

UNIVERSITY OF CRETE
FACULTY OF SCIENCES AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

Master Thesis

VISUAL UAV NAVIGATION

STAVROS TIMOTHEATOS

HERAKLION, NOV., 2017



University of Crete (UOC)
Department of Computer Science



FORTH
INSTITUTE OF COMPUTER SCIENCE
Computational Vision and Robotics
Laboratory (CVRL)

Master Thesis

VISUAL UAV NAVIGATION

STAVROS TIMOTHEATOS

HERAKLION, NOV., 2017

Visual UAV Navigation

Stavros Timotheatos

Thesis submitted in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Prof. *Panos Trahanias*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

UNIVERSITY OF CRETE
COMPUTER SCIENCE DEPARTMENT

Visual UAV Navigation

Thesis submitted by
Stavros Timotheatos
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author:



Stavros Timotheatos

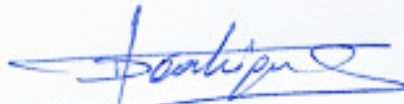
Committee approvals:



Panos Trahanias
Professor, Thesis Supervisor

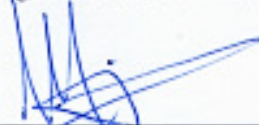


Antonios Argyros
Professor, Committee Member



Dimitris Tsakiris
Visiting Instructor, Committee Member

Departmental approval:



Antonios Argyros
Professor, Director of Graduate Studies

Heraklion, November 2017

Visual UAV Navigation

Abstract

Unmanned Aerial Vehicles (UAVs), particularly multi-rotor UAVs, have gained significant popularity in the autonomous robotics research field. The small size and agility of such aircrafts makes them an ideal candidate for use in restricted environments. With the ability to navigate freely in 3D space, UAVs offer the possibility to reach places that are otherwise inaccessible to humans. As such, UAVs find numerous applications in commercial and research tracks, such as inspection, surveillance, search and rescue (SaR) and aerial mapping. However, their remote operation is challenging for non-highly trained personnel, due to the fast dynamics of UAVs and the inherent difficulty to estimate distances to far away objects. In order for an autonomous UAV to safely and reliably navigate within a given environment, the control system must be able to determine the state of the UAV at any given moment. The state consists of a number of extrinsic variables, such as position, velocity and attitude of the UAV. The latter are commonly provided in outdoor operations via the Global Positioning System (GPS). While GPS has been widely used for long range navigation in open environments, its performance degrades significantly in constrained environments and is unusable indoors. Accordingly, state estimation for UAVs in the named environments is a popular, contemporary research area. Many successful solutions have been developed using laser-range finder sensors. These sensors provide very accurate measurements at the cost of increased power and weight requirements. Visual sensors constitute an attractive alternative state estimation sensor; they offer high information content per image coupled with light weight and low power consumption. As a result, many recent works have focused on state estimation on UAVs, where a camera is the only exteroceptive sensor.

In this thesis we aim at advancing the navigation capabilities of UAV systems, through the development of a framework that facilitates autonomous flights in previously unknown indoor and outdoor workspaces, using dual on-board cameras as the primary sensors. Those sensors introduce interesting challenges and need to be considered in the design of each part of the navigation framework. Therefore, the work presented in this thesis focuses on the problem of visual based navigation for UAVs. For the sake of robustness in autonomous flights, visual information is fused with measurements provided by an Inertial Measurement Unit (IMU), available on every UAV. The complementary nature of visual and inertial data renders this setup a particularly powerful sensor combination, albeit posing challenges in their synchronization and calibration. With this minimal sensory setup, in the current dissertation we have studied, implemented and evaluated methods for UAV control, state estimation, and autonomous navigation.

The first contribution of this thesis regards state estimation and control. We have shown that both parts can strongly benefit from a good model that provides valuable information about possible motions. Including the model in the state estimator, combined with a pressure sensor, renders the velocity and the two inclination angles observable. We present a versatile framework, using an Extended Kalman Filter (EKF) to fuse visual information with inertial measurements from the on-board sensor suite. Given that UAVs should be able to estimate their state from visual observations and also share acquired environment information with other members of the team, a common frame of reference for both positioning and observations is also derived.

For UAV control, we study the necessary dynamics and differential flatness of multi-rotor systems. We discuss potential abstractions in order to employ position- and trajectory controllers, on different types of multi-rotor platforms. Based on our investigation, we propose a control approach based on dynamic inversion, which avoids the commonly used attitude control loop and significantly reduces the mathematical operations necessary, rendering the approach appropriate for constrained on-board hardware.

Furthermore, we present a novel framework for horizon line (HL) detection that can be effectively used for UAV navigation. Our scheme is based on a Canny edge and a Hough detector along with an optimization step performed by a Particle Swarm optimization (PSO) algorithm. The PSO's objective function is based on a variation of the Bag of Words (BOW) method to effectively consider multiple image descriptors and facilitate computational efficiency. More specifically, the image descriptors employed are $L*a*b$ color features, texture features and PHOW-SIFT features.

Finally, a formulation of a visual navigation technique that enables the Asctec Firefly UAV to localize and navigate in outdoor environments is outlined. This is achieved through the use of a general, scalable, tracking and mapping system (SLAM). The overall goal of this dissertation has been to combine and integrate the above partial contributions, and enable

UAV-navigate in rather large, unknown outdoor environments. The latter implies that the UAV is stable flying on its own, while an operator at a ground station only gives high-level commands, such as simple way-points, desired velocities and dynamic trajectories. These outdoor tests prove the validity of the presented real-time and on-board navigation system. Also, they show the capability of the entire sensor-fusion framework to convert an aerial vehicle into a power-on-and-go system for real-world, large-scale and long-term operations.

The results in this thesis, showed that vision only navigation is possible as long as we are only concerned about the local consistency of our environment. The advantages of multi-fusion systems were demonstrated by switching between single and dual visual sensor setups and tested the different configurations with state noises and erroneous measurements. SLAM framework was able to initialize and to produce results in real-time on-board navigation, in self-similarity environments, where futures like grass in rural areas and asphalt in urban areas, causing the tracker to fail tracking the current pose, and the mapper to fail re-localizing. Furthermore the navigation system was shown to be able to operate, even with low resolution and distort image sequences from visual sensors, in contrast to other visual SLAM approaches.

To summarize, in the current dissertation we studied and developed an integrated framework to facilitate autonomous UAV navigation in large, unknown outdoor environments. Accordingly, robust and stable UAV flight has been accomplished, while an operator at a ground station only gives high-level commands, such as simple way-points, desired velocities and dynamic trajectories.

Οπτική Πλοήγηση Μη Επανδρωμένου Εναέριου Οχήματος

Περίληψη

Τα μη επανδρωμένα εναέρια οχήματα (UAV), ιδιαίτερα τα πολύ κινητήρια UAV, έχουν αποκτήσει σημαντική αποδοχή στον τομέα της έρευνας σε αυτόνομα ρομποτικά συστήματα. Το μικρό μέγεθος και η ευελιξία αυτών των αεροσκαφών τα καθιστά ιδανικά υποψήφια για χρήση σε περιβάλλοντα με περιορισμούς. Με την ικανότητα της ελεύθερης πλοήγησης τους στον τριδιάστατο χώρο, τα UAV προσφέρουν τη δυνατότητα πρόσβασης σε μέρη που είναι απρόσιτα για τον άνθρωπο. Ως εκ τούτου, τα UAV βρίσκουν πολυάριθμες βιομηχανικές και ερευνητικές εφαρμογές, όπως η επιθεώρηση και επιτήρηση χώρων, η έρευνα και διάσωση (SaR) και η εναέρια χαρτογράφηση. Ωστόσο, η απομακρυσμένη λειτουργία τους είναι μια πρόκληση για μη άρτια καταρτισμένο προσωπικό, λόγω των ταχέων δυναμικών κινήσεων των ΥΑ΄ και της εγγενούς δυσκολίας της εκτίμησης των αποστάσεων σε απομακρυσμένα αντικείμενα.

Για να καταστεί δυνατό ένα αυτόνομο UAV να πλοηγηθεί με ασφάλεια και αξιοπιστία σε ένα δεδομένο περιβάλλον, το σύστημα αυτόματου ελέγχου πρέπει να είναι σε θέση να καθορίζει το διάνυσμα κατάστασης του UAV σε οποιαδήποτε δεδομένη στιγμή. Το διάνυσμα κατάστασης αποτελείται από μια σειρά εξωγενών μεταβλητών, όπως η θέση, η ταχύτητα και η στάση (γωνίες Euler) του UAV. Τα τελευταία παρέχονται συνήθως σε υπαίθριους χώρους μέσω του παγκόσμιου συστήματος εντοπισμού θέσης (GPS). Ενώ το GPS έχει χρησιμοποιηθεί ευρέως για πλοήγηση μεγάλων αποστάσεων σε ανοικτά περιβάλλοντα, η απόδοσή του περιορίζεται σημαντικά σε περιορισμένα περιβάλλοντα και είναι ακατάλληλο για χρήση σε εσωτερικούς χώρους. Κατά συνέπεια, η εκτίμηση του διανύσματος κατάστασης των UAVs στα κατονομαζόμενα περιβάλλοντα είναι μια σύγχρονη και δημοφιλής ερευνητική περιοχή. Πολλές επιτυχημένες λύσεις έχουν αναπτυχθεί χρησιμοποιώντας αισθητήρες λέιζερ ευρείας απόστασης. Αυτοί οι αισθητήρες παρέχουν πολύ ακριβείς μετρήσεις με κόστος τις αυξημένες απαιτήσεις ισχύος και βάρους. Οι οπτικοί αισθητήρες αποτελούν έναν ελκυστικό αισθητήρα για εναλλακτική εκτίμηση του διανύσματος κατάστασης του UAV. Προσφέρουν υψηλό περιεχόμενο πληροφοριών ανά εικόνα σε συνδυασμό με μικρό βάρος και χαμηλή κατανάλωση ενέργειας. Ως αποτέλεσμα, πολλές ερευνητικές προσπάθειες έχουν επικεντρωθεί στην εκτίμηση του διανύσματος κατάστασης του UAV, όπου η κάμερα είναι ο μοναδικός αισθητήρας εξωτερικής αντίληψης.

Στην παρούσα διατριβή στοχεύουμε στην περαιτέρω προώθηση των δυνατοτήτων πλοήγησης των UAV συστημάτων, μέσω της ανάπτυξης μίας πλατφόρμας πλοήγησης που διευκολύνει τις αυτόνομες πτήσεις σε πρότερα άγνωστους εσωτερικούς και εξωτερικούς χώρους, χρησιμοποιώντας διπλές ενσωματωμένες κάμερες ως πρωτεύοντες αισθητήρες. Αυτοί οι αισθητήρες παρουσιάζουν ενδιαφέρουσες προκλήσεις που πρέπει να λαμβάνονται υπόψη στο σχεδιασμό κάθε μέρους του συστήματος πλοήγησης. Ως εκ τούτου, το έργο που παρουσιάζεται σε αυτή τη διατριβή επικεντρώνεται στο πρόβλημα της οπτικής πλοήγησης για UAVs. Για λόγους ευρωστίας σε αυτόνομες πτήσεις, οι οπτικές πληροφορίες συνδυάζονται κατάλληλα με μετρήσεις που παρέχονται από μια μονάδα μέτρησης της αδράνειας (IMU), που είναι διαθέσιμη σε κάθε (UAV). Η συμπληρωματική φύση των οπτικών και δεδομένων αδράνειας καθιστά αυτή τη ρύθμιση έναν ιδιαίτερα ισχυρό συνδυασμό αισθητήρων, αν και δημιουργεί προκλήσεις στον μεταξύ τους συγχρονισμό και την βαθμονόμηση τους. Με αυτή την ελάχιστη αισθητηριακή δομή, στην παρούσα διατριβή μελετήσαμε, εφαρμόσαμε και αξιολογήσαμε μεθόδους για τον αυτόματο έλεγχο (UAV), την εκτίμηση του διανύσματος κατάστασης και την αυτόνομη πλοήγηση.

Η πρώτη συμβολή της παρούσας εργασίας αφορά την εκτίμηση του διανύσματος κατάστασης και τον αυτόματο έλεγχο του (UAV). Έχουμε δείξει ότι και τα δύο αυτά μέρη μπορούν να επωφεληθούν έντονα από ένα καλό μοντέλο που παρέχει πολύτιμες πληροφορίες σχετικά με πιθανές κινήσεις. Συμπεριλαμβάνοντας το μοντέλο στον εκτιμητή του διανύσματος κατάστασης, σε συνδυασμό με έναν αισθητήρα πίεσης, καθιστά παρατηρήσιμη την ταχύτητα και των δύο γωνιών κλίσης. Παρουσιάζουμε ένα ευέλικτο σύστημα, χρησιμοποιώντας ένα **Extended Kalman Filter (EKF)** για να συνδυάσουμε (fuse) τις οπτικές πληροφορίες με τις μετρήσεις αδράνειας από την ενσωματωμένη συσκευή αισθητήρων. Δεδομένου ότι τα UAV θα πρέπει να είναι σε θέση να εκτιμήσουν το διάνυσμα κατάστασης τους από οπτικές παρατηρήσεις και επίσης να μοιραστούν πληροφορίες σχετικά με το περιβάλλον με άλλα μέλη της ομάδας, ένα κοινό πλαίσιο αναφοράς τόσο για την θέση όσο και για τις παρατηρήσεις εξάγεται συγχρόνως.

Για τον αυτόματο έλεγχο (UAV), μελετήσαμε τα απαραίτητα δυναμικά του συστήματος και τη διαφορική επιπεδότητα των συστημάτων πολλαπλών κινητήρων. Επίσης, προτείνουμε ελεγχτές θέσης και τροχιάς σε διαφορετικούς τύπους

πλατφορμών πολλαπλών κινητήρων. Με βάση την έρευνά μας, προτείνουμε μια προσέγγιση αυτόματου ελέγχου βασισμένη στη δυναμική αναστροφή (δύναμις ινερσιον), η οποία αποφεύγει τον συνήθως χρησιμοποιούμενο βρόχο ελέγχου της στάσης (γωνίες Euler) του UAV και μειώνει σημαντικά τους αναγκαίους μαθηματικούς υπολογισμούς, καθιστώντας την προσέγγιση κατάλληλη για το περιορισμένο ενσωματωμένο υλικό.

Επιπλέον, παρουσιάζουμε ένα νέο σύστημα για την ανίχνευση γραμμής ορίζοντα (HL) που μπορεί να χρησιμοποιηθεί αποτελεσματικά για πλοήγηση UAV. Η υλοποίηση μας βασίζεται στον Canny Edge Detector και σε έναν ανιχνευτή Hough μαζί με ένα βήμα βελτιστοποίησης που εκτελείται από έναν αλγόριθμο Particle Swarm Optimization (PSO). Η αντικειμενική συνάρτηση του PSO βασίζεται σε μια παραλλαγή της μεθόδου Bag of Words (BOW) για να εξετάσει αποτελεσματικά πολλούς περιγραφικούς δείκτες (descriptors) εικόνας και να βελτιώσει την υπολογιστική αποδοτικότητα. Πιο συγκεκριμένα, οι περιγραφόμενοι δείκτες εικόνας είναι χαρακτηριστικά χρώματος L^*a^*b , χαρακτηριστικά υφής και χαρακτηριστικά PHOW-SIFT.

