Fig. 6.6c. The pose tracking process is rather fast, with an average time cost of 0.0152 *second*. The bundle adjustment in the mapping thread is the most time intensive process, which only runs when new keyframes are added to the map, however. Time costs of both threads reach peaks when the MAV flies above the place where it takes off (with highly textured background). This results from the way we initialize the SLAM system: We use all feature corners (after non-maximum suppression) of the image which are sent by the initialization module for initialization, resulting in a very large number of map points in the local map if the background is highly textured.



(a)



(b)



(c)

Figure 6.5: MAV pose-estimation results using the proposed SLAM system and visual odometry during the manual flight compared with ground truth data: (a) MAV position on x_W, y_W, and z_W axes, (b) on $x_W - y_W$ plane, (c) the translation errors.

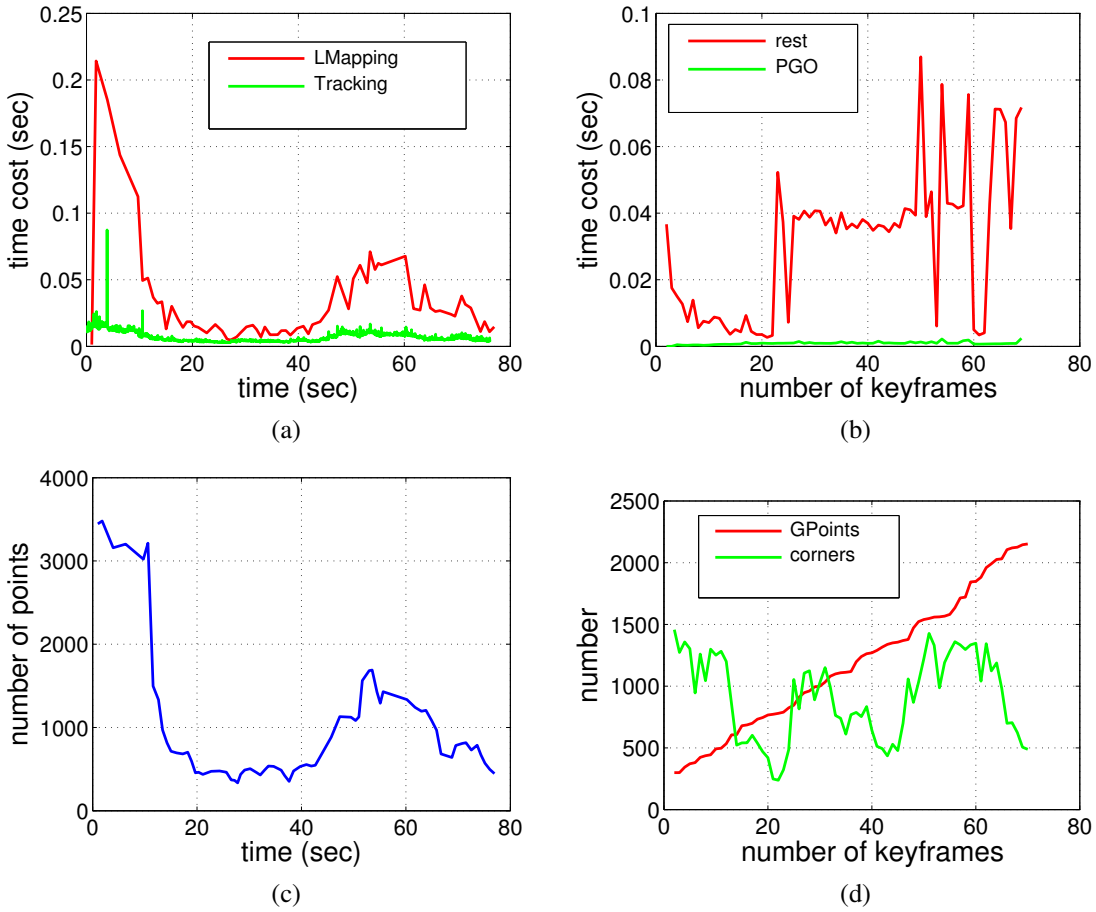


Figure 6.6: (a) Time cost of the tracking thread (Tracking) and mapping thread (LMapping) of the visual odometry during the exploration, and (c) the number of map points in the local map. (b) Time costs of the PGO process (PGO) and the other processes in the back-end (rest) when each new keyframe is added to the back-end. (d) Total number of map points in the global map (GPoints) and the number of feature corners of each new keyframe (corners).

Fig. 6.6b shows the time cost of the back-end process when each new keyframe is added to the global map: the pose-graph optimization (PGO) process, and the other processes of the back-end (rest) which mainly consists of the loop-closure detection and the preparation processes. Since this experiment is performed in a small area, the maximum time cost of the PGO process is only less than three milliseconds. Both the currently visible map points and the feature corners in the new keyframe can affect the time cost of the local loop closure and the appearance-based loop-closure detection. The time cost of the loop-closure detection and related preparation contributes the most to that of the back-end. Nevertheless, the maximal time cost is less than 0.1 *second*. The low time costs before 25 keyframes and after 53 keyframes are added can be explained mainly by two reasons: First, we ignore the most recent 15 keyframes in local loop closure detection. Second, if the geometric distance between the current keyframe and a previous keyframe is not within a certain range, the time-intensive keyframe registration process, as described in Sec. 6.3.2, will not be performed by the local loop closure detection process. Peaks of the time costs after 50 keyframes are added may happen if potential appearance-based loop closures are detected, when keyframe registration processes will be performed. Fig. 6.6d shows the total number of map points in the global map (GPoints), as well as the number of feature corners in each new keyframe (corners). The number of map points in the global map increases linearly during the exploration. However, their positions will only be implicitly updated, and thus they do not affect the real-time performance of the SLAM system.