Τέλος, περιγράφεται η ανάπτυξη μιας τεχνικής για οπτική πλοήγηση που επιτρέπει σε ένα Asctec Firefly UAV να εντοπίζει την θέση του και να πλοηγηθεί σε υπαίθρια περιβάλλοντα. Αυτό επιτυγχάνεται με τη χρήση ενός γενικού, κλιμακούμενου συστήματος παρακολούθησης και χαρτογράφησης (SLAM). Ο γενικός στόχος αυτής της διατριβής ήταν να συνδυάσει και να ενσωματώσει τις παραπάνω μερικές συμβολές και να επιτρέψει την πλοήγηση UAV σε αρκετά μεγάλα, άγνωστα υπαίθρια περιβάλλοντα. Το τελευταίο υποδηλώνει ότι το UAV είναι ικανό να εκτελέσει ευσταθείς πτήσεις αυτόνομο, ενώ ένας χειριστής σε έναν επίγειο σταθμό δίνει μόνο εντολές υψηλού επιπέδου, όπως απλά σημεία πλοήγησης (way-points), επιθυμητές ταχύτητες και δυναμικές τροχιές. Αυτές οι υπαίθριες δοκιμές αποδεικνύουν την εγκυρότητα του παρουσιαζόμενου ενσωματωμένου συστήματος πλοήγησης σε πραγματικό χρόνο. Επίσης, δείχνουν την ικανότητα ολόκληρης της πλατφόρμας συνδυασμού των αισθητήρων να μετατρέψει ένα εναέριο όχημα σε ένα σύστημα ποωερ-ον-ανδ-γο για πραγματικές, μεγάλης κλίμακας και μακρινές διάρκειας εναέριες επιχειρήσεις.

Τα αποτελέσματα σε αυτή τη διατριβή έδειξαν ότι η πλοήγηση μόνο με οπτικούς αισθητήρες είναι δυνατή εφ' όσον μόνο μας απασχολεί η τοπική συνοχή του περιβάλλοντός. Τα πλεονεκτήματα των συστημάτων πολλαπλού συνδυασμού αισθητήρων καταδείχθηκαν με τη εναλλαγή μεταξύ μονής και διπλής διαμόρφωσης των οπτικών αισθητήρων και δοκιμάστηκαν σε διαφορετικές καταστάσεις θορύβου και εσφαλμένων μετρήσεων. Το σύστημα SLAM ήταν σε θέση να αρχικοποιήσει και να παράγει αποτελέσματα για πλοήγηση σε πραγματικό χρόνο, σε περιβάλλοντα αρκετά όμοια χαρακτηριστικά, όπου χαρακτηριστικά όπως το γρασίδι και η βλάστηση σε υπαίθριες περιοχές και η ασφάλτος στις αστικές περιοχές, προκαλούν την αποτυχία της παρακολούθησης της τρέχουσας θέσης και στον χαρτογράφο την αποτυχία επανατοποθέτησης. Επιπλέον, το σύστημα πλοήγησης έχει δείξει ότι είναι σε θέση να λειτουργεί, ακόμη και με χαμηλής ανάλυσης και παραμόρφωσης των ακολουθιών εικόνων από οπτικούς αισθητήρες, σε αντίθεση με άλλες οπτικές προσεγγίσεις SLAM.

Συνοψίζοντας, στην παρούσα διατριβή μελετήσαμε και αναπτύξαμε ένα ολοκληρωμένο σύστημα για να επιτύχουμε την αυτόνομη πλοήγηση UAV σε μεγάλα, άγνωστα υπαίθρια περιβάλλοντα. Κατά συνέπεια, έχει επιτευχθεί εύρωστη και σταθερή πτήση UAV, ενώ ένας χειριστής σε σταθμό εδάφους δίνει μόνο εντολές υψηλού επιπέδου, όπως απλά σημεία πλοήγησης, επιθυμητές ταχύτητες και δυναμικές τροχιές.

Acknowledgments

The work presented in this thesis represents the completion of one of the biggest challenges of my life to date. I would not have reached this point without the guidance, understanding and support from all the people who have been involved in this endeavor over the last few years.

First of all, I would like to thank Professor Panos Trahanias for having offered me the possibility of doing a master thesis under his supervision in the Computational Vision and Robotics Laboratory (CVRL), Institute of Computer Science (ICS), Foundation for Research and Technology - Hellas (FORTH). He engaged me in a highly inspiring environment, provided all the necessary resources to achieve the present work, and guided me at all stages of my thesis. Prof. Trahanias's vision of robotics inspired me and paved the ground for lots of exciting opportunities. In addition, he greatly encouraged me to pursue my own ideas. Being part of prof. Trahanias's group at CVRL has been a highly appreciated experience and gave rise to many pleasant and interesting moments.


Furthermore, I'm extremely grateful to Professor Antonios Argyros for his advice and many helpful and inspiring discussions on computer vision and specifically on the important topic of horizon line detection. Also the essential contributions in particular in topics of control and estimation of Dr. Dimitris Tsakiris, CVRL/FORTH Principal Researcher, are greatly appreciated. I feel privileged that the mentioned scientists formed the committee that examined and approved my thesis.

Nowadays, a (scientific) work is rarely accomplished by a single person – neither was this work. In fact, without the support of many people, this work would never have been possible. The inspiration at CVRL was not only due to the high quality personnel and infrastructure, but also thanks to the friendly and pleasant atmosphere. I therefore wish to express my gratitude to all the current and past members, visitors, and students of CVRL. In particular, I want to express my warmest thanks to Stylianos Piperakis, PhD Candidate at CVRL, for the fruitful discussions we had and most importantly for his constructive feedback to my work. Also, I am grateful to Jason Oikonomidis, Postdoctoral Researcher at CVRL, for his help on critical aspects on horizon line optimization. Additionally, this work would not have been possible without the support on hardware and infrastructure. My thanks go to Thodoris Evdaimon who was always there to help on relevant issues. As UAVs are complex systems to fly with, I like to also thank Ammar Qammaz, MSc student at CVRL, who was on my side as an excellent safety-pilot.

Finally, I am deeply grateful to my family, my parents Spyros and Anni and my siblings, Eleni and Michalis, for all the support they gave me to accomplish this goal. Last but certainly not least, to my closest friends who supported me in every difficult and fun situation in the course of this dissertation.

The research conducted throughout my thesis has been supported in a multitude of ways from the Institute of Computer Science (ICS), Foundation for Research and Technology - Hellas (FORTH). This support is greatly appreciated and acknowledged.

Heraklion, November 2017



Stavros Timotheatos

στους γονείς μου

Contents

Table of contents	i
List of figures	iii
List of tables	ix
1 Introduction	1
1.1 Motivation and Objectives	1
1.2 Related Work	3
1.3 Contributions of the thesis	4
1.4 Structure of this Dissertation	5
2 Visual sensors for Robot Vision and Perception	7
2.1 Relating 3D Position to 2D visual sensor	7
2.2 Camera Calibration	10
2.3 Visual Pose Sensor: Structure from Motion	13
2.3.1 Perspective Multi-Point Approach	13
2.3.2 Visual front-end	14
2.3.3 Least Squares	15
2.3.4 Visual back-end	16
2.4 Corner Feature Detection	19
2.5 Feature Matching	20
3 Control and State estimation	23
3.1 UAV Motion Dynamics	23
3.2 UAV Motion Control	25
3.2.1 UAV Position Control	26
3.2.2 UAV Attitude Control - Inner Loop	30
3.3 Sensor Fusion	31
3.3.1 Extended Kalman Filter Framework	32
3.3.2 EKF Error State Representation	34
3.3.3 EKF Measurement Model	37
3.3.4 EKF Multi-Sensor Model	39
3.3.5 False Pose Estimation	42
3.3.6 EKF Design Model	42
4 Horizon Line Estimation	45
4.1 Computation of Candidate HLs	47
4.2 HL Derivation	48
4.3 Bag of Words - BOW	51
4.4 Derivation of Roll and Pitch Angles	53

5 UAV Visual Navigation	57
5.1 Localization Approaches	58
5.2 Map base Localization	59
5.2.1 VISUAL Odometry (VO)	61
5.2.2 VISUAL SLAM (VSLAM)	62
5.3 RT-PTAM - VISUAL SLAM	64
5.3.1 RT-PTAM CORE	64
5.3.2 RT-PTAM Key-frame Handling	67
5.3.3 RT-PTAM Robust Map	68
5.4 Horizon Line Navigation Fusion	70
5.4.1 Horizon Fault recovering	71
5.5 EKF Integration	72
6 Results	75
6.1 System setup	75
6.2 State Estimation Results	78
6.3 Horizon Line Results	81
6.3.1 Qualitative Results	83
6.3.2 Quantitative Results	84
6.4 UAV SLAM Navigation	87
6.4.1 RT-PTAM SLAM Results	88
6.4.2 Navigation Framework Results	90
7 Discussion and Conclusions	95

List of Figures

1.1.1	Various Autonomous Air Vehicle (UAV) platforms.	2
1.1.2	The dominant challenges for the successful application of UAVs navigation is robustness during agile and fast maneuvers, during illumination changes, and in scenes of difficult texture such as high-frequency or repetitive texture.	2
2.1.1	Camera perspective projection model, used for relating 3D position to position in 2D images. The rays converge on the origin of the camera frame C and a non inverted image is projected onto the image plane located at $z = f$. The z -axis intersects the image plane at the principal point which is the origin of the 2D image coordinate frame. The point $P = (X, Y, Z)$ is projected onto the camera image plane to the point $p(u, v)$. Left: Central projection model showing image plane and discrete pixels, the pixel coordinates are a 2-vector u, v with principal point as x_0, y_0 . Image plain is a $W \times H$ grid. The positive z -axis is the camera's optical axis. Right: Pinhole camera field of View (FOV) is the World area, that can be seen from the camera lens. It is represented as an angle. Typically the reported FOV from camera vendors is the diagonal field of view ($DFOV$). The FOV can be determined from the geometry point of view as horizontal ($HFOV$) and vertical ($VFOV$) with α the relative angle, which along with the $DFOV$ are coupling all camera field of Views based on equation $DFOV = \sqrt{HFOV^2 + VFOV^2}$	9
2.2.1	Pinhole camera Lens geometric distortion. Upper: Radial distortion causes image points to be translated along radial lines from the principal point. A: Negative radial distortion (pincushion). Pincushion distortion occurs when magnification increases with distance from the principal point and causes straight lines near the edge of the image to curve inward. B: Positive radial distortion (barrel). Barrel distortion occurs when magnification decreases with distance from the principal point which causes straight lines near the edge of the image to curve outward. Bottom: Tangential distortion, or decentralizing distortion, occurs at right angles and radius displacements but is generally less significant than radial distortion. C: Tangential distortion is caused by a displacement between camera plane and vertical plane with radius displacement as a result.	11
2.3.1	Perspective n-Point approach, mapping p_1, p_2, p_3 3D points to x_1, x_2, x_3 2D points using C_0 camera matrix relative transformation.	14
2.3.2	The re-projection error between the measure point location q_{ij} of an observed 3D point X_j , and the value predicted by the model $P(C_i, X_j)$	16
2.3.3	Bundle Adjustment approach. The name bundle adjustment comes from the representation of the problem, in which the lines between cameras and the observed points represent the bundles to be incrementally adjusted till an improved solution is found.	17
2.4.1	FAST corner detector example, here $r = 3$ and $n = 12$, forming a 12-corner detector. The dashed cyan arc highlights the pixels, which are brighter than point $p(x, y)$ and therefore p segment is indentified as a corner feature.	19
2.5.1	Image Pyramid, with five levels. Level 0 is the original image, in levels 1, 2, 3, 4 a smoothing filter is applied and the image is sub sampling each time.	21

3.1.1	Right: hex-rotor used in this work, with rotor rotation directions. Left: UAV body and IMU representation frame (cyan) with respect to world reference frame (red). A_I^w notes the full attitude of the IMU-centered UAV, while A_w^0 notes an intermediate frame (cyan) with a rotation of the yaw-angle ψ around z_w . The attitude angles are encoded in the rotation matrix A_I^w , as can be seen from Newton's second equation in 3.1.2, but we cannot directly invert this equation to solve for the attitude angles. In order to simplify the inversion, the 0-frame has been introduced as a leveled frame with the same ψ angle as the local body frame denoted by I	24
3.2.1	Structure of the position controller, [1]. Subscript "0" denotes coordinates with respect to the world frame, while subscript "B" denotes coordinates with respect to the current body frame. The names in braces denote on which physical device the corresponding part is executed. By the inversion shown, the desired attitude angles roll, pitch, and yaw $[\phi \ \theta \ \psi]_{des}^T$ and the thrust T_{des} are comprised. Outputs of the low-level attitude controller are the rotational velocities $[r_1^2, r_2^2, \dots, r_6^2]$ of the rotors.	26
3.2.2	Response of the reference model (equation 3.2.2) on an input step of $5m$, in this case a change in the desired position. Note that the model outputs a negative acceleration after $t = 3.5s$ to slow the vehicle down in advance in order to arrive at the desired position fast, but without overshoot. By transforming the input step to a ramp, we limited the velocity. Acceleration was limited by setting ω_0 . Setting bounds in this way, the reference models stability was not influenced, [2].	28
3.2.3	System overview: the computational demanding tasks are executed on the onboard computer, while the less computationally complex but time critical parts are executed on the high-level micro processor at a rate of 1 kHz. The EKF propagation part (see Section 3.2.1) provides a current state estimate and IMU bias corrections to the controller, which obtains its reference (Ref) and feed forward (FF) signals from the trajectory generator.	29
3.3.1	top: Standard (tightly design) EKF-SLAM approach exhibits computational complexity of at least $O(M^2)$ with M the number of features. bottom: In contrast, the proposed approach (loosely coupled design) runs at constant computational complexity. Moreover, by treating the pose estimation part as a black box, but still able to detect failures and drifts of the black box, our approach is independent of the underlying pose estimation method. The whole framework inherits thus the complexity of the chosen pose estimator, [3].	32
3.3.2	Setup depicting the UAV body with one sensor. For UAV navigation, we are interested in obtaining an estimate of p_I^w and q_{-I}^w denoted by the green arrow. To accomplish this, we compute auxiliary states denoted by the red arrow and measurements p_s^w and q_{-s}^w denoted by the blue arrow. Depending on the type of sensor, only one of the measurements may be available, for example 6 DoF pose sensors measures both position and attitude, in contrast 3 DoF pose sensors, only return the position. The additional states do not have any dynamics during the propagation phase.	38
3.3.3	Full coordinate frame setup for multiple sensors, and with multiple measurements, e.g. two vision algorithms Mes_{f1} (bright green) and Mes_{f2} (dark green), having different reference frames, can use the images from the same sensor Sensor 1 (orange), which has a calibration with respect to the IMU p_I^s and q_I^{-s} , (cyan dash line). In contrast with system in figure 3.3.2, the measurements vectors point from the sensor to its measurement frame, since this frame is moving from the point of view of the sensor. With multi-sensor setup we can combine sensors or algorithms arbitrarily. For instance, a known visual feature, the Mes_{f1} (bright green) measurement reference frame, may be seen by both sensors.	41
4.0.1	Rover localization, [4].	45
4.1.1	Computation of candidate Horizon Lines. The input is an <i>RGB</i> image. A Canny Edge detector is applied, with a Hough transform following. The Output is the set of candidate HLs. HLs subsets, particularly \mathbb{S}_{SP} , $\mathbb{S}_{P_{set}}$ and \mathbb{S}_{RP} , are produced based on P_H CDF distribution and Hough space points.	48

4.1.2 Hough Transform with polar variables ρ as the radius and θ as the angle. The (x,y) are the 2D space coordinates. Basic Principle of Hough. Upper image: (a) shows representation on equation of straight line, (b) illustrates intersection of many lines to a point, (c) Transformation of point in image space to polar space. Bottom image: Working of Hough as feature extractor. (a) Shows a set of points in image space, (b) illustrates convergence of points in image space to sinusoidal waves in polar space, (c) represents the accumulator space. The limits of the Hough transform are $[-D,D]$ for ρ , where D is the diagonal of the input image and θ lies in $[-89^\circ, 90^\circ]$	49
4.1.3 Schematic description of HL derivation. PSO objective is based on χ -square distance, computed by the two image parts.	50
4.3.1 A key-points BOW approach. Denoted by small circles (blobs), key-points are usually around the corners and edges in image objects, such as the edges of the map and around people's faces. A SIFT descriptor can automatically detect regions/points of interest and compute local descriptors over those regions/points. Then, the descriptors are quantized into visual words to form the visual vocabulary.	51
4.3.2 BOW visual words. The occurrences of each specific visual word in the image are computed, constructing the BOW histogram, based on visual word frequencies.	51
4.3.3 BOW formulation. Color- $L*a*b$, texture and SIFT-Phow descriptors, vocabulary and words are extracted from input image.	52
4.4.1 The point at infinity in the direction of the principal axis P always projects on the principal point p_0 if $\theta_{pitch} = 0$ holds. Additionally, rotation of the unit normal n_0 around the coordinate system is shown. The ground equation at any instance can be described by rotating the canonical unit normal $n_0 = (0, 1, 0)^T$ around the camera coordinate system $\alpha (\phi_{roll})$ and $\beta (\theta_{pitch})$, where $Z = a + bX + Y$ is the equation of the ground plane in the camera coordinate system, where $a = -h/(\sin(\theta_{pitch}) \cos(\phi_{roll}))$, $b = \tan(\phi_{roll})/\sin(\theta_{pitch})$ and $c = -1/\tan \theta_{pitch}$, [5].	54
4.4.2 Pitch and roll angles detection. The ϕ_{roll} angle is readily computed from the horizon angle. The θ_{pitch} angle is the angle between the camera optical axis (cyan line) and the principal axis (black dashed line) in camera image plane. In addition in Horizon line plane, the roll angle ϕ_{roll} is the angle between the HL_{Img}^0 (UAV in ground level) and HL_{Img}^1 horizon lines. The pitch θ_{pitch} is the angle between the HL_{Img}^0 and the line joining the camera center S with point p_1 , where the position of point p_1 is determined by the intersection of horizon line HL_{Img}^1 with the image vertical line (green dashed line), which passed through the p_0 principal point. The p_0 is detected by HL_{Img}^0 line, when camera is initially leveled with the ground (i.e. $\theta_{pitch} = 0, \phi_{roll} = 0$).	55
5.0.1 Navigation framework with multiple sensors. Sensor 1: A monocular visual sensor and the primary RT-PTAM is used. Sensor 2: Second monocular visual sensor, where a combined Horizon Line detection and RT-PTAM is utilized. Auxiliary sensors: May include a pressure sensor (presented in section 3.3.4, where we additional mentioned that is relative easy to include additional sensors if available), or a optical flow module. The EKF is then responsible for the sensors measurements fusion and is updating the state estimation to provide UAV micro-controller with the updated state and velocities, enabling real time navigation.	58
5.2.1 A simple example of a loop closure situation. The UAV's trajectory is shown as a solid line and it's own trajectory estimate is given as a dashed line. In this example, a visual sensor equipped in a UAV follows the trajectory shown building a map of visual features, the stored sensor poses or key-frames are indicated as circles. The accumulated errors in the UAV's position estimate, give rise to a drift for both the current position estimation and the key-frame position, with respect to the ground truth (solid line). To correct this the system must be able to compute the transformation (the loop closure correction) required to realign the map. This correction can then be back propagated throughout the map to correct all key-frame positions.	62

5.3.1 RT-PTAM flowchart, which illustrates its working procedure is consists by three threads, namely Extraction, Tracking and Mapping. A frame stream acquired by the visual sensor on-board a UAV, records the scenes of a relatively large region. The acquired image is processed to generate a pyramid key-frame, containing multiple levels of the frame at different resolutions. In order to be able to track the camera and map new points, a base map initialization phase is necessary. This is achieved using a standard five-point stereo algorithm between two key-frames, developed in [6]. The mapping phase of the RT-PTAM starts immediately after the map initialization process has completed and runs parallel along the tracking part. Mapping thread is responsible for providing the map, and optimizing the map using local and global BA with multiple loops runs, until the resulting map to be as detail and as accurate as possible. However, map optimization is most of the time a very slow procedure. In contrast, the tracking thread is implemented to be fast and robust, since its main task is the camera pose accurate computation, which is constantly needing in real time navigation.	65
5.3.2 RT-PTAM SLAM scheme. The top picture is a RT-PTAM’s visual SLAM algorithm real-time visualization. The tracking of the FAST corners can be observed. This is used for the localization of the camera. In the bottom picture, the 3D map built by the mapping thread is shown. The three-axis coordinate frames represent the location where new key frames were added. Each key-frame consists of a 6 DoF camera pose, the camera image and its down-sampled pyramidal levels schematically depicted on the bottom right.	66
5.3.3 RT-PTAM SLAM mapping and tracking threads time representation. Mapping thread can take a great amount of time, in contrast tracking thread is more robust and computed in 20-30 Hz. Map is not updated in every frame.	67
5.3.4 RT-PTAM and base RT-PTAM SLAM features detection. Right image: In base RT-PTAM the detected features of high-frequent, and self-similar structures are extracted in the lowest pyramidal level 0. Since most details are preserved in this level, the search radius calculated from the algorithm’s motion model (blue circle) is covered several potential matches which leads to a wrong data-association. Left image: RT-PTAM, down-sampling the original image (i.e. higher pyramidal level 1) and acts as a low pass filter. Only features of low-frequent and self-similar structures are extracted and the motion model can disambiguate well between the different features.	68
5.3.5 RT-PTAM feature detected for each pyramid level. Right image: shows the ratio of outliers that RT-PTAM reported after optimization, with respect to the number of features detected. Left image: Feature detection in a typical image taken by UAV onboard camera during outdoors navigation. The colors of the features points are based on the pyramid level that are detected [3].	69
5.3.6 Example of level 0 features extracting significance. Pyramidal levels 1-3, that RT-PTAM uses for feature detection and map are corresponded to a half-sampled version of the original camera image level 0. Since features on level 0 are unstable, we do not include them into the RT-PTAM map. However RT-PTAM is still tracked them as the camera is moved away from the feature, in particularly a feature (red point) was prior detected on most resent key-frame level 1 (black pyramid), and since camera was moved upwards, the red point is reappeared on the current key-frame (green pyramid) level 0. If level 0 feature extraction is disabled, we are not be able to find this previously mapped point feature.	70
6.1.1 Asctec firefly hex-rotor UAV.	76
6.1.2 Asctec firefly internal system design. Left: The firefly CAD model illustrating the vibration damping between the two parts of the frame: the motors and the landing gear are connected to the outer UAV frame, and the inner frame to the IMU, battery, and payload. The silicon dampers are highlighted in red. Right: The redundancy against single-rotor failure is presented. Assuming that motor 1 is failing, motors 2, 3, 5, and 6 are controlled by the thrust command and the roll and pitch output. Motor 4, on the opposite side of the failing motor compensates and controls the yaw momentum by repeatedly changing its direction, [1], [7].	76

6.1.3 Overview of the data flow, interfaces and electronic architecture of the firefly, provided by Asctec, [7]. All sensors, except the GPS, are connected to the LLP, which communicates via the serial peripheral interface (SPI) with the HLP and with a I2C to the motor controllers. SPI has the advantage of very low delay, so HLP can access rotor speeds, angular rates, linear acceleration and estimated attitude from the LLP very fast. The LLP position and way-point control module, is implemented with only GPS based control, and will therefore not be used in this work, since we work in GPS-denied environments.	77
6.1.4 Thesis system overview: the computational demanding tasks are executed on the on-board computer, while the less intensive and time critical parts are executed on the high-level micro processor at a rate of 1 kHz. Way-point commands are sent from the ground station to the on-board computer, which forwards them to the position controller. The ground station and on-board computer communicate over a Wi-Fi connection. Note that all critical parts necessary to keep the UAV airborne run entirely on board and do not rely on the Wi-Fi link.	78
6.2.1 Representative image of the machine hall, where the EuRoC micro aerial vehicle dataset was collected.	79
6.2.2 Position estimation comparison of NSS (red line), TS (blue line) and NDS (green line) sensors configurations and datasets position ground truth (black line); Left column: Full frame rate deployment; Right column: Specific frames comparison.	80
6.2.3 Attitude estimation comparison of NSS (red line), TS (blue line) and NDS (green line) sensors configurations and datasets position ground truth (black line); Left column: Full frame rate deployment; Right column: Specific frames comparison.	81
6.2.4 Position estimation mean error of NSS, TS, NDS sensors configurations with respect to datasets ground truth.	82
6.2.5 Attitude estimation mean error of NSS, TS, NDS sensors configurations with respect to datasets ground truth.	82
6.2.6 A 3D navigation trajectory comparison of NSS (red line), TS (blue line) and NDS (green line) sensors configurations and datasets ground truth (black line) trajectory.	83
6.3.1 HL detection. Top row: sky-sea image; bottom row: sky-ground image. Left column: Hough-HL; right column: PSO-HL.	84
6.3.2 HL detection in a challenging image. Left column: Wide angle orientation; right column: 90 deg orientation, Hough-HL (red) and PSO-HL(green).	84
6.3.3 HL detection in the case of a moving camera, Left column: Hough-HL; right column: PSO-HL.	85
6.3.4 HL detection in the case of a UAV on-board moving camera, Left column: Camera distortion based on logitech C920 rolling shutter ; right column: IMU measurements frame are translated to HL measurements frame PSO-HL, Hough-HL (red), PSO-HL(green), IMU-untraslated(blue) and IMU-translated(black)	85
6.3.5 HL detection accuracy. <i>Hough – HL</i> : (red bar) 39%, <i>HL-Tree</i> : (black bar) 33%; <i>V-HL</i> : (magenta bar) 56%, <i>CNN-HL1</i> (single CNN):(cyan bar) 62.66%; <i>CNN-HL2</i> (multiple CNN):(blue bar) 82.66% , and <i>PSO – HL</i> :(green bar) 77.66%.	86
6.3.6 Outdoor experiments UAV with dual camera set-up; Firefly.	87
6.3.7 Roll angle estimation: IMU (blue line), PSO-HL (red line), Hough-HL (green line).	87
6.3.8 Pitch angle estimation: IMU (Blue line), PSO-HL (Red line), Hough-HL (Green line).	88
6.4.1 KITTI datasets trajectory mean error of LSD-SLAM ORB-SLAM2, SVO, SVO2, PTAM and PTAM2 with respect to datasets ground truth. Upper-Left: Position x error, Upper right: Position z error and Bottom: 2D position error.	89
6.4.2 KITTI datasets navigation trajectory based on SLAM algorithms. LSD-SLAM (cyan line), ORB-SLAM2 (green line), SVO and SVO2 (red and blue line respectively), PTAM (magenta line) and PTAM2 (orange dashed line) and ground truth (black line). Left: Full navigation trajectory, Upper right: Specific navigation frames comparison	89
6.4.3 Representation of outdoors navigation experiments in complex realistic field area, acquired with on-board UAV visual sensors.	90
6.4.4 Overall UAV navigation initialization process.	91

6.4.5 Initialization process of RT-PTAM(blue line), NAV-EKF(red line) and Ground Truth(black line) in x,y and z axes. A dataset of 180 sec was used. Upper: Possition- $x(m)$, Mid: Possition- $y(m)$ and Bottom: Possition- $z(m)$	92
6.4.6 UAV homing experiment. Red line: Forward navigation trajectory. Blue line: Return trajectory.	93
6.4.7 UAV homing experiment visualizer. Left column: ROS RVIZ ; Right column: RT-PTAM visualizer. . . .	94

List of Tables

6.1	Root mean square error for single noisy sensor (NSS), optimized single sensor (TS) and dual sensor set up.	79
6.2	KITTI datasets, root mean square error for RT-PTAM, PTAM, SVO, SVO2, LSD, ORB2 SLAM with respect to the datasets ground truth.	90
6.3	Visual Navigation, Root mean square error for PTAM, RT-PTAM and Navigation framework, with respect to ground truth (ORB-SLAM).	94

Chapter 1

Introduction

Mobile robots and autonomous air vehicles (UAV) are poised to become a central part of our everyday lives. Mobile robots can improve quality of life and offer unprecedented utility in all sorts of applications. In addition, robots can explore remote or hazardous areas, transport goods, or perform manual labor such as cleaning, farming, and construction. Moreover, there are numerous applications for UAVs such as: Search and Rescue (SaR), aerial inspection, exploration, aerial photography and conservation activities, including wildlife or crop monitoring. Several companies, e.g. Google and Amazon, have commenced large development of UAV for tasks such as delivering medical supplies to remote locations as well as commercial goods in urban environments. Such applications have one thing in common, a requirement for a robust and reliable navigation system. In contemporary UAV deployments, when operated by trained pilots and with the help of GPS, photographs can be taken from locations that are neither reachable by full-scale helicopters and fixed-wing aircraft nor by camera-cranes. Possible applications include aerial photographs or videos from sport events or from buildings. However, given that SaR is an ideal application for UAVs, their actual deployment in real SaR situations is surprisingly infrequent [8]. In general it comes down to a lack of trust in the autonomous systems; the pilots do not feel comfortable in delegating control to an autonomous navigation system even in cases where the autonomous system is capable of achieving the current objectives. This is evidenced by the fact that, while there is increasing use of UAVs in real world applications, these tend to be either manually controlled or in highly structured settings where lack of advanced capabilities such as collision avoidance and scene perception are not required.

One major difficulty in multi-rotor research is the significant investment of both time and capital required to set-up a safe, reliable framework with which to conduct research. Some of the most common UAV research platforms are the AscTec [7] line of multi-rotor UAVs. These platforms are, by no means complete solutions as additional sensors and on-board computers are still required, and also UAVs do not include any robust emergency recovery system. In addition, It's generally common to include an emergency stop button on a ground robot which will immediately cut-off power to the actuators in the case of a runaway robot. However this is not possible with a UAV, since cutting power to the actuators on a UAV, in practice guarantees significant damage to the UAV. Therefore, with the inclusion of a navigation system the safety of the craft immediately improves.

1.1 Motivation and Objectives

Autonomous flight and navigation in air-ground systems is a large field of study which has significant room for improvements in terms of the algorithms and systems used. It is also a field that is rapidly expanding with the design and deployment of new state of the art algorithms. But how can something, that is actually heavier than air, fly? This is probably the question that triggers the immense fascination of flying amongst many people. In addition, modeling and designing UAV, produced enormous interest, particularly as the UAV are become larger, more sophisticated and technological advance, which enhanced people seeking for further challenges. Even though the autonomous flight idea, theoretically renders a trained pilot useless, there have always been more persistent questions than, not only of why these UAV fly, but how can we actually achieve these UAVs, to fly autonomously.

For several decades research UAVs has been dominated by the military and aerospace industries. However, nowadays,

research groups and industry are overwhelmingly interested in research and development endeavors with UAVs, and deploying these systems in real time scenarios, where it would otherwise be too dangerous, time-consuming or even impossible for humans to physically intervene. The barrier for entry into these fields being the ability to deploy and support a large unmanned aircraft. However with advances in computer science, sensor and battery technologies, powered largely by the mobile phone market, it has become possible to develop small size, low weight and inherent safety UAVs. These UAVs come in a variety of configurations from fixed-wing aircraft capable of long duration, high altitude flight, to highly stable and maneuverable rotary-wing craft such as quadcopters, hexacopters and octocopters UAVs, shown in figure 1.1.1.

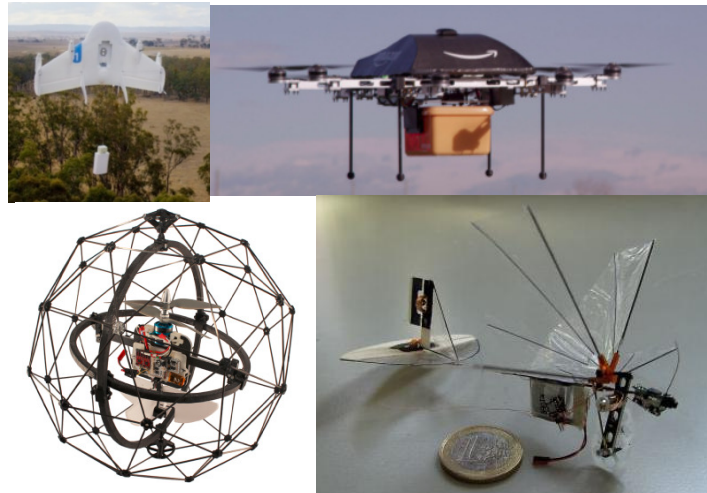


Figure 1.1.1: Various Autonomous Air Vehicle (UAV) platforms.