Fig. 6.7 illustrates two views of the resulting global map of the SLAM system, with references of the real-world pictures. In this figure, we can find the pose graph with nodes (keyframe poses) and edges (in green), and map points at their absolute positions which are only computed for visualization purpose.

6.6 Conclusions and Discussions

Our proposed visual SLAM system utilizes two cameras to achieve a constant-time visual odometry, which can provide robust pose tracking for autonomous navigation of our MAV. The back-end of the SLAM system performs loop-closure detection and adaptive-window pose-graph optimization to correct pose drift of the visual odometry and to maintain a consistent global map. Autonomous navigation of an MAV following a predefined path has been achieved using the proposed visual SLAM system.

In general, doing PGO in the Similarity space Sim3, instead of SE3, can provide better scale corrections to the SLAM system (Strasdat *et al.*, 2010b), which could be a near future work. A map merging strategy would also be considered to achieve a more consistent map after loop closures. Then the performance of the resulting visual SLAM system in large-scale outdoor environments would be evaluated in the future. Currently, the keyframes and map points in the global map are not used in the visual odometry. It would be worthwhile to investigate how to more efficiently merge the back-end with

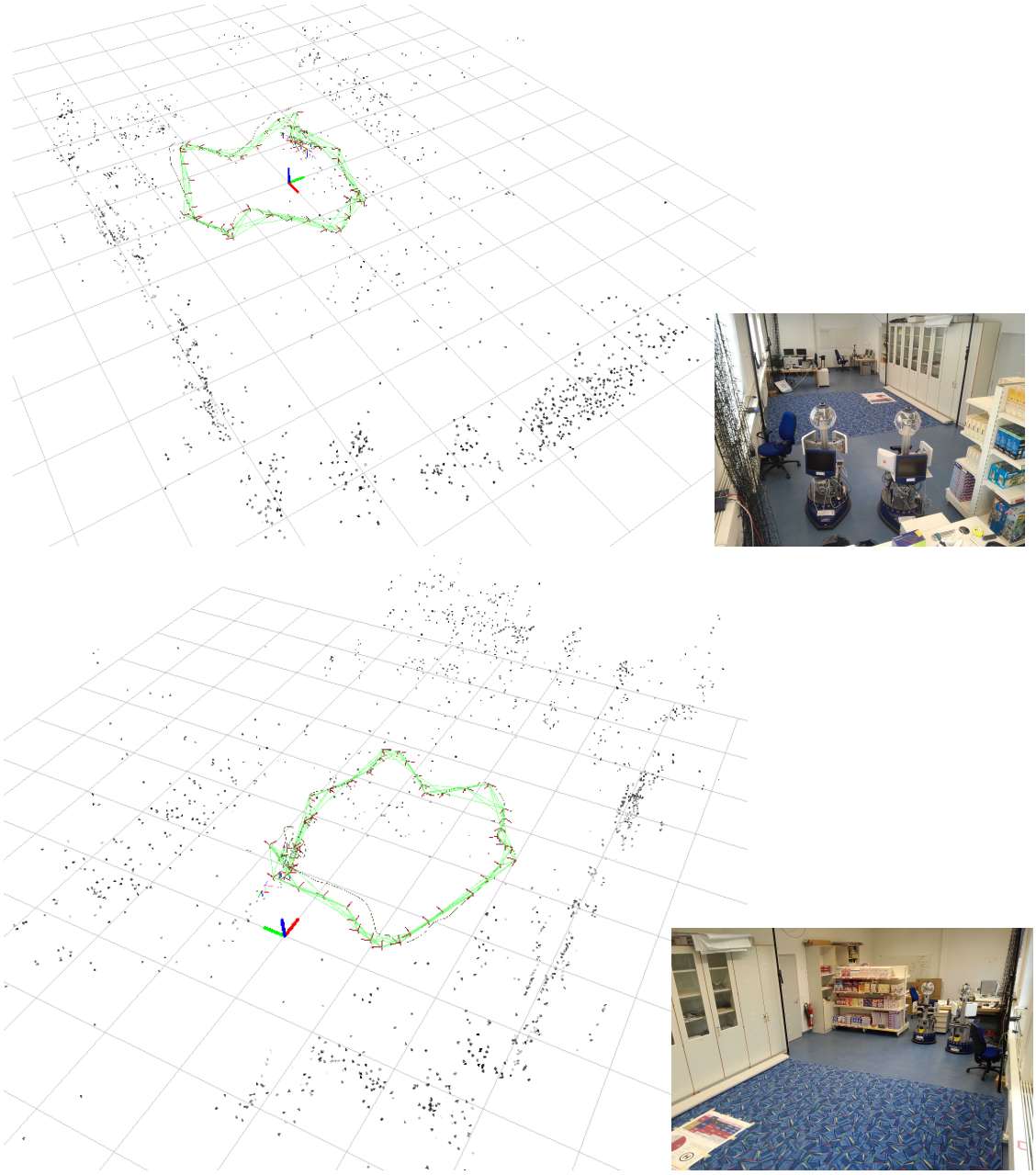


Figure 6.7: Two views of the global map built by the visual SLAM system during a manual flight in the lab. The grid cell size is $1m \times 1m$. The global map points are shown in grey-scale. The vertices of the pose graph are illustrated with red tri-axes, and the edges are plotted in green.

the visual odometry, so that the updated global map can be directly used by the visual odometry.

Chapter 7

Conclusions

7.1 Summary

This thesis deals with the visual SLAM problem for autonomous navigation of MAVs. The major concerns of this thesis have been the efficiency and pose tracking robustness of visual SLAM. We have worked on monocular visual SLAM and extended it to multi-camera visual SLAM in order to achieve more robust pose tracking. The presented visual SLAM systems provide robust pose estimates for an MAV to enable the autonomous flight in a previously unknown environment, and simultaneously build an environment map. Furthermore, these systems are efficient enough to be used onboard of MAVs which have limited payload and computational capability.

We started by solving the visual pose estimation based on artificial landmarks (visual markers) using a monocular camera in Chapter 3. The resulting vision system was used as a robust vision module for automatic initialization of the presented visual SLAM systems in this thesis. This vision system can also enable autonomous takeoff, hovering and landing of an MAV, provided that the visual marker is located within the field of view (FOV) of the camera. The general approach of this visual solution is using an artificial-neural-network (ANN) based method for robust visual-marker recognition, and then calculating the six degrees-of-freedom (6DOF) pose of an MAV based on a computational projective-geometry method.

In Chapter 4, by integrating an efficient ORB-feature-based object detection algorithm into a PTAM-based visual SLAM system, we have presented a monocular vision system enabling an MAV to autonomously search for and land on an arbitrarily textured landing site. The proposed visual solution enables the MAV to detect the landing site while navigating autonomously in unknown environments. After the landing site is detected, its location is estimated by a RANSAC-based method utilizing those map points (built by the SLAM system) associated with it, without the knowledge of its metric size. We have demonstrated the efficiency of the proposed system both in indoor and in challenging outdoor environments.

In Chapter 5, to improve pose tracking robustness of a monocular visual SLAM system in complex environments, we have extended PTAM to utilize measurements from multiple cameras, resulting in a robust multi-camera visual SLAM system. By analyzing

the optimization problems in pose tracking and map refinement in visual SLAM, we have provided the mathematical foundations for correctly integrating multi-camera measurements into those optimizations. The final implementation used a dual-camera setting, as a compromise between pose tracking robustness and time cost. Using the proposed visual SLAM system, we have achieved to enable autonomous navigation of an MAV in a challenging scenario, in which the pose tracking of a monocular visual SLAM system fails, and can build the environment map using measurements from both cameras.

Finally in Chapter 6, to work towards multi-camera visual SLAM for MAVs operating in large-scale environments, we have modified the SLAM system proposed in Chapter 5 to be a robust constant-time visual odometry, and have developed an efficient back-end to maintain the global environment map and to correct the pose drift of the visual odometry by performing loop-closure detection and pose-graph optimization (PGO). Two types of loop closures have been considered: local loop closures, which can be predicted using metric information, and global loop closures, which are detected using appearance information of images. An adaptive-window PGO method has been presented to efficiently utilize pose constraints from bundle adjustment and loop closures. Autonomous flight of an MAV following a predefined path has been achieved using the visual SLAM system proposed in this chapter.

To summarize, in this thesis, we have focused on developing robust and efficient visual SLAM systems to enable autonomous navigation of MAVs. We have developed an efficient monocular-visual-SLAM-based vision system for autonomous landing of an MAV on an arbitrarily textured landing site. The proposed multi-camera visual SLAM systems have been able to provide more robust pose tracking for autonomous MAVs than conventional monocular visual SLAM systems.

7.2 Future work

Although promising results have been obtained, our approaches still have their limitations and open questions. Those limitations and possible solutions related to the work in individual chapters have been discussed in the corresponding conclusion sections. In the following, we discuss more general questions and future research.

Since a monocular vision system cannot provide direct metric scale information of the environment, our monocular visual SLAM system will suffer from scale drift during explorations. A multi-camera system will have the same problem if the respective FOVs of the cameras share no overlap. A possible solution to improve the scale tracking accuracy is to fuse data from other sensors which can provide metric measurements, like fusing IMU data (Weiss *et al.*, 2012). However, considering that MAVs are normally equipped with low-cost IMUs which provide rather noisy data, more effort is required in order to achieve better scale tracking performance in the SLAM system. Fusing data from a laser scanner with visual SLAM results can be another option. The existing work on fusing GPS signal with laser scan has achieved promising results (Adler *et al.*, 2013). It

would be interesting to investigate fusing laser scans in visual SLAM systems. Another alternative solution is to add an asynchronous camera into the vision system with reasonable overlapping areas in the FOVs of the cameras, and then to do stereo triangulation in selective time instances similarly as the work in Shen *et al.* (2013a), in order to get depth constraints for certain visual features. An additional benefit of this solution is that the metric scale of the visual SLAM system can be initialized without external artificial landmarks.