The challenges for the successful adoption of navigation system in commercial autopilots of UAVs are to improve accuracy, efficiency and robustness. Thus, for visual navigation, it is crucial to guarantee robustness in case of fast motions, illumination changes, motion blur, low textured environments, or in the presence of dynamic obstacles as illustrated in figure 1.1.2. .



Figure 1.1.2: The dominant challenges for the successful application of UAVs navigation is robustness during agile and fast maneuvers, during illumination changes, and in scenes of difficult texture such as high-frequency or repetitive texture.

Such issues increase the difficulty to track visual cues, which is fundamental to enable vision based motion estimation. A UAV system is most robust if selected features can be reliably tracked from frame to frame and the motion is such that the 3D position of those features may be triangulated accurately. Furthermore, in terms of accuracy, the optimal camera

motion estimate is obtained through joint optimization of structure (i.e., landmarks) and motion (i.e., camera poses) in real-time, which is still an open research problem. For this reason, the standard approach, is to estimate the latest camera pose with respect to a previously accumulated scene map and subsequently, given a set of estimated camera poses, update the map.

The third requirement for the successful use of visual navigation on-board UAVs is efficiency. Differently from many state-of-the-art systems in computer vision, the on-board robotic application, imposes hard constraints on the update rate of the running navigation algorithms. Ideally, the use of small UAVs is desired for safety reasons. However, the small size comes with low inertia and thus higher agility. The fast dynamics of the flying robot require a high update rate of the real time localization and mapping navigation approaches. Additionally, small UAVs are typically equipped with computationally constrained processors and a limited power budget. Since every camera image provides a good number of measurements, it poses a great challenge to infer motion from this wealth of data in real-time on computationally constrained platforms. Not surprisingly, the primary performance limitation on first robots was the image processing module. Although meanwhile processors have significantly improved, more efficient navigation algorithms are still very desirable to minimize both the power consumption and the required computational budget on-board UAVs.

Therefore, in this dissertation, these questions are approached by splitting the problem into three different topics: (a) control, (b) perception or state estimation, and (c) navigation. As it is often the case that the accuracy of GPS is not sufficient, or GPS cannot be considered as a reliable or available service, we exclude GPS data from consideration in our framework. The on-board sensors considered are an Inertial Measurement Unit (IMU) and dual monocular cameras. This is due to their complementary nature and their suitability for UAV navigation in terms of power consumption and weight. By combining control, state estimation employed on-board a UAV, we present a visual navigation loop closure, on computationally and payload constrained UAVs, to facilitate long-term fly missions in large and initially unknown environments.

1.2 Related Work

The research in autonomous UAVs is relatively young but advancing very fast. Even though, the research towards autonomous UAVs has been increasingly active over the past few years, resulting in significant progress in the field. We present an overview about the achievements directly related to UAV systems, while pointing to the detailed related work and state of the art in the respective chapters.

The research being done on UAVs can be classified into several fields, mostly in terms of the various sensor(s), used to obtain information about the state of the vehicle, but also into fields as motion control and design, and navigation. In the following chapter, we present the benefits of using external tracking or motion capturing systems, followed by the attainments, which are being made with active on-board sensors with a perceiving depth. A discussion focus on the work related on localization and mapping with (passive) cameras follows, after which we discuss approaches for control and design of UAVs. Finally, related work on navigation approaches for UAVs is presented.

A few approaches exist based on using standard cameras and either known markers or shapes to track UAVs [9], [10], while most proposed approaches rely on professional motion capture systems deployments. Other works showing flights with aggressive maneuvers like flying through narrow horizontal or vertical openings or perching almost upside down on Velcro-tape [11]. Furthermore, performing precision maneuvers, such as flips and choreographed dancing with multiple quad-rotors, balancing an inverted pendulum while flying or playing ping-pong, was shown in [12], [13].

While these approaches focus on control and/or trajectory planning aspects, all these approaches heavily rely on motion capturing systems (e.g. Vicon) providing almost perfect state estimation, and do not focus on how to exploit information from on-board the UAV inertial sensors. We would also like to point out the main difference between on-board cameras and external (i.e., off-board) camera. When using an external Vicon systems, it is not the UAV that is autonomous, but rather the system comprising the external cameras plus the UAV. Therefore, It is only in the case of on-board cameras that the UAV can be considered truly autonomous. Furthermore, a link between the UAV and off-board sensor or/and processor cannot be taken as granted. The benefits of using the camera as an on-board sensor for navigating UAVs in large and initially unknown environments are at hand.

Furthermore, UAV motion control is very important, since it provides the actual control of the robot to allow the robot to

accomplish different tasks. For a ground robot that maybe navigating around obstacles to get to a goal location, or for a manipulator for picking up an object that needs to be sorted and put away. Motion control is a very large field that covers a large range of different subtopics. In this thesis, the motion control is largely based off a recent work, [14] in creating different controller-based motion primitives, and as a result of this being a relatively new approach, the work already done in fields similar to air-ground systems is relatively small.

In addition, a commonly used in the UAV community approach, to stabilize the UAV position is based on Proportional and Differential, (PD) control. Usually this is done by having several nested control loops for position, velocity, attitude and angular velocity, as summarized by [9]. Lee et al., [15], presented a nonlinear controller, performing control in the special Euclidean group $SE(3)$ and thus avoiding singular cases, as would be the case for standard nested control approaches. The latter controller applicability was shown, by authors in [11], [16], on UAVs performing fast and aggressive maneuvers.

Furthermore, a novel control approach was presented by Wang et al., [17], based on nonlinear dynamic inversion. Instead of multiple nested loops, an outer position loop, directly controls angular rates of the UAV, significantly simplifying the math and allowing for higher bandwidth. In [18], the authors, employed a Model Predictive Control (MPC) approach for the outer position loop, and fusing UAV angular velocities, as control signals as well. In the same spirit is the work in Mina Kamel et al. [19], proposed a fast nonlinear MPC approach based on a geometric formulation of the error to track the MAV attitude on the $SO(3)$. However, the MPC approach causes significant load on computational constrain UAV micro-controllers.

State estimation is an another huge area in the field of mobile robotics. The simplest method for estimating state comes from Kalman filtering, employed with different kind of sensors, e.g IMU, a visual sensors (camera), and an optimal flow sensor. These setups is often found on commercial UAVs and are the best methods to use with the sensors listed above. The approaches, for state estimation get significantly better with the addition of additional cameras and LIDAR sensors to the robot. 2D laser range-finders have been largely explored for ground mobile robots, [20] and similar strategies have been extended to UAVs [21]. Although laser scanners are very reliable and robust, today's sensors are too heavy and consume too much power for lightweight UAVs. Therefore, vision sensors are very appealing, their effective range is not limited as in the case of time-of-flight or camera-projector systems allowing to both very far and very close operation to the surface.

Fusing IMU data with a vision sensor is a well studied topic in the literature. As analyzed in, [22], there are two different ways to combine inertial and visual measurements for absolute scale structure and motion estimation, which are called as: (a) loosely-coupled and (b) tightly-coupled. The loosely-coupled approach treats the inertial and vision units as two separate modules running at different rates and exchanging information, [23], [24], while in contrast the tightly-coupled approach combines them into a single, optimal filter, [25], [26]. The proposed work, in [26], present a Kalman Filter (KF) based calibration solution to estimate rotation and translation between a camera and an inertial sensor mounted on a rigid body. They retrieve the scale from a feature pattern with known dimensions. The authors in [27], extended the calibration method to the observation of an a priori unknown static structure. In fact, since their work estimates the gravity vector in the vision frame ,e.g. the frame with respect to which the camera pose is estimated, it is a complete vision-inertial, self-calibrating EKF, which is implemented in navigation system.

In addition, early works on vision-based UAV navigation, have used biologically-inspired control algorithms, such as optical flow [28]. However, since optical flow only measures the relative velocity, drift in the UAV position is still inevitable. Therefore, recent camera-based autopilots used advanced computer vision algorithms relying on visual odometry (VO) technology to perform autonomously basic maneuvers, such as take off, landing, and way-point-based navigation [29], [1].

1.3 Contributions of the thesis

In the preceding section, we discussed several challenges for autonomous UAVs. In order to address the latter in this thesis, we focus on solutions incorporating a sensor setup, comprising of dual on-board monocular cameras as main localization sensors, and an Inertial Measurement Unit (IMU). However, while this setup is very suitable for MAV navigation, it imposes great research challenges. This dissertation investigates and proposes methods in the key areas, enabling autonomous operation of UAVs and thoroughly combines them, while specifically focusing on feasibility and integration issues on UAVs, to be successfully employed on a real working system. The following list summarizes the key contribu-

tions of the work presented in this thesis:

1. *Development of a modular visual - inertial sensory fusion framework, and vision algorithms for real-time, on-board pose estimation on computationally constrained airborne vehicles. This framework is capable of constantly estimating the UAV's pose and motion in real-time. it is also able to incorporate dual monocular vision sensors along with an IMU.*
2. *Introduction of a novel Horizon Line detector solution, that employs Particle Swarm Optimization , to assess candidate horizon lines. The extracted Horizon line is used to derive UAV roll and pitch angles. The latter are included as sensor measurements in the proposed sensor fusion framework.*
3. *Formulation of a visual navigation technique, that enables the Asctec Firefly UAV to localize and navigate in outdoors environment. This is achieved through the use of a general, scalable, tracking and mapping system (SLAM).*

The overall goal of this dissertation has been to combine and integrate the above, and enable a UAV to navigate in large, unknown outdoor environments. The latter implies that the UAV is stable flying on its own, while an operator at a ground station only gives high-level commands, such as simple way-points commands, desired velocities and dynamic trajectories.

1.4 Structure of this Dissertation

The rest of this thesis is organized as follows. This introduction is followed by Chapter 2, where we present the basic method of rendering the camera as a real-time on-board visual pose estimator. Furthermore we discuss camera model, feature detection and structure from motion approaches, theoretical background. In Chapter 3, we present the control system of the UAV. A short introduction of the necessary dynamics is followed by the position and attitude control approaches. The presented controllers rely on knowledge about the UAV's state. Therefore, accurate state estimation is key to autonomous operation, which is covered in Chapter 3. We introduce a multi-sensor fusion system, from an IMU with additional sensors and algorithms, that are able to estimate the pose or position of the UAV within an Extended Kalman Filter (EKF) formulation. We start the analysis with a core sensor-suite consisting of an IMU and a camera only. Then, we gradually introduce an additional (drift compensating) visual sensor and a pressure sensor and end the chapter on EKF task distribution.

Following the description of the control principles governing our MAV and the onboard state estimation, Chapter 4 describes a Horizon line detection framework based on Particle Swarm Optimization. This is accomplished via the PSO's objective function which is formulated as a variant of the Bag of Words (BOW) encoding of the image content. We conclude the chapter by describing how the extracted HL is used to derive UAV roll and pitch angles.

Chapter 5, describes a visual navigation approach that takes both uncertainty of localization and state estimation, as well as the UAV dynamics and control into account. This uncertainty and motion awareness in the navigation process is of major importance. On one hand, it enables dynamic flights, and also facilitates observability given that, several states within our state estimation framework require motion in order to become observable. A proposed navigation framework based on visual simultaneous localization and mapping (RT-PTAM SLAM) approach is described. The full navigation framework consists by two camera sensors, where RT-PTAM and horizon line are utilized, and the multi-sensor fusion module used to control the UAV.

In Chapter 6 both qualitative and quantitative experimental results are presented in order to demonstrate the effectiveness of the proposed visual navigation framework. Initially, we explain how the various modules or building blocks, hardware as well as software, interact with each other. Furthermore Horizon Line detection experimental results, along with the horizon estimated pitch and roll angles are provided. In addition, navigation results are shown in simulation as well as in a larger size outdoor environment. This dissertation concludes in Chapter 7, where we summarize our results and achievements, and finally point out directions.

Chapter 2

Visual sensors for Robot Vision and Perception

The ability and efficiency of the visual cortex to interpret the large amount of complex information perceived by the human vision sensor, e.g eyes, is astonishing. Promptly we can describe our motion in the three dimensional space, characterize the size and structure of the surrounding environment that we are in. Replicating human understanding of space and motion in artificial systems represents an enormous challenge for scientists and engineers. However, even the smallest steps in this direction have the potential to unlock a multitude of exciting applications, such as autonomous robots. Very rapid progress has been made in the last decade and continues to be made, aided by steady improvements in computing and sensing hardware. Today, computers are better than humans in detecting visual signs and complex patterns in images and Google's autonomous car has autonomously driven more than a million miles, [30].

Today, cameras are capable to record many times a second an image that consists of millions of individual pixel measurements. The main camera approach in robotic applications occurs as a real-time pose sensor. This approach is based on ensuring the robot can achieve a desired pose and it is far from being straightforward. In such a case a camera can be seen as real-time pose sensor, in a map based approach (SLAM), or a real-time motion sensor in map-less approach. The latter is actually an inertial-optical flow, a 3 degree of freedom (DoF) sensor that does not need any sort of map or feature history. Employing optical flow in visual-inertial tracking is increasingly popular, and it can be even integrated in a real-time system (inertial-optical flow), based on body-velocity recovery with a conventional camera, while all processing is done on-board with a low-end computer. In fact, two features matches in two consecutive camera frames (i.e. optical flow) are sufficient for an optical flow method to recover the body velocity [31]. Pose drifts and map losses are thus not an issue, making it an ideal and complementary method to the aforementioned map-based approach. In this thesis, we focus only on the map based pose sensor, presented in chapter 5, a 6 DoF pose sensor and SLAM framework. In order to proceed with the SLAM framework in chapter 5, on computationally and payload constrained UAV for long-term navigation in large and initially unknown environments, this chapter establishes the camera sensor as a valid on-board and real-time motion sensor.

The current chapter is mainly motivated as an introduction to various concepts, required later, such as camera model, efficient bundle adjustment and feature matching. First Section 2.1 explores how we represent images, in particular, we look at perspective projection, and the description of how the camera is modeled as a pinhole camera is provided. Section 2.2, introduces the topic of camera calibration, specifically the estimation of the parameters of the perspective transformation. Section 2.3 covers Structure from Motion (SFM) theory, exploiting either the motion of the object being observed or the motion of the camera itself, and algorithms for formulating the bundle adjustment (BA) problem. In section 2.4 we focus on extracting interest Fast - Corner features from images. We conclude the chapter in section 2.5, by providing on how feature extracting points may be used to solve problems such as image feature correspondences, stitching and camera tracking.

2.1 Relating 3D Position to 2D visual sensor

A basic pinhole model is often used to model how a point in 3D space is projected onto a 2D camera image. This model, however, often does not accurately reflect the actual behavior of cameras due to the physical optical distortions of the

camera lens. In this section, an overview of the pinhole camera model is first presented, followed by an overview of some corrections for camera calibration and lens distortion in section 2.2. A perspective projection model of a pinhole camera allows position in a 2D camera image to be inferred from 3D position as shown in figure 2.1.1. A pinhole camera has no focus implementation, meaning that all objects are in focus irrespective of the object distance, and produces very obscure images since its radiant power is only the scene luminance (Wm^{-2}), if no appropriate objective lens is used. The z-coordinate of the object and its image, with respect to the lens center, are related by the thin lens equation, where z_{obj} , z_{img} are the distances to the object and the image, and f is lens focal length:

$$\frac{1}{z_{obj}} + \frac{1}{z_{img}} = \frac{1}{f} \quad (2.1.1)$$

The pinhole camera model projects an arbitrary point $P = (X, Y, Z)$ to a pixel point $p(u, v)$ on the image plane, performing a mapping from 3-dimensional space to the 2-dimensional image plane $\mathcal{P} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$. We can write the image-plane point coordinates in homogeneous form (The subscript_H indicates homogeneous quantities) $p_H = (x_H, y_H, z_H)$.

$$\begin{aligned} x_H &= \frac{f}{X} \quad \text{and} \quad x = \frac{x_H}{z_H} \\ y_H &= \frac{f}{Y} \quad \text{and} \quad y = \frac{y_H}{z_H} \\ z_H &= Z \end{aligned} \quad (2.1.2)$$

Equations 2.1.2 represent a transformation from the world to the image plane with notable characteristics. Straight lines and conics in the world are projected to straight lines and conics on the image plane. Parallel lines in the world are projected to lines that intersect at a vanishing point (or focal point). The exception are parallel lines lying in a plane parallel to the image plane, which always remain parallel. The mapping does not preserve shape (depends on distance), internal angles, translation, rotation and scaling.