Let us recall the three basic questions for autonomous navigation of a mobile robot (Leonard and Durrant-Whyte, 1991) that we introduced in Chapter 1: *"where am I?"*, *"where am I going?"* and *"how should I get there?"* Our multi-camera visual SLAM systems have been able to provide robust pose estimates for a robot, which answers the first question. They can also build an environment map consisting of a set of keyframes and a group of sparse 3D map points measured in those keyframes, which provides an understanding of the environment. For autonomous explorations of a mobile robot in unknown environments without predefined paths for safe navigation, we have to find further answers to the later two questions. This requires an efficient exploration strategy and sufficient environmental information. An exemplary work towards this goal can be found in the work of Keidar and Kaminka (2014), which presents an efficient frontier detection algorithm for robot exploration using data from laser scanners. Then an interesting future work rises: to investigate *how to effectively use our current map with sparse 3D points to facilitate those tasks*. A possible approach is to build an 3D occupancy grid map, like in Schauwecker and Zell (2014), based on the current map. However, since the 3D map points generated in our SLAM systems are rather sparse, the resulting grid map may not be sufficient to support the operations like obstacle avoidance. New approaches should be developed to effectively utilize the information in the maps generated by the visual SLAM systems.

Bibliography

Abbeel, P., Coates, A., and Ng, A. Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, **29**(13), 1608–1639.

ACFR (2014). Autonomous underwater vehicle (AUV) - sirius. `http://www.acfr.usyd.edu.au/research/projects/subsea/auvSIRIUS.shtml`. [Online; accessed 10-April-2014].

Achtelik, M., Achtelik, M., Weiss, S., and Siegwart, R. (2011). Onboard IMU and monocular vision based control for MAVs in unknown in-and outdoor environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3056–3063.

Adler, B., Xiao, J., and Zhang, J. (2013). Finding next best views for autonomous UAV mapping through GPU-accelerated particle simulation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1056–1061.

Angst, R. and Pollefeys, M. (2009). Static multi-camera factorization using rigid motion. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1203–1210.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, volume 1. springer New York.

Blanco, J.-L. (2010). A tutorial on SE(3) transformation parameterizations and on-manifold optimization. Technical report, University of Malaga.

Bleser, G. (2009). *Towards visual-inertial SLAM for mobile augmented reality*. University of Kaiserslautern.

Bouabdallah, S. (2007). Design and control of quadrotors with application to autonomous flying. *Ph.D. dissertation*.

Bouguet, J. (2001). Camera calibration toolbox for matlab.

Bradley, D. and Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of Graphics, GPU, and Game Tools*, **12**(2), 13–21.

Bradski, G. (2000). The opencv library. *Doctor Dobbs Journal*, **25**(11), 120–126.

Brown, D. C. (1971). Close-range camera calibration. *Photogrammetric engineering*, **37**(8), 855–866.

Bry, A., Bachrach, A., and Roy, N. (2012). State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1–8.

Cadena, C., Galvez-López, D., Tardos, J., and Neira, J. (2012). Robust place recognition with stereo sequences. *Robotics, IEEE Transactions on*, **28**(4), 871–885.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, pages 778–792.

Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **8**(6), 679 –698.

Carrera, G., Angeli, A., and Davison, A. (2011). SLAM-based automatic extrinsic calibration of a multi-camera rig. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2652–2659.

Castle, R. and Murray, D. (2011). Keyframe-based recognition and localization during video-rate parallel tracking and mapping. *Image and Vision Computing*, **29**(8), 524 – 532.

Castle, R., Klein, G., and Murray, D. (2010). Combining monoSLAM with object recognition for scene augmentation using a wearable camera. *Image and Vision Computing*, **28**(11), 1548 – 1556.

Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P., and Longhi, S. (2010). A vision-based guidance system for UAV navigation and safe landing using natural landmarks. In K. Valavanis, R. Beard, P. Oh, A. Ollero, L. Piegl, and H. Shim, editors, *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, U.S.A. June 8–10, 2009*, pages 233–257. Springer Netherlands.

Chen, Q., Wu, H., and Wada, T. (2004). Camera calibration with two arbitrary coplanar circles. In T. Pajdla and J. Matas, editors, *Computer Vision - ECCV 2004*, volume 3023 of *Lecture Notes in Computer Science*, pages 521–532. Springer Berlin Heidelberg.

Clemente, L. A., Davison, A. J., Reid, I. D., Neira, J., and Tardós, J. D. (2007). Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems*.

Collins, R. T., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., *et al.* (2000). *A system for video surveillance and monitoring*, volume 2. Carnegie Mellon University, the Robotics Institute Pittsburg.

Cummins, M. and Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, **27**(6), 647–665.

Cummins, M. and Newman, P. (2011). Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, **30**(9), 1100–1123.

Davison, A. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410 vol.2.

Davison, A. and Murray, D. (2002). Simultaneous localization and map-building using active vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **24**(7), 865–880.

Davison, A., Reid, I., Molton, N., and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **29**(6), 1052–1067.

Dayhoff, J. E. (1990). *Neural Network Architectures: An Introduction*. Van Nostrand Reinhold Co., New York, NY, USA.

Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part I. *Robotics Automation Magazine, IEEE*, **13**(2), 99–110.

Eade, E. and Drummond, T. (2007). Monocular SLAM as a graph of coalesced observations. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8.

Eberli, D., Scaramuzza, D., Weiss, S., and Siegwart, R. (2011). Vision based position control for MAVs using one single circular landmark. *Journal of Intelligent & Robotic Systems*, **61**, 495–512.

Engelhard, N., Endres, F., Hess, J., Sturm, J., and Burgard, W. (2011). Real-time 3D visual SLAM with a hand-held RGB-D camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, volume 2011.

Esquivel, S., Woelk, F., and Koch, R. (2007). Calibration of a multi-camera rig from non-overlapping views. In F. A. Hamprecht, C. Schnörr, and B. Jähne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 82–91. Springer Berlin Heidelberg.

Faessler, M., Mueggler, E., Schwabe, K., and Scaramuzza, D. (2014). A monocular pose estimation system based on infrared LEDs. In *IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014.*

Faig, W. (1975). Calibration of close-range photogrammetric systems: mathematical formulation. *Photogrammetric engineering and remote sensing*, **41**(12), 1479–1486.

Faugeras, O. (1993). *Three-dimensional Computer Vision: A Geometric Viewpoint*. Artificial intelligence. MIT Press.

Faugeras, O. D. and Toscani, G. (1986). The calibration problem for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 86, pages 15–20.

Fisher, P. S. F. (1994). Testing the convexity of a polygon. *Graphics gems IV*, **4**, 7.

Fitzgibbon, A., Pilu, M., and Fisher, R. (1999). Direct least square fitting of ellipses. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **21**(5), 476 –480.

Forsyth, D., Mundy, J., Zisserman, A., Coelho, C., Heller, A., and Rothwell, C. (1991). Invariant descriptors for 3D object recognition and pose. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **13**(10), 971 –991.

Forsyth, D. A. and Ponce, J. (2002). *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference.

Frahm, J.-M., Köser, K., and Koch, R. (2004). Pose estimation for multi-camera systems. In C. Rasmussen, H. Bülthoff, B. Schölkopf, and M. Giese, editors, *Pattern Recognition*, volume 3175 of *Lecture Notes in Computer Science*, pages 286–293. Springer Berlin Heidelberg.

Fraundorfer, F., Heng, L., Honegger, D., Lee, G., Meier, L., Tanskanen, P., and Pollefeys, M. (2012). Vision-based autonomous mapping and exploration using a quadrotor MAV. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4557–4564.

GA-ASI (2014). Predator B UAS. `http://www.ga-asi.com/products/aircraft/predator_b.php`. [Online; accessed 11-April-2014].

Gálvez-López, D. and Tardós, J. (2012). Bags of binary words for fast place recognition in image sequences. *Robotics, IEEE Transactions on*, **28**(5), 1188–1197.

Gao, X., Hou, X., Tang, J., and Cheng, H. (2003). Complete solution classification for the perspective-three-point problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **25**(8), 930–943.

Garcia-Pardo, P. J., Sukhatme, G. S., and Montgomery, J. F. (2002). Towards vision-based safe landing for an autonomous helicopter. *Robotics and Autonomous Systems*, **38**(1), 19 – 29.

Grzonka, S., Grisetti, G., and Burgard, W. (2009). Towards a navigation system for autonomous indoor flying. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2878–2883.

Harmat, A., Sharf, I., and Trentini, M. (2012). Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle. In *Proc. 5th International Conference on Intelligent Robotics and Applications*, volume 7506 of *Lecture Notes in Computer Science*, pages 421–432. Springer Berlin Heidelberg.

Harris, C. and Pike, J. (1988). 3D positional integration from image sequences. *Image and Vision Computing*, **6**(2), 87 – 90. 3rd Alvey Vision Meeting.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *the 4th Alvey Vision Conference*, volume 15, pages 147–151. Manchester, UK.

Hartley, R. and Zisserman, A. (2004). Cambridge University Press.

Hartley, R. I. (1994). Self-calibration from multiple views with a rotating camera. In J.-O. Eklundh, editor, *Computer Vision — ECCV '94*, volume 800 of *Lecture Notes in Computer Science*, pages 471–478. Springer Berlin Heidelberg.

He, L., Chao, Y., and Suzuki, K. (2008). A run-based two-scan labeling algorithm. *Image Processing, IEEE Transactions on*, **17**(5), 749 –756.

Heikkila, J. and Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106–1112.

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2014). RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In O. Khatib, V. Kumar, and G. Sukhatme, editors, *Experimental Robotics*, volume 79 of *Springer Tracts in Advanced Robotics*, pages 477–491. Springer Berlin Heidelberg.

Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., and Lepetit, V. (2011). Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 858–865.

HiSystems (2014). Mikrokopter. `http://www.mikrokopter.de/en/home`. Online; accessed 10-April-2014.

Hoffmann, G. M., Waslander, S. L., and Tomlin, C. J. (2008). Quadrotor helicopter trajectory tracking control. In *AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii*, pages 1–14.

Honda (2014). Asimo. `http://asimo.honda.com/`. [Online; accessed 10-April-2014].

Horaud, R. and Dornaika, F. (1995). Hand-eye calibration. *The International Journal of Robotics Research*, **14**(3), 195–210.

Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2011). Visual odometry and mapping for autonomous flight using an RGB-D camera. In *International Symposium on Robotics Research (ISRR)*, pages 1–16.

Huber, P. (2011). Robust statistics. In M. Lovric, editor, *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer Berlin Heidelberg.