In addition an important topic is that the mapping is not one-to-one and no unique inverse exists. That is, given $p(u, v)$ is not possible to uniquely determine $P(X, Y, Z)$. All that can be said is that the world P point lies somewhere along the red projecting ray as shown in figure 2.1.1. The pixels are uniform in size and centered on a regular grid so the pixel coordinates are related to the image-plane coordinates, where ρ_w and ρ_h are the width and height of each pixel respectively and x_0, y_0 is the principal point, e.g. the coordinate of the point where the optical axis intersects the image plane with respect to the new origin.

$$\begin{aligned} u &= \frac{x}{\rho_w} + x_0 \\ v &= \frac{y}{\rho_h} + y_0 \end{aligned} \quad (2.1.3)$$

We can write the camera projection in general form, where all the terms are rolled up into the matrix C , known as the camera matrix, which performs scaling, translation and perspective projection and It is often also referred to as the projection matrix or the camera calibration matrix.

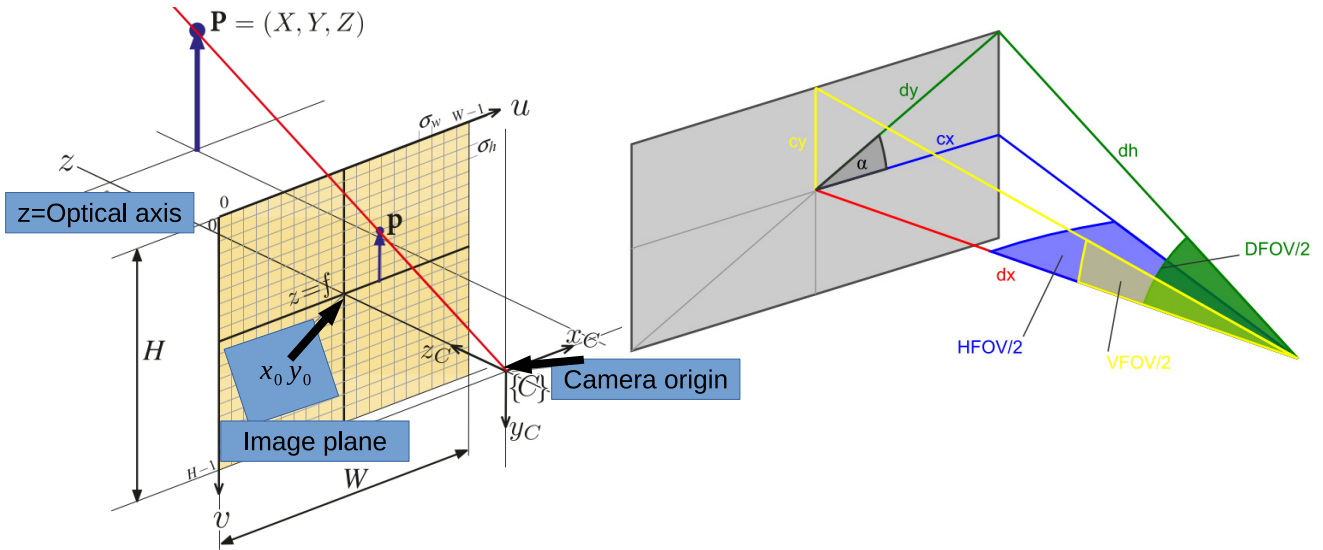


Figure 2.1.1: Camera perspective projection model, used for relating 3D position to position in 2D images. The rays converge on the origin of the camera frame C and a non inverted image is projected onto the image plane located at $z = f$. The z -axis intersects the image plane at the principal point which is the origin of the 2D image coordinate frame. The point $P = (X, Y, Z)$ is projected onto the camera image plane to the point $p(u, v)$. Left: Central projection model showing image plane and discrete pixels, the pixel coordinates are a 2-vector u, v with principal point as x_0, y_0 . Image plain is a $W \times H$ grid. The positive z -axis is the camera's optical axis. Right: Pinhole camera field of View (FOV) is the World area, that can be seen from the camera lens. It is represented as an angle. Typically the reported FOV from camera vendors is the diagonal field of view ($DFOV$). The FOV can be determined from the geometry point of view as horizontal ($HFOV$) and vertical ($VFOV$) with α the relative angle, which along with the $DFOV$ are coupling all camera field of Views based on equation $DFOV = \sqrt{HFOV^2 + VFOV^2}$.

$$\begin{aligned}
 p_H &= \begin{bmatrix} \frac{f}{\rho_w} & 0 & x_0 \\ 0 & \frac{f}{\rho_h} & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} (T_c^0)^{-1} P_H \\
 &= K P_0 (T_c^0)^{-1} P_H = C P_H \\
 K &= \begin{bmatrix} \frac{f}{\rho_w} & 0 & x_0 \\ 0 & \frac{f}{\rho_h} & y_0 \\ 0 & 0 & 1 \end{bmatrix} \\
 t_c^0 &= \begin{bmatrix} R_{3 \times 3} & t_{3 \times 1} \\ 0_{3 \times 1} & 1 \end{bmatrix}
 \end{aligned} \tag{2.1.4}$$

This is a 3×4 matrix, multiply by a projection matrix (not in homogeneous form), P_0 is the point coordinate vector in the world frame. The variable K is the camera parameter matrix and comprises the intrinsic parameters of the camera and sensor such as $f, x_0, y_0, \rho_h, \rho_w$. The matrix t_c^0 is the 6 DoF pose of the camera and comprises of 4×4 matrix parameters (the extrinsic parameters, homogeneous form) that describe camera translation t and orientation R in $SE(3)$. There are 5

intrinsic and 6 extrinsic parameters, a total of 11 independent parameters to describe a camera. The camera matrix has 12 elements so one degree of freedom, the scale factor, is unconstrained and can be arbitrarily chosen. In practice the camera parameters are not known and must be estimated using a camera calibration procedure.

Focal length set in equation 2.1.2 is set as equal for x and y direction. However, in reality, the focal lengths in the horizontal and vertical directions may be different. So the first part of equations in 2.1.1 expressions will be rewritten as:

$$x_H = \frac{f_x}{X} \tag{2.1.5}$$

$$y_H = \frac{f_y}{Y}$$

These focal lengths f_x, f_y can be computed from knowledge of the width and height of the camera image plane W and H and from the horizontal $HFOV$ and vertical fields of view $VFOV$.

$$f_x = \frac{W/(2\rho_w)}{\arctan(HFOV/2)} \tag{2.1.6}$$

$$f_y = \frac{H/(2\rho_h)}{\arctan(VFOV/2)}$$

We note that the field of view is also a function of the dimensions of the camera chip $W \cdot \rho_w \times H \cdot \rho_h$. Knowing the horizontal W and vertical H camera dimensions and f_x, f_y , we can compute the ρ_w, ρ_h from equation 2.1.6.

$$\rho_w = \frac{W}{2f_x \arctan(HFOV/2)} \tag{2.1.7}$$

$$\rho_h = \frac{H}{2f_y \arctan(VFOV/2)}$$

However, these camera parameters $W, H, f_x, f_y, HFOV, VFOV, \rho_w$ and ρ_h in many occasions not known. Many camera vendors only provide camera FOV , which refers to the diagonal field of view ($DFOV$) and a general focal length f . To overcome this, camera calibration approaches are presented in the following section.

2.2 Camera Calibration

In this section, before we proceed with the camera calibration approaches, a summary relative to lens distortion is shown. The perspective projection model for an ideal camera often does not accurately reflect the behavior of physical cameras due to the effects of lens distortions and imperfections. Lens imperfections result in a variety of distortions including chromatic aberration (color fringing), spherical aberration or astigmatism (variation in focus across the scene). In addition, the most problematic effect that we encounter for robotic applications, is the geometric distortion presented in figure 2.2.1, where a point on the image plane are displaced from its computed position according to equation 2.1.4 and is consisted with the radial and tangential geometric distortion.

The point $p(u, v)$ in figure 2.1.1 after distortion is given by :

$$\begin{aligned} u_{distortion} &= u + \delta_u \\ v_{distortion} &= v + \delta_v \end{aligned} \tag{2.2.1}$$

where the δ_u, δ_v are computed as:

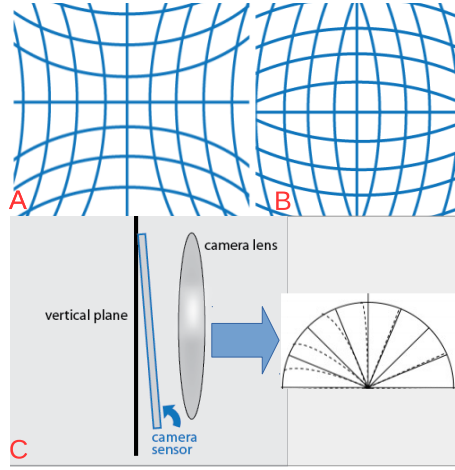


Figure 2.2.1: Pinhole camera Lens geometric distortion. Upper: Radial distortion causes image points to be translated along radial lines from the principal point. A: Negative radial distortion (pincushion). Pincushion distortion occurs when magnification increases with distance from the principal point and causes straight lines near the edge of the image to curve inward. B: Positive radial distortion (barrel). Barrel distortion occurs when magnification decreases with distance from the principal point which causes straight lines near the edge of the image to curve outward. Bottom: Tangential distortion, or decentralizing distortion, occurs at right angles and radius displacements but is generally less significant than radial distortion. C: Tangential distortion is caused by a displacement between camera plane and vertical plane with radius displacement as a result.

$$\delta_u = \delta_u^{radial} + \delta_u^{tangential} = u(1 + a_1t^2 + a_2t^4 + a_3t^6 + \dots) + 2b_1uv + b_2(t^2 + 2u^2) \quad (2.2.2)$$

$$\delta_v = \delta_v^{radial} + \delta_v^{tangential} = v(1 + a_1t^2 + a_2t^4 + a_3t^6 + \dots) + b_1(t^2 + 2v^2) + 2b_2uv$$

In order to correct the distortion displacement, a negative δ_u , δ_v from equation 2.2.2 is applied to different image points, and the distortion model parameters $a_1, a_2, a_3, \dots, b_1, b_2$ are added as camera parameters.

The camera projection model 2.1.4, as already mentioned, has a number of parameters that in practice are unknown, and even the ones that are provided by camera vendors are not always accurate for every manufactured camera. The process of determining the camera's intrinsic parameters and the extrinsic parameters with respect to the world coordinate system, is camera calibration. Calibration techniques rely on sets of world points, whose relative coordinates are known and whose corresponding image-plane coordinates are also known. The classic linear calibration model, based on the homogeneous transformation, [32], and similar the linear calibration model obtained by using features such as lines within the image scene, [33], require a single view of a 3-dimensional calibration target. However linear techniques such as this cannot estimate lens distortion parameters. The distortion will introduce errors into the camera matrix elements, even so in the most of the cases, the distortion is low, this might be not acceptable.

Thus, distortion parameters are better estimated using a non-linear optimization over all parameters, typically 16 or more parameters, with the linear solution used as the initial parameters estimate. Distortion modeling is implemented in state of the art calibration techniques such as Bouquet's Camera Calibration Toolbox for MATLAB [34], OpenCV [35], ROS (Robot Operating System) camera calibrator [36] and the Parallel Tracking and Mapping (PTAM) Slam calibrator suite [37].

These calibration suits are able to estimate distortion with different distortion models. Devernay and Faugeras [38], proposed the ATAN FOV distortion model, the model is used by PTAM Slam and can be also calibrated using PTAM calibrator. The tangential distortion in ATAN model, can often be neglected yielding a faster projection and un-projection model. It was found that for many robotic vision applications, tangential distortion need not to be considered [39]. The

OpenCV and ROS calibration approach is build in with the typical pinhole model, presented in section 2.1. This distortion model uses three radial and two tangential distortion parameters, shown in equations 2.2.2. Another distortion model approach was developed by Scaramuzza et al, [40], the Ocam model for central fish-eye or omnidirectional cameras, which can be used to model cameras with high field of view or even omnidirectional cameras. We do not provide more information about fish-eye distortion models, as in this work only pinhole perspective cameras are utilized. The non linear distrotion model and calibration suite used here refer to the ATAN-PTAM models. The decision was made mostly with reference to the slam algorithm that we implemented, the PTAM.

The PTAM camera parameters, that are reported by PTAM calibration, are the x -direction, y -direction normalized focal lengths, f_x , f_y , normalized x -center c_x and y -center c_y and the normalized distortion factor s , with $s < 1$ for wide FoV (> 120 deg). The focal lengths and image centers are transformed to image coordinates (pixels) and defined as:

$$\begin{aligned} f_u &= \rho_w f_x \\ f_v &= \rho_h f_y \\ c_u &= \rho_w c_x \\ c_v &= \rho_h c_y \end{aligned} \quad (2.2.3)$$

The radial distortion, in ATAN approach, is the FOV model of [38]. This model has only one parameter, which is the field of view w_{FOV} of the lens. The radial distortion transformation factor is defined as the ratio of distorted radius $r_d = \sqrt{u_{distortion}^2 + v_{distortion}^2}$ and the undistorted radius is $r_u = \sqrt{u^2 + v^2}$ and $r_{factor} = r_d/r_u$, where the distorted and undistorted radius are given by:

$$r_d = f_{distortion \rightarrow un-distrtortion}(r_u) = \frac{1}{w_{FOV}} \arctan\left(2r_u \tan\left(\frac{w_{FOV}}{2}\right)\right) \quad (2.2.4)$$

$$r_u = f_{un-distortion \rightarrow distortion}(r_d) = \frac{\tan(r_d w_{FOV})}{2 \tan\left(\frac{w_{FOV}}{2}\right)}$$

Tangential distortion is not considered here for the PTAM-ATAN model used in this work. If this one-parameter FOV model, e.g. ATAN, is not sufficient to model the complex distortion of lens, the distortion model in equation 2.2.2 can be applied before equations 2.2.4 with $a_1 = 0$ (w_{FOV} , as a first order distortion parameter, would be redundant with a_1). A second order model ATAN will have $a_2 \neq 0$, and a third order ATAN model will have $a_3 \neq 0$. The Ptam-calibrator using the atan model in equation 2.2.3 and perspective projection camera model in equation 2.1.4, is able to compute the 3D projection point center (X, Y, Z) from camera frame to 2D image coordinates (u, v) .

$$r_u = \sqrt{\frac{X^2 + Y^2}{Z^2}} \quad (2.2.5)$$

$$r_d = f_{distortion \rightarrow un-distrtortion}(r_u) = \frac{1}{w_{FOV}} \arctan\left(2\sqrt{\frac{X^2 + Y^2}{Z^2}} \tan\left(\frac{w_{FOV}}{2}\right)\right)$$

When setting $Z = 1$ the last equations simplify to:

$$r_u = \sqrt{X^2 + Y^2} \quad (2.2.6)$$

$$r_d = f_{distortion \rightarrow un-distrtortion}(r_u) = \frac{1}{w_{FOV}} \arctan\left(2\sqrt{X^2 + Y^2} \tan\left(\frac{w_{FOV}}{2}\right)\right)$$

Then we un-project from the 2D image coordinates (u, v) to camera z 1-plane and receiving the 2D projection point (X, Y) by using the inverse transformation.

$$r_d = \sqrt{X^2 + Y^2} = \sqrt{\frac{u - c_u}{f_u}^2 + \frac{v - c_v}{f_v}^2} \quad (2.2.7)$$

$$r_u = f_{\text{distortion} \rightarrow \text{un-distortion}}(r_d) = \frac{\tan\left(\left(\sqrt{\frac{u - c_u}{f_u}^2 + \frac{v - c_v}{f_v}^2}\right) w_{FOV}\right)}{2 \tan\left(\frac{w_{FOV}}{2}\right)}$$

Below the un-distortion 2D projection point (X, Y, Z) is presented:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \frac{r_u}{r_d} \begin{bmatrix} \frac{1}{f_u} & 0 \\ 0 & \frac{1}{f_v} \end{bmatrix} \begin{bmatrix} u - c_u \\ v - c_v \end{bmatrix} \quad (2.2.8)$$

2.3 Visual Pose Sensor: Structure from Motion

Important application areas in robotics and beyond open up, if similar performance can be demonstrated using monocular vision, since a single camera will always be cheaper, more compact and easier to calibrate than a multi-camera rig. Recovering 3D information from a 2D camera sensor is a well known problem in robotic vision, the key is exploiting either the motion of the object being observed or the motion of the camera itself to recover the missing, from single sensor, depth information. Structure from Motion, (SFM) has been extensively studied in the case of offline 3D reconstruction, however in recent years the techniques developed have been applied to real-time problems in robotics. This concept of SFM is very related to visual odometry: a purely incremental motion estimate. The concept of odometry comes from the field of mobile robotics where wheel encoders measure the rotations of the individual wheels of a vehicle.

In this section we will introduce some key issues in SFM approach and describe common methods also implemented in this work visual pose framework, presented in chapter 5. In particular we look at the problems of estimating the pose of a camera when observing a known shape, recovering the available stereo information from two distinct views of the same point and estimating the relative pose between two cameras. We will conclude with the bundle adjustment, a powerful technique to iteratively refine both structure (3D points) and motion (camera poses) estimate.

2.3.1 Perspective Multi-Point Approach

The Perspective n-Point problem addresses the issue of estimating camera poses and camera parameters from observations of known objects and points. Section 2.1, already established the relating problem of converting a 3D Position to 2D visual sensor. This can be easily generalized with a set of n 3D points together with their corresponding 2D positions in the camera frame, shown in figure 2.3.1. Given the 2D to 3D point mapping $(\tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_i) \leftrightarrow (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_i)$, with $i = 1, \dots, n$, we may compute, using equation 2.1.4, the $\tilde{X}_i = C\tilde{x}_i$. While inherently non-linear, this problem can be linearized, if we make the assumption that the unknown variables are independent.

$$x = \frac{c_{11}X + c_{12}Y + c_{13}Z + c_{14}W}{c_{31}X + c_{32}Y + c_{33}Z + c_{34}W} \quad (2.3.1)$$

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

Where x, y are the 2D image coordinates of feature point x_i , X, Y, Z, W are the 3D homogeneous coordinates of 3D point X_i and C is a 3×4 camera matrix. These can be arranged into a homogeneous linear equation of the form $Ax = 0$, and multiplying each equation by the denominator, result in the following system:

$$\begin{bmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -x \\ 0 & 0 & 0 & 0 & X & Y & 0 & Z & 1 & -yX & -yY & -yZ & -y \end{bmatrix} x = 0 \quad (2.3.2)$$

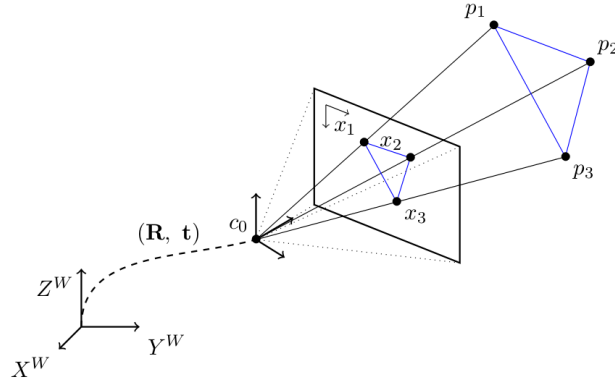


Figure 2.3.1: Perspective n-Point approach, mapping p_1, p_2, p_3 3D points to x_1, x_2, x_3 2D points using C_0 camera matrix relative transformation.

Given that camera matrix has 12 unknowns, at least 6 correspondences are required to produce a unique solution. This can be solved using Singular Value Decomposition (SVD) where x is the Eigenvector corresponding to the smallest Eigenvalue of the matrix. Once C has been obtained, we can recover both the intrinsic and extrinsic camera parameters by factoring C into $KP_0(R_c^0)^{-1}P_H$ using QR decomposition in equation 2.1.4. For increased accuracy the parameters of C can be further refined using least squares approach, finding the parameters of C that minimize the re-projection error for all points based on the initial solution. If points are affected by noise or bad correspondences, a Random Sampling and Consensus (RANSAC) based approach can be used. Camera parameter can be recovered from calibration techniques presented in section 2.2. The converse of the PnP is the Triangulation problem. Triangulation can be solved using the same system of linear equations 2.3.1 as before, however in this case the unknowns are the X, Y, Z coordinates of the 3D point. As before we can solve this system of equations using SVD and again increased accuracy is obtained by taking a least squares solution. A complete description of triangulation approach, is beyond the scope of this work.

2.3.2 Visual front-end

Difficulties had to be overcome before a probabilistic sequential approach could successfully be implemented for a visual pose sensor framework, e.g. SLAM, to be able to work, due to the fact that image features are being observed from multiple viewpoints before fully constrained 3D points (landmarks) estimates can be made. In addition, difficulties in feature initialization schemes, which are permitted the use of a sequential probabilistic estimation of the joint camera and map, should be eventually treated. These issues, and the fact that visual maps from single sensor are generally less well constrained than those built by metric sensor (Lidar-Laser), as outcome, it is more difficult using visual pose approach in comparison with other sensors to build algorithms which could tackle large areas by composing local fragments. In this section we concentrate on visual front-end (image processing), followed in the next section by the optimization of the back-end (joint structure and motion estimation).

The first problem in SFM approaches, assuming we would like to estimate the path of the camera through the space of rigid transformations $\mathbf{SE}(3)$, is to associate points or regions in one image with points or regions in subsequent images. While performing this feature tracking, it is not only important to account for appearance of similarities among the images, but also guarantee that these feature tracks mutually agree so they can be explained by the camera motion. A framework which fulfills this task is called a visual front-end. Front-end is the implementation of scene geometry in visual sensor images, using primitives in the images. By far the most popular primitives are point features (e.g. corners, or blobs), even though other primitives such as line-based features and more complex computed features, such as the SIFT [41], SURF [42] and

Orb [43] features, are occasionally used too. In this problem, given a set of camera frames, we would like to associate two dimensional point measurements among them. Ideally, each such set of two dimensional points is associated with a single three dimensional set in the world. In this case, each point corresponds to the re-projection of the three dimensional point in image frame. For this problem of feature matching, there are two canonical approaches: Bottom-up and top-down matching.

Bottom-up approach, tackles feature tracking in terms of matching correspondences between a pair of images. Thus, one associates feature points in one image with feature points in the other image using appearance information only. Good features have desirable properties such as distinctiveness and repeatability. The ability to match unique features between two images allows a number of interesting applications such as image stitching, object tracking or, for the work in this thesis, motion estimation. Features can describe geometric shapes such as lines and corners or less rigidly defined regions such as colored/textured blobs. To be robust to view-point changes, rotational and scale invariant features such as SIFT or SURF are used. The visual sensor framework deployed in this thesis is able to detect Corners features, the latter detector being explained in more detail in section 2.4.

Given a set of candidate matches, we must look for a geometric constraint between the two images which agrees with the candidate matches. In order to select a subset of inliers from all candidates which agrees with specific threshold, specified usually from the problem epipolar constraint, a robust estimation process is required. By far the most popular approaches are based on RanSaC and its variants. Once an inlier set is found, the initial constraint can be improved upon using least squares optimization.

On the other hand, top-down tracking approach is a model based method and therefore a recursive method. The main idea is to guide the feature tracking using our geometric model which was estimated from the previous frames, e.g. described by a Gaussian distribution and using an EKF-based SLAM. Assuming it is available, some kind of motion prior which allows us to predict the distribution over the camera pose. Then, we can calculate the distribution in the image space, the mean and the innovation covariance, where the point is expected to be seen in the image.

To summarize, bottom-up matching has the advantage that it does not require any motion prior and can therefore detect matches between frames with arbitrary camera configuration. However, it is computationally expensive since it does not only require to visit every single pixel in the image at least once, but also depends on a high number of RanSaC iterations to run robustly. On the other hand, top-down matching is model-based, and more efficient since feature matching can be restricted to small elliptical search regions. However, it usually requires an un-certainty estimate for the point maps as well as a narrow motion prior. Before we proceed to the visual back-end approach, based on bundle adjustment, an introduction to Least square is in order.

2.3.3 Least Squares

In the most case, least squares is a parameter estimation approach to find an approximate solution to an overdetermined system of equations. This is done by minimising the sum of squared errors for each equation. In many cases there is no exact solution to an overdetermined system of equations, in such a case least squares can be used to find the closest solution to the exact one. Least squares is particularly useful in the context of fitting a model to a set of noisy observations. For example given dependent measurements $m = (m_1, m_2, m_3, \dots, m_n)^T$ and independent data points $d = (d_1, d_2, d_3, \dots, d_n)^T$, the aim is to find a set of parameters $T = (T_1, T_2, T_3, \dots, T_n)^T$, which represent the best fit between the model and the measured data points. Best fit is defined as the set of parameters at which the sum F_Σ of the residuals is minimal, and where the residuals r_i is the difference between the measured value and the value predicted by the model :

$$F_\Sigma = \sum_{i=1}^n ||r_i||^2 \tag{2.3.3}$$

$$r_i = y_i - f(d_i, m_i)$$

The form of the function f determines the nature of the least squares problem, that is linear or, in many occasions, non-linear least squares. In SFM approaches, the re-projection error is commonly used as the residual in least squares problems. The re-projection error is the difference between the pixel location of an observed 3D point and its estimated position based on the camera pose and the points 3D position, figure 2.3.2.

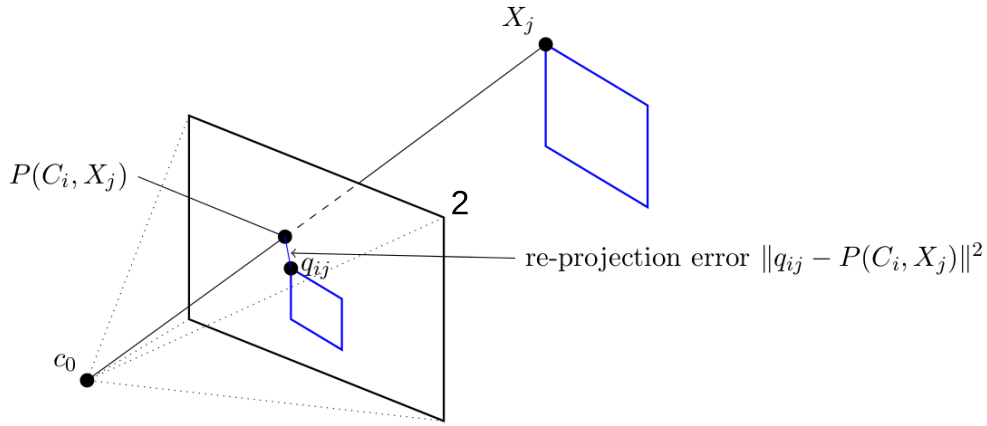


Figure 2.3.2: The re-projection error between the measure point location q_{ij} of an observed 3D point X_j , and the value predicted by the model $P(C_i, X_j)$

In this thesis, the residual function f is the projection function based on the pinhole camera mode with lens distortion, which we used as visual sensor. This results to a non-linear function, and a appropriate non-linear least squares approach is required. As we have already mentioned, non-linear least squares can be applied to the Multi-Point Approach and Triangulation problems to produce more accurate results, and to a more general problem the Bundle Adjustment which we consider in the next section.

2.3.4 Visual back-end

The visual back-end task is to jointly estimate camera motion and structure given the visual feature associations from the front-end. This task is tackled by the back-end optimization, which estimates the structure and motion parameters by minimizing the distance between the projections of the estimated 3D points onto the image and their visual measurements. Therefore, SFM in a strict sense refers to incremental pose estimation, where we estimate the current pose given by the temporally previous one and visual measurements from the previous image pair. By allowing the integration of measurements from non-consecutive frames, we can continue to track visual features which are temporarily occluded and therefore can deal with mini loop closures. Furthermore, we relax the concept of temporally consecutive frames to key-frames which are sampled in the spatial domain, [44]. The back-end optimization usually is done in visual SLAM frameworks, by a joint estimation over a set of poses p_i and a set of points x_j called bundle adjustment (BA) [45]. Another popular approach is based on Gaussian filters such as the Extended Kalman Filter. The advantages of BA over filtering are presented in chapter 5.

The BA problem can be seen as a combination of the triangulation problem and the multi-projection problem. In the case of Visual SLAM we have existing estimates of both the 3D structure and camera trajectory which are estimated incrementally. The initial estimates are affected by both noise (assumed normally distributed) as well as accumulated error from the incremental motion estimation. In bundle adjustment, presented in figure 2.3.3 we seek to jointly refine the 3D structure and camera trajectory. We can then define the bundle adjustment in terms of a non-linear least squares problem where we seek to minimize the cost function, defined as the re-projection error between observed and predicted image points:

$$d_{i,j}^{min} = \min \sum_{i=1}^n \sum_{j=1}^k ||h(X_i, C_j) - x_{ji}||^2 \quad (2.3.4)$$

where X_n are the n 3D map-points, C_j are k camera matrices, representing k scene images. For each map-point a set of 2D

image measurements x_{ji} is defined, where each x_{ji} measurement is an observation of map-point X_i in the camera image, expressed by the camera matrix C_j .

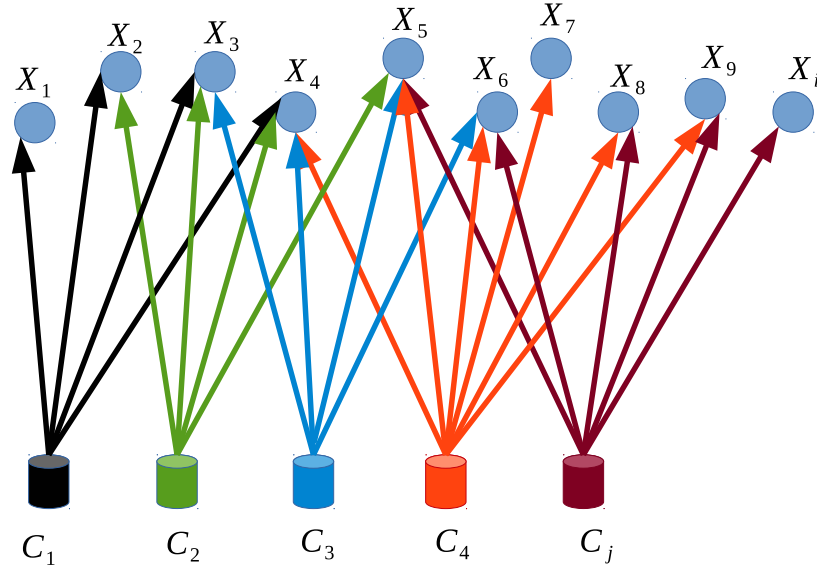


Figure 2.3.3: Bundle Adjustment approach. The name bundle adjustment comes from the representation of the problem, in which the lines between cameras and the observed points represent the bundles to be incrementally adjusted till an improved solution is found.

Algorithms such as Gauss-Newton (GN) and Levenberg - Marquardt (LM), [46], [47] have been successfully applied to such problems in [33], [48] and [49]. The typical approach for the non-linear case is to first linearize the problem using a first-order Taylor expansion around the current solution, compute an adjustment based on this linearization, apply it and repeat until convergence is reached. The GN algorithm works well and fast when the initial solution is close to the optimal, whereas LM is more effective when the quality of the initial solution cannot be guaranteed. LM provides this flexibility by a combination of the gradient descent and Gauss-Newton. The behaviour of the LM algorithm is as follows: far from the local minimum LM uses a gradient descent approach which is slower than Gauss-Newton but guaranteed to converge. Close to the local minimum the LM algorithm switches to Gauss-Newton in order to speed up convergence.