Jimenez Lugo, J., Masselli, A., and Zell, A. (2013). Following a quadrotor with another quadrotor using computer vision. In *European Conference on Mobile Robots (ECMR 2013)*, Barcelona, Catalonia, Spain.

Kaess, M. and Dellaert, F. (2006). Visual SLAM with a multi-camera rig. Technical report.

Kaess, M. and Dellaert, F. (2010). Probabilistic structure matching for visual SLAM with a multi-camera rig. *Computer Vision and Image Understanding*, **114**(2), 286 – 296.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, **82**(Series D), 35–45.

Kanatani, K. (1991). Computational projective geometry. *CVGIP: Image Understanding*, **54**(3), 333 – 348.

Kanatani, K. and Wu, L. (1993). 3D interpretation of conics and orthogonality. *CVGIP Image Understanding*, **58**, 286–286.

Keidar, M. and Kaminka, G. A. (2014). Efficient frontier detection for robot exploration. *The International Journal of Robotics Research*, **33**(2), 215–236.

Kettnaker, V. and Zabih, R. (1999). Bayesian multi-camera surveillance. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages –259 Vol. 2.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan.

Knight, J. and Reid, I. (2000). Binocular self-alignment and calibration from planar scenes. In D. Vernon, editor, *Computer Vision — ECCV 2000*, volume 1843 of *Lecture Notes in Computer Science*, pages 462–476. Springer Berlin Heidelberg.

Krumm, J., Harris, S., Meyers, B., Brumitt, B., Hale, M., and Shafer, S. (2000). Multi-camera multi-person tracking for EasyLiving. In *Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on*, pages 3–10.

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613.

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

Lee, G. H., Faundorfer, F., and Pollefeys, M. (2013a). Motion estimation for self-driving cars with a generalized camera. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2746–2753. IEEE.

Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2013b). Structureless pose-graph loop-closure with a multi-camera system on a self-driving car. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 564–571.

Leonard, J. and Durrant-Whyte, H. (1991). Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, **7**(3), 376–382.

Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **28**(9), 1465–1479.

Li, H., Hartley, R., and Wang, L. (2005). Auto-calibration of a compound-type omnidirectional camera. In *Digital Image Computing: Techniques and Applications, 2005. DICTA '05. Proceedings 2005*, pages 26–26.

Lim, J., Frahm, J.-M., and Pollefeys, M. (2011). Online environment mapping. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3489–3496.

Lim, J., Frahm, J.-M., and Pollefeys, M. (2012). Online environment mapping using metric-topological maps. *The International Journal of Robotics Research*, **31**(12), 1394–1408.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, **60**(2), 91–110.

Lu, H. and Zheng, Z. (2010). Two novel real-time local visual features for omnidirectional vision. *Pattern Recognition*, **43**(12), 3938–3949.

Lu, H., Zhang, H., Yang, S., and Zheng, Z. (2010). Camera parameters auto-adjusting technique for robust robot vision. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1518–1523.

Lu, H., Yang, S., Zhang, H., and Zheng, Z. (2011). A robust omnidirectional vision sensor for soccer robots. *Mechatronics*, **21**(2), 373 – 389.

Luong, Q.-T. and Faugeras, O. (1993). Self-calibration of a stereo rig from unknown camera motions and point correspondences. Rapport de recherche RR-2014, INRIA.

Luong, Q.-T. and Faugeras, O. (1997). Self-calibration of a moving camera from point correspondences and fundamental matrices. *International Journal of Computer Vision*, **22**(3), 261–289.

Masselli, A. and Zell, A. (2012). A novel marker based tracking method for position and attitude control of MAVs. In *Proceedings of International Micro Air Vehicle Conference and Flight Competition*, pages 1–6, Braunschweig, Germany. DGON.

Masselli, A., Yang, S., Wenzel, K., and Zell, A. (2014). A cross-platform comparison of visual marker based approaches for autonomous flight of quadrocopters. *Journal of Intelligent & Robotic Systems*, **73**(1-4), 349–359.

Maybank, S. J. and Faugeras, O. D. (1992). A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, **8**(2), 123–151.

Mei, C., Sibley, G., Cummins, M., Newman, P. M., and Reid, I. D. (2009). A constant-time efficient stereo SLAM system. In *BMVC*, pages 1–11.

Mei, C., Sibley, G., Cummins, M., Newman, P., and Reid, I. (2011). RSLAM: A system for large-scale mapping in constant-time using stereo. *International journal of computer vision*, **94**(2), 198–214.

Meier, L., Tanskanen, P., Fraundorfer, F., and Pollefeys, M. (2011). PIXHAWK: A system for autonomous flight using onboard computer vision. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2992–2997.

Mellinger, D., Shomin, M., and Kumar, V. (2010). Control of quadrotors for robust perching and landing. In *Proceedings of the International Powered Lift Conference*.

Mellinger, D., Michael, N., and Kumar, V. (2012). Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, **31**(5), 664–674.

Merz, T., Duranti, S., and Conte, G. (2006). Autonomous landing of an unmanned helicopter based on vision and inertial sensing. In J. Ang, MarceloH. and O. Khatib, editors, *Experimental Robotics IX*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 343–352. Springer Berlin Heidelberg.

Michael, N., Mellinger, D., Lindsey, Q., and Kumar, V. (2010). The grasp multiple micro-UAV testbed. *Robotics & Automation Magazine, IEEE*, **17**(3), 56–65.