Hence, in the general form of a least squares problem, we have a function f which describes the relation between a parameter vector $x \in \mathbb{R}^k$ and a measurement vector $\hat{y} \in \mathbb{R}^n$, with $\hat{y} = f(x)$. An initial parameter estimate x_0 and a measurement vector y are provided and the aim is to iteratively refine the parameter vector, such that it minimizes the squared distance $\varepsilon^T \varepsilon$, where $\varepsilon = y - \hat{y}$. A assumption is made, that f is locally linear. The LM algorithm generates an incremental update δx , using the initial parameter estimate x_0 to iteratively redefine x . To determine the update δx , the function f is approximated by $f(x + \delta x) \approx f(x) + J\delta x$, where $J = \frac{\partial f}{\partial x}$ is the Jacobian of f . Starting with the initial parameter estimate x_0 , the algorithm produces a sequence of parameter vectors $x_1, x_2, x_3, x_4, \dots$, until it converges towards a local minimum for x . Therefore at each iteration it is necessary to compute a δx that minimizes the following equation:

$$\|y - f(x + \delta x)\| \approx \|x - f(x) - J\delta x\| = \|\varepsilon - J\delta x\| \quad (2.3.5)$$

Using normal equations, the solution to a linear least squares problem $\|\varepsilon - J\delta x\|$ over the $J\delta x$ and δx is the Gauss-Newton step :

$$J^T J \delta x = J^T \varepsilon \quad (2.3.6)$$

The matrix $J^T J$ is the first order approximation of the $\frac{1}{2} \varepsilon^T \varepsilon$ Hessian matrix. The $-J^T J$ corresponds to steepest descent's $\varepsilon^T \varepsilon$ error term. The LM uses an augmented version of the normal equations 2.3.6 where a damping factor $\lambda > 0$ is added.

$$(J^T J + \lambda \mathcal{I}) \delta x = J^T \varepsilon \quad (2.3.7)$$

The λ is adjusted at each iteration based on the outcome of equation 2.3.7, if the outcome results in a reduction in the $\varepsilon^T \varepsilon$ term, then the value of λ is decreased for the next iteration, on the other hand, if the outcome results in increased error then λ adjusts until a solution is obtained that results in a reduction of the error. This means, that it may require several computations of equation 2.3.7, before an adjustment δx that reduces error is found. As the GN method uses an approximation of the Hessian of the residuals, this approximation method is valid only in cases where the error is small (i.e. the current set of parameters are close to the local minimum) or the residual function is close to linear. If either assumption does not hold the method is not guaranteed to converge. Therefore the convergence is achieved using a damping factor λ , when the value of λ is large the adjustment is skewed towards the steepest descent, this results in smaller steps or more iterations but is guaranteed to reach a local minimum. If λ is initially set to a large value the algorithm will follow the steepest descent method initially, but as λ is reduced for each solution that reduces error, as the solution approaches the local minimum the Gauss-Newton term dominates the adjustment resulting in faster convergence to the local minimum. This means, that the LM algorithm converges quickly to a minimum when the initial parameters are close to the solution, but also provides some robustness (with the gradient descent method) in cases where the initial solution may not be close to the minimum. It is this adaptive behavior that makes the LM algorithm one of the most widely used algorithm in the literature for solving the non-linear least squares optimization.

Typically in robotic applications we have an estimate of the uncertainty of the measurement vector y in the form of the covariance matrix S . This modifies the least squares problem to a weighted least squares problem and is also incorporating in LM algorithm, by replacing the Euclidean norm, with the squared Mahalanobis distance $\varepsilon^T S^{-1} \varepsilon$. The aim then is to minimize the S^{-1} .

In real time applications, e.g UAV navigation, the visual pose detection is performed over time, and the number of parameters, frames and points, keeps constantly increasing. Thus, if we were to apply bundle adjustment on all these parameters, the computational cost of a single iteration is unbounded. Given that a bundle adjustment problem may consist of several hundred camera matrices and several thousand map-points this can result in a non-linear least squares problem with hundreds of thousands of parameters. The computational complexity of the bundle adjustment problem is cubic ($O(n^3)$) based on the number of parameters, which makes directly solving the augmented normal equations a computational infeasible task for sufficiently large problems.

Accordingly, we need a constant-time algorithm, such that the cost of a single iteration does not exceed a certain threshold. It is realistic to restrict the number of frames involved, which leads to a restriction on the number of poses and points. There are two common approaches to select a subset of frames. In a more classic SFM approach, one can simply select the last m frames and therefore apply BA in a sliding window fashion. Another approach is to sample key-frames from the whole trajectory such that they approximate uniformly distributed over the explored space. A typical heuristic to achieve this is to only add a new key-frame to the back-end if the distance to the closest key-frame exceeds a threshold. This key-frame approach achieves constant-time performance only if the area of operation is bounded. Combination of both approaches is also possible, more precisely, performing BA in a sliding window over a number of spatially separated key frames, [44]. Thus, only add a new key-frame to sliding window once the camera has moved significantly far away from the previous key-frame. In this way we can deal with large scale exploration. A pure sliding window approach might have problems with varying camera motion.

However, the above mentioned, frame and sliding window reduction approaches, require a specific selection algorithm. In the case that we are not able to properly define the sliding window, or the frame removal rate, alternative methods to speed up the BA implementation have to be used. A modern approach is the implementation of a Sparse Bundle Adjustment (SBA), based on LM approaches, which take advantage of the sparsity of the bundle adjustment problem. This sparsity comes from the fact that not every map-point is observed in every image. This results in a sparse block structure of the Jacobian matrices which can be exploited to speed up computation. This can be solved by iteratively solving the augmented weighted normal equation, by replacing the Euclidean norm, with the squared Mahalanobis distance $\varepsilon^T S^{-1} \varepsilon$. The aim then is to minimize S^{-1} , as we previously comment:

$$(J^T S^{-1} J + \lambda \mathcal{I}) \delta x = J^T S^{-1} \varepsilon \quad (2.3.8)$$

A key property of the structure of the bundle adjustment problem is the inter-dependence between feature points and cameras. This leads to a sparse block structure in the normal equations. The null values in the sparse Jacobian matrix address the case where a point is not visible in an image and thus the measurements are given null weight. This sparse Jacobian structure can be exploited to reformulate the augmented normal equation 2.3.8, the full derivation can be found in [50]. The sparse structure and ordering can be exploited to improve the performance of the bundle adjustment algorithm. Using the re-formulation described in [50], the dependency between camera and point parameters is removed, hence the update δx can be solved separately for feature points and cameras.

2.4 Corner Feature Detection

Corner points have been widely used in computer vision as interest points as they are highly constrained in both axes (as opposed to lines) and are therefore easier to match. A feature detector is an algorithm which extracts visual features from images. A good feature detector is one that is both computationally efficient as well as reliable. For real-time tracking applications (visual SLAM) feature extraction will be carried out on every image, as a result in many approaches, it is vital to achieve an efficient detector, including PTAM, therefore they use the features from accelerated segment test (FAST) detector. The FAST feature detector, [51], uses the Segment-Test algorithm, [52], to determine if there is a corner feature at some image location $p(x,y)$.

Based on this Segment-Test algorithm, a circle C_{thres} with radius r and centered on pixel x,y is computed. If more than an amount of n image pixels, including on C_{thres} circle are either brighter or darker than x,y by a chosen threshold Th_{corner} , then x,y image point belongs to a corner feature, illustrated in figure 2.4.1. Adjusting the value of Th_{corner} determines the sensitivity of the corner detector, higher values of Th_{corner} result in a detector that finds fewer, but stronger, corners, a lower Th_{corner} value increases the number of corners detected, but the resulting corners have smoother gradients, which may occasionally affect repeatability.

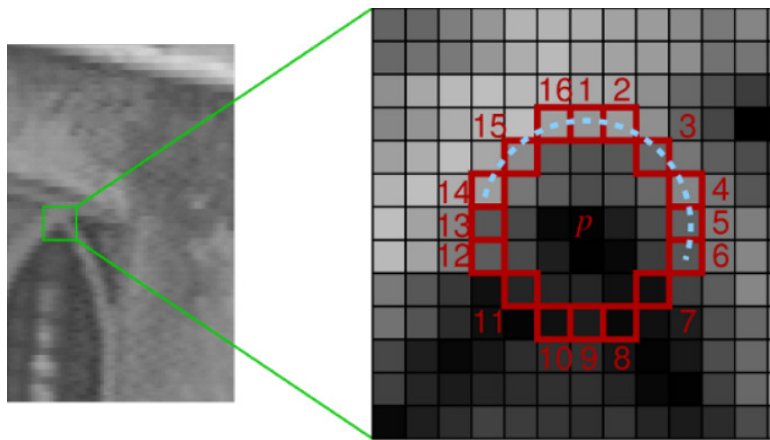


Figure 2.4.1: FAST corner detector example, here $r = 3$ and $n = 12$, forming a 12-corner detector. The dashed cyan arc highlights the pixels, which are brighter than point $p(x,y)$ and therefore p segment is identified as a corner feature.

In Figure 2.4.1, a 12-corner detector is presented, where $n = 12$. In this particular example pixels at each compass point, i.e. 1, 9, 5 and 13 can be checked first. If pixel 9 is checked first the intensity of this point is close to p , and the p cannot be identified as a corner feature. This leads to the question, whatever the value of n what is the fewest number of pixel intensity tests we can do to determine if p is a corner feature and what is the best value for n . We note that, the value of n determines the maximum angle of the corner features detected while still rejecting edges, therefore $n = 9$ is a good value as it will detect corners with the largest variety of angles while still rejecting edges. The authors in [51] demonstrated that the $FAST_9$ -corner detector, where $n = 9$, is up to twice as fast as the original detector as well as being highly repeatable

when compared to other state-of-the-art detectors. Using machine learning approaches, they were able to produce the *FAST₉*-corner detector, which only needed to test an average of 2.83 pixels to determine if a corner feature is present. This reduces the adaptability of the detector, an issue addressed in [53], with their Adaptive Generic Accelerated Segment Test (AGAST) detector. The *AGAST* corner detector uses a more generic binary decision tree instead of the learned tree used in *FAST₉*-corner detector. It also combines two decision trees, one optimized for homogeneous or cluttered image regions and the other optimized for uniform surfaces or high structured regions with texture. The *AGAST* detector switches trees based on the local structure of the image, making it adaptable to different environments without training. This adaptability is particularly useful in the application of Visual SLAM-PTAM, in outdoor scenes as the environment can change from highly cluttered scene, e.g. buildings and infrastructure, to uniform, highly textured regions such as fields or roads. A potential issue with corner and general feature detection is the problem of features detected adjacent to one another. This can lead to issues when matching features from other views in that the adjacent features could be erroneously matched given their proximity. A common solution is applied here, namely the non-maximum suppression approach, where features adjacent to one another are filtered by their relative intensity. This is done by defining a scoring function for feature points, $S(p)$, where S returns the sum of absolute difference between p and the 16 surrounding pixels. Scores of adjacent feature points are computed and the point with the lower score is discarded. Alternative scoring function can be used.

2.5 Feature Matching

Corner features detection is only one part of the problem, on the other, in order for the corner features to be useful, an approach for feature correspondences should be established, that is, determine if feature a_i in image Im_i is the same 3D world point as feature b_j in image Im_j . The local appearance of a feature in an image is subject to change in affine transformations, i.e. rotation, scale, and illumination changes, making this one of the most challenging problems in visual pose sensors. Henceforth, the discussion will be restricted to discussing methods for matching the features employed in this thesis visual pose sensor framework SLAM, namely the corner detector. One particular method commonly used in conjunction with *FAST* features is template matching. Template matching is based on a direct comparison of the intensity values I_i of a specific and often small, fixed size, i -patch of image around a detected feature image point. It is based on the assumption that the I values of the templates remain consistent between frames. In real-time tracking applications, this assumption is only valid if the motion between the frames is relative small. We can then compare image patches using the Sum of Squared Differences (*SSD*):

$$SSD = \sum_{(u,v)} ((I_1(u,v) - (I_2)(a_i + u, b_j + v))^2 \quad (2.5.1)$$

Where an exact match of the intensity values will return a score of null. This technique is later combined with an image pyramid to achieve a limited invariance to scale. An image pyramid, presented in 2.5.1, is constructed by taking the source image, and applying a smoothing filter and sub sampling it by some factor λ along each axis. This multi-scale representation of the image allows us to detect features at different scale levels providing some invariance to scale. The scaling factor λ determines the number of pyramid levels.

This patch-based approach has the advantage of being computationally efficient, however it relies on the consistency of intensity values between frames. Where there is a large difference in viewpoint the local intensity of matching templates may be vastly different.

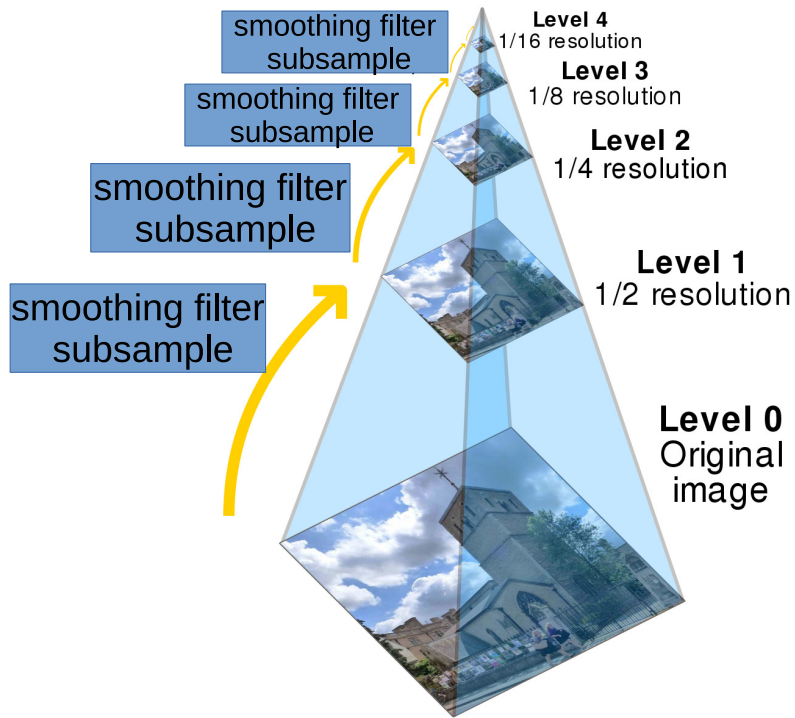


Figure 2.5.1: Image Pyramid, with five levels. Level 0 is the original image, in levels 1, 2, 3, 4 a smoothing filter is applied and the image is sub sampling each time.

Chapter 3

Control and State estimation

In the field of Mobile Robotics, state estimation and motion control present challenging problems that continuously call for improvements in the state of the art. As the hardware in robots gets faster, sensors get better with increased accuracy, and new algorithms are developed, robots become capable of executing even more impressive tasks. Relevant accomplishments rely heavily on methods that give rise to module implementations of control, localization and state estimation. Integration is also of paramount importance, as seamless integration of different modules is far from trivial.

In this chapter, we discuss the system dynamics and the essential aspects of state estimation, enabling autonomous flights with a UAV. In Section 3.1 the higher-level dynamics of UAVs are presented, which were used to implement a two loop control policy (section 3.2). More specifically, the latter consists of (a) an outer loop controller, implemented as position controller and presented in section 3.2.1, and (b) an inner loop controller that consists of the UAV provided attitude controller, described in section 3.2.2. In Section 3.3 the state estimation framework is covered. We present our proposal for a loosely coupled sensor fusion design, which nicely aligns with the employed Indirect Extended Kalman Filter (EKF) fusion approach. Additionally, we discuss and propose a visual-inertial sensor fusion approach. The propagation part of the EKF fusion framework is introduced in Sections 3.3.1, 3.3.2. Section 3.3.3 covers the EKF fusion framework update part, and multi-sensor fusion is covered in Section 3.3.4. More specifically, a dual visual and pressure measurement model is implemented. Finally, EKF task distribution is presented, to facilitate real-time UAV navigation.