Michael, N., Shen, S., Mohta, K., Mulgaonkar, Y., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K., Ohno, K., Takeuchi, E., and Tadokoro, S. (2012). Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, **29**(5), 832–841.

Milford, M. (2013). Vision-based place recognition: how low can you go? *The International Journal of Robotics Research*, **32**(7), 766–789.

Milford, M. and Wyeth, G. (2012). SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1643–1649.

Mondragón, I. F., Campoy, P., Martinez, C., and Olivares-Méndez, M. A. (2010). 3D pose estimation based on planar object tracking for UAVs control. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 35–41. IEEE.

Murray, R., Li, Z., Sastry, S., and Sastry, S. (1994). *A Mathematical Introduction to Robotic Manipulation*. Taylor & Francis.

NASA (2014). Curiosity Mars rover. `http:www.nasa-usa.de/mission_pages/msl/`. [Online; accessed 10-April-2014].

Naturalpoint (2014). Optitrack. `http://www.naturalpoint.com/optitrack/products/tracking-tools-bundles`. Online; accessed 10-April-2014.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136.

Nicosevici, T. and Garcia, R. (2012). Automatic visual bag-of-words for online robot navigation and mapping. *Robotics, IEEE Transactions on*, **28**(4), 886–898.

Nonami, K., Kendoul, F., Suzuki, S., Wang, W., and Nakazawa, D. (2010). *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles*. Springer.

Oskiper, T., Zhu, Z., Samarasekera, S., and Kumar, R. (2007). Visual odometry system using multiple stereo cameras and inertial measurement unit. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8.

Pless, R. (2003). Using many cameras as one. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–587–93 vol.2.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3.

Ragab, M. E. M. (2008). *Multiple Camera Pose Estimation*. Ph.D. thesis, The Chinese University of Hong Kong (People's Republic of China). AAI3348872.

Raibert, M., Blankespoor, K., Nelson, G., Playter, R., *et al.* (2008). Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th IFAC World Congress*, pages 10823–10825.

Rosales, R. and Sclaroff, S. (1999). 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages –123 Vol. 2.

Rosten, E. and Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443. Springer.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE.

Saripalli, S. and Sukhatme, G. (2007). Landing a helicopter on a moving target. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2030–2035.

Saripalli, S., Montgomery, J., and Sukhatme, G. (2002). Vision-based autonomous landing of an unmanned aerial vehicle. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2799–2804.

Saripalli, S., Montgomery, J., and Sukhatme, G. (2003). Visually guided landing of an unmanned aerial vehicle. *Robotics and Automation, IEEE Transactions on*, **19**(3), 371 – 380.

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *Robotics Automation Magazine, IEEE*, **18**(4), 80–92.

Schauwecker, K. and Zell, A. (2013). On-board dual-stereo-vision for autonomous quadrotor navigation. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 333–342.

Schauwecker, K. and Zell, A. (2014). Robust and efficient volumetric occupancy mapping with an application to stereo vision. In *2014 International Conference on Robotics and Automation (ICRA'14)*, Hongkong, China.

Schauwecker, K., Ke, N. R., Scherer, S. A., and Zell, A. (2012). Markerless visual control of a quad-rotor micro aerial vehicle by means of on-board stereo processing. In *22nd Conference on Autonomous Mobile Systems (AMS)*, pages 11–20. Springer.

Scherer, S. and Zell, A. (2013). Efficient onbard RGBD-SLAM for autonomous MAVs. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1062–1068.

Scherer, S., Dube, D., and Zell, A. (2012). Using depth in visual simultaneous localisation and mapping. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 5216–5221.

Scherer, S. A., Dube, D., Komma, P., Masselli, A., and Zell, A. (2011). Robust real-time number sign detection on a mobile outdoor robot. In *Proceedings of the 6th European Conference on Mobile Robots (ECMR 2011)*, Örebro, Sweden.

Scherer, S. A., Yang, S., and Zell, A. (2014). DCTAM: Drift-corrected tracking and mapping for autonomous micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. Submitted.

Schilling, R. (1990). *Fundamentals of robotics: analysis and control*. Prentice Hall.

Sharp, C., Shakernia, O., and Sastry, S. (2001). A vision system for landing an unmanned aerial vehicle. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1720–1727 vol.2.

Shen, S., Nathan, M., and Kumar, V. (2011). Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 20–25.

Shen, S., Mulgaonkar, Y., Michael, N., and Kumar, V. (2013a). Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In *Robotics: Science and Systems (RSS)*. Citeseer.

Shen, S., Mulgaonkar, Y., Michael, N., and Kumar, V. (2013b). Vision-based state estimation for autonomous rotorcraft MAVs in complex environments. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Karlsruhe, Germany.

Siegwart, R., Nourbakhsh, I., and Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots*. Intelligent robotics and autonomous agents. MIT Press.

Sivic, J. and Zisserman, A. (2003). Video Google: a text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477 vol.2.

Slama, C. C., Theurer, C., Henriksen, S. W., *et al.* (1980). *Manual of photogrammetry.* Number Ed. 4. American Society of photogrammetry.

Solà, J., Monin, A., Devy, M., and Vidal-Calleja, T. (2008). Fusing monocular information in multicamera SLAM. *Robotics, IEEE Transactions on*, **24**(5), 958–968.

Strasdat, H., Montiel, J. M. M., and Davison, A. (2010a). Real-time monocular SLAM: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664.

Strasdat, H., Montiel, J., and Davison, A. J. (2010b). Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems*, volume 1.

Strasdat, H., Davison, A., Montiel, J. M. M., and Konolige, K. (2011). Double window optimisation for constant time visual SLAM. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2352–2359.

Strasdat, H., Montiel, J., and Davison, A. J. (2012). Visual SLAM: Why filter? *Image and Vision Computing*, **30**(2), 65 – 77.

Sunday, D. (2014). Inclusion of a point in a polygon. `http://geomalgorithms.com/a03-_inclusion.html`. [Online; accessed 12-April-2014].

Thrun, S. and Leonard, J. (2008). Simultaneous localization and mapping. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 871–889. Springer Berlin Heidelberg.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, **23**(9), 661–692.

Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment – a modern synthesis. *Vision Algorithms: Theory and Practice*, pages 153–177.

Tsai, R. and Lenz, R. (1988). Real time versatile robotics hand/eye calibration using 3D machine vision. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 554–561 vol.1.

Vicon (2014). Bonita. `http://www.vicon.com`. Online; accessed 10-April-2014.

Wagner, D. and Schmalstieg, D. (2007). Artoolkitplus for pose tracking on mobile devices. In *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, pages 139–146.

Weiss, S. and Siegwart, R. (2011). Real-time metric state estimation for modular vision-inertial systems. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4531–4537. IEEE.

Weiss, S., Achtelik, M., Lynen, S., Chli, M., and Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964.

Weiss, S., Achtelik, M. W., Lynen, S., Achtelik, M. C., Kneip, L., Chli, M., and Siegwart, R. (2013). Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics*, **30**(5), 803–831.

Weng, J., Cohen, P., and Herniou, M. (1992). Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**(10), 965–980.

Wenzel, K. E., Masselli, A., and Zell, A. (2010a). Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. *Journal of Intelligent & Robotic Systems*, **61**, 221–238.

Wenzel, K. E., Rosset, P., and Zell, A. (2010b). Low-cost visual tracking of a landing place and hovering flight control with a microcontroller. In *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, pages 297–311. Springer.

Wenzel, K. E., Masselli, A., and Zell, A. (2012). Visual tracking and following of a quadrocopter by another quadrocopter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pages 1–6, Vilamoura, Algarve, Portugal. IEEE. Accepted for publication.

Wikipedia (2014a). General Atomics MQ-9 Reaper — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=General_Atomics_MQ-9_Reaper&oldid=603501944`. [Online; accessed 11-April-2014].

Wikipedia (2014b). Wii remote — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Wii_Remote&oldid=596344373`. [Online; accessed 12-April-2014].

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardos, J. (2008). An image-to-map loop closing method for monocular SLAM. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2053–2059.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2009). A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, **57**(12), 1188 – 1197. Inside Data Association.

Wooden, D., Malchano, M., Blankespoor, K., Howardy, A., Rizzi, A., and Raibert, M. (2010). Autonomous navigation for BigDog. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4736–4741.

Wright, S. and Nocedal, J. (1999). *Numerical optimization*, volume 2. Springer New York.

Xiao, J., Adler, B., Zhang, J., and Zhang, H. (2013). Planar segment based three-dimensional point cloud registration in outdoor environments. *Journal of Field Robotics*, **30**(4), 552–582.

Xu, G., Zhang, Y., Ji, S., Cheng, Y., and Tian, Y. (2009). Research on computer vision-based for UAV autonomous landing on a ship. *Pattern Recognition Letters*, **30**(6), 600 – 605.

Yang, S., Scherer, S. A., and Zell, A. (2012). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. In *2012 International Conference on Unmanned Aircraft Systems (ICUAS'12)*, Philadelphia, PA, USA.

Yang, S., Scherer, S. A., Schauwecker, K., and Zell, A. (2013a). Onboard monocular vision for landing of an MAV on a landing site specified by a single reference image. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS'13)*, pages 317–324, Atlanta, GA, USA.

Yang, S., Scherer, S. A., and Zell, A. (2013b). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, **69**, 499–515.

Yang, S., Scherer, S. A., Schauwecker, K., and Zell, A. (2014a). Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision. *Journal of Intelligent & Robotic Systems*, **74**(1-2), 27–43.

Yang, S., Scherer, S. A., and Zell, A. (2014b). Robust onboard visual SLAM for autonomous MAVs. In *2014 International Conference on Intelligent Autonomous Systems (IAS-13)*, Padova, Italy. Accepted.

Yang, S., Scherer, S. A., and Zell, A. (2014c). Visual SLAM for autonomous MAVs with dual cameras. In *2014 International Conference on Robotics and Automation (ICRA'14)*, pages 5227–5232, Hong Kong, China.

Zell, A., Mache, N., Huebner, R., Mamier, G., Vogt, M., Schmalzl, M., and Herrmann, K.-U. (1994). SNNS (Stuttgart Neural Network Simulator). *Neural Network Simulation Environments*, pages 165–186.

Zhang, L. (2013). *Line Primitives and Their Applications in Geometric Computer Vision*. Ph.D. thesis, Christian-Albrechts-Universität Kiel.

Zhang, Z. (1997). Parameter estimation techniques: a tutorial with application to conic fitting. *Image and Vision Computing*, **15**(1), 59 – 76.

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666–673 vol.1.