3.1 UAV Motion Dynamics

Motion control is very important since it provides the actual control of the robot to allow the robot to accomplish different tasks. Example cases constitute a ground robot navigating around obstacles to get to a goal location or a manipulator picking up an object that needs to be sorted and put away. Motion control is a very large field that covers a large range of different subtopics. With the term control, we refer to position and attitude control of a UAV. Its combination with the state estimation framework from section 3.3 not only enables higher-level algorithms such as exploration or global planning, but it also allows an untrained pilot to safely operate the UAV. Both the control block and the state estimation framework are closely connected, meaning that a controller requiring state information is required in order to stabilize the vehicle. The controller separation from the estimation block, has the benefit that makes the controller independent from the employed localization methods, as long as their measurements are sufficient (in terms of observability) for the state estimator to provide state information of the UAV.

Knowing the dynamics, which are necessary for position control, trajectory generation and trajectory tracking, we present the overall system setup and show how this is integrated on the UAV produced by Ascending Technologies [7], which is used for all experimental work within this dissertation. The algorithms described in this chapter focus on hex-rotor helicopters, a common platform due to its agility and mechanical simplicity [54], although they can be applied to other aerial platforms (quad-rotor). The modeling of this UAV is based on [55], where UAV modeling with respect to aerodynamic effects, and its attitude control have been thoroughly studied. Therefore, we focus on the dynamics from higher level point of view, which are necessary for position control, trajectory generation and trajectory tracking. We omit the aerodynamic effects here.

The hex-rotor UAV setup, presented in figure 3.1.1 is an under-actuated system in which translation and rotation are coupled. For the vehicle three coordinate frames are defined, the world frame and in addition the body frame of the vehicle "I", centered around its inertial measurement unit (IMU), where $A_I^w = [x_I, y_I, z_I]$ denotes the orientation of the IMU frame expressed in the world frame. An intermediate frame "0", $A_0^w = [x_0, y_0, z_0]$, which just differs from A_I^w in rotation, and denotes the orientation of the vehicle by the yaw angle around z_w -axis. In order to simplify the transformation equations and inversions, the "0"-frame has been introduced as a leveled frame with the same yaw angle ψ as the local body frame denoted by "I" and desired accelerations in the world frame are transformed into the "0"-frame by a simple rotation through yaw angle ψ .

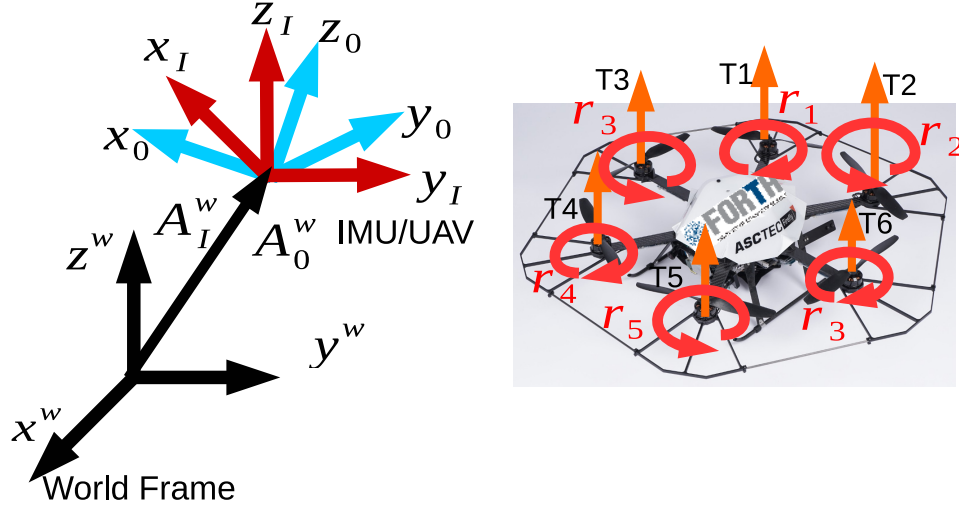


Figure 3.1.1: Right: hex-rotor used in this work, with rotor rotation directions. Left: UAV body and IMU representation frame (cyan) with respect to world reference frame (red). A_I^w notes the full attitude of the IMU-centered UAV, while A_0^w notes an intermediate frame (cyan) with a rotation of the yaw-angle ψ around z_w . The attitude angles are encoded in the rotation matrix A_I^w , as can be seen from Newton's second equation in 3.1.2, but we cannot directly invert this equation to solve for the attitude angles. In order to simplify the inversion, the 0-frame has been introduced as a leveled frame with the same ψ angle as the local body frame denoted by I .

Essentially, all dynamics of the UAV depend on the rotational velocities r_i of the individual rotors. Equation 3.1.1 maps the rotor velocities $r_1 \dots i$ over the UAV geometry and rotor constants matrix \mathcal{C}_c , and the inertia matrix \mathcal{I} to the thrust \mathcal{T} (acceleration along the body z -axis) and angular acceleration $\dot{\omega}$. This definition of thrust can always be computed from the total amount of forces and the mass m . The inertia matrix \mathcal{I} consists of the moment of inertia matrix \mathcal{M} and the mass m of the UAV.

$$\begin{bmatrix} \dot{\omega} \\ \mathcal{T} \end{bmatrix} = \mathcal{C}_c \cdot \mathcal{I}^{-1} \cdot \begin{bmatrix} r_1^2 \\ r_2^2 \\ \cdot \\ \cdot \\ \cdot \\ r_i^2 \end{bmatrix}, \mathcal{I} = \begin{bmatrix} \mathcal{M} & 0 \\ 0 & m \end{bmatrix} \quad (3.1.1)$$

From equation 3.1.1, we compute the effects of the rotational velocities of the rotors on the thrusts and torques on the rigid body of the multi-rotor with i -rotors. By changing thrust of individual rotors, the UAV is rotated in all directions, while it can only accelerate in its z -axis, giving a total of four degrees of freedom, the \mathcal{T} and $\dot{\omega}$, which we use to control the vehicle. This result indicates that the system is under-actuated since it moves and rotates in 3 D space, yielding six DoF. Besides the \mathcal{T} and $\dot{\omega}$, in order for the UAV to accelerate to a certain direction, a specific attitude is required to turn the thrust vector towards this direction. Applying Newton's second law yields the acceleration \mathbf{a} with respect to the world

frame, depending on the attitude A_I^w , gravity g and the total amount of thrust \mathcal{T} and F_T total amount of forces:

$$\mathbf{a} = A_I^w \cdot \begin{bmatrix} 0 \\ 0 \\ \mathcal{T} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, T = \frac{F_T}{m} \quad (3.1.2)$$

Taking the first derivative of \mathbf{a} , i.e.the jerk \mathbf{j} (the third derivative of position), using the chain differentiation rule as both A_I^w and \mathcal{T} are time-dependent, with $\dot{A} = A \cdot [\boldsymbol{\omega} \times]$ ($[\]$ symbolizes the a skew symmetric matrix), yields:

$$\mathbf{j} = A_I^w \cdot [\boldsymbol{\omega} \times] \cdot \begin{bmatrix} 0 \\ 0 \\ \mathcal{T} \end{bmatrix} + A_I^w \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{\mathcal{T}} \end{bmatrix} \quad (3.1.3)$$

Due to the cross product (\times), only the x - and y - components of $\boldsymbol{\omega}$ remain. It is rather straightforward that the body-fixed angular velocities $\boldsymbol{\omega}$ directly relate to \mathbf{j} . With one more derivative of position, the snap, it is easily assumed that the angular acceleration $\dot{\boldsymbol{\omega}}$ appears. In [11], the authors showing that position (p) and yaw (ψ) are differentially flat outputs of the UAV and their derivatives. The full attitude A_I^w and thrust \mathcal{T} , resulting from equation 3.1.2 and \mathbf{a}_ω denotes the current acceleration:

$$\mathbf{z}_i = \frac{t}{\mathcal{T}}, t = \mathbf{a}_\omega + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, \mathcal{T} = |\mathbf{t}|$$

$$\mathbf{y}_I = \frac{\mathbf{z}_I \times \mathbf{x}_0}{|\mathbf{z}_I \times \mathbf{x}_0|}, \mathbf{x}_0 = \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \\ 0 \end{bmatrix} \quad (3.1.4)$$

$$\mathbf{x}_I = \mathbf{y}_I \times \mathbf{z}_I$$

$$A_I^0 = [\mathbf{x}_I \quad \mathbf{y}_I \quad \mathbf{z}_I]$$

The thrust vector \mathbf{t} includes the gravity g and the current linear acceleration and defines the direction of the unit vector \mathbf{z}_i of the body fixed coordinate system. Since ψ has only a component in the z -axis, this reduces to a projection along \mathbf{z}_i :

$$\mathbf{w}_x = -\frac{\mathbf{y}_i^T \cdot \mathbf{j}}{\mathcal{T}}$$

$$\mathbf{w}_y = \frac{\mathbf{x}_i^T \cdot \mathbf{j}}{\mathcal{T}} \quad (3.1.5)$$

$$\mathbf{w}_z = \dot{\psi} \cdot \mathbf{z}_\omega^T \cdot \mathbf{z}_i$$

3.2 UAV Motion Control

In this section, we briefly describe the Position Control implementation in the UAV. While there is physically no other choice than mapping thrust linearly to the squared rotor speeds of the UAV, as introduced in Section 3.1, for the UAV motion control, the main idea is to implement a two-level loop control structure. We employ the approach proposed in [1], [14], [56], [17]. This results in having a simple low-level controller specifically designed for the given hardware in

the inner loop and at the same time provides generic controls for a high-level position or trajectory tracking controller in the outer loop, hiding vehicle specific details such as mass, inertia, or number of rotors from high-level control. The low level controllers are aimed to be simple, such that they can be implemented on computationally constrained micro-controllers. As inner loop, we use the attitude loop, while for the outer loop, a position loop.

3.2.1 UAV Position Control

For the position controller presented in this section, a control structure of relative degree two is implemented. Position and speed control are performed in one control loop based on [56], yielding desired accelerations $\mathbf{u} = \mathbf{a}$ as pseudo commands. Additionally, by the inversion shown below these commands, are turned into UAV specific low-level control commands.

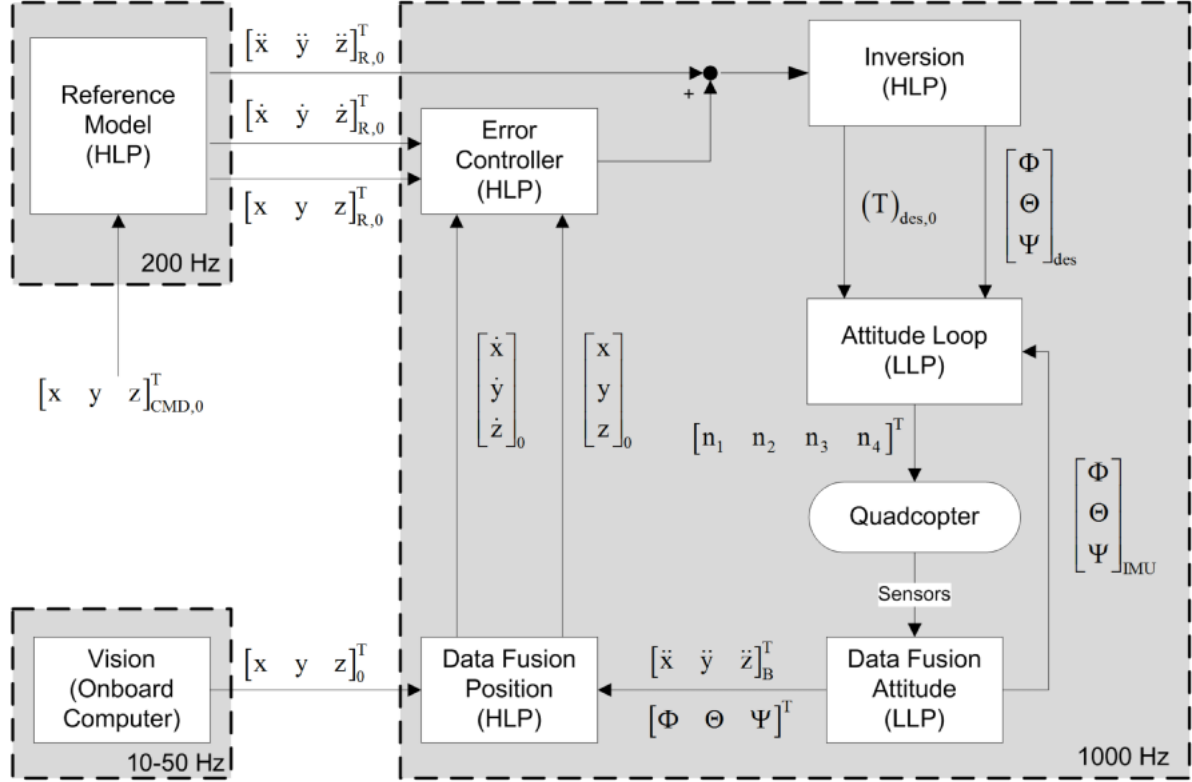


Figure 3.2.1: Structure of the position controller, [1]. Subscript "0" denotes coordinates with respect to the world frame, while subscript "B" denotes coordinates with respect to the current body frame. The names in braces denote on which physical device the corresponding part is executed. By the inversion shown, the desired attitude angles roll, pitch, and yaw $[\phi \ \theta \ \psi]_{des}^T$ and the thrust T_{des} are comprised. Outputs of the low-level attitude controller are the rotational velocities $[r_1^2 r_2^2, \dots, r_6^2]$ of the rotors.

The outer loop (Position control) is implemented on the UAV HLP micro-controller, the low level controller is on the UAV LLP micro-controller, which is a black box controller for the user from the UAV provider, and hence it is not available for any further changes in this work. However in section 3.2.2, we briefly presented the provided by the UAV LLP attitude control loop.

The LLP and HLP control structures are detailed in figure 3.2.1, including the rates of the different parts. The main task for position controller implementation is to model and invert the translation dynamics of the UAV and to achieve a linear dynamic between the inputs, the position commands and the pseudo controls. The world frame is used as inertial frame for reference trajectories generation.

By applying Newton's second law in equation 3.1.2 and using the inversion simplification of 0-frame(see figure 3.1.1), we

compute the roll angle ϕ , pitch angle θ and thrust \mathcal{T} :

$$\begin{aligned}\theta &= -\tan^{-1}\left(\frac{\mathbf{a}_{0,x}}{\mathbf{a}_{0,x} + g}\right) \\ \phi &= -\sin^{-1}\left(\frac{\mathbf{a}_{0,y}}{\mathcal{T}}\right)\end{aligned}\tag{3.2.1}$$

$$\mathcal{T} = \sqrt{\mathbf{a}_{0,x}^2 + \mathbf{a}_{0,y}^2 + (\mathbf{a}_{0,z} + g)^2}$$

These equations transform the pseudo controls $\mathbf{u} = \mathbf{a}$ or more precisely $\mathbf{u} = \mathbf{a}_0$ into the controls of the system $\theta, \phi, \mathcal{T}$. Then we apply a linear control technique, and given that a two-level controller is implemented, a simple PD control loop is used. However, the proposed implementation requires accurate knowledge of the attitude. If we employ only gyros and accelerometers as sensors, the attitude of the UAV is not observable, as shown in [57] and [31]. As a result, a wrong attitude estimate leads to a bias in the control inputs. In practice however, attitude estimation based on this information works more or less well, based on some heuristics and the designer's experience, [58].

Therefore, as we can only control the acceleration's commands (the second time derivatives of the position commands) as pseudo controls, the command trajectory needs to be sufficiently smooth, such that its second time derivative exists. Linear reference models are used to generate the reference trajectories \mathbf{p}_{Ref} . The reference dynamics, similar with [14], are set by the natural frequency ω_0 and the relative damping ζ . Hence, for the position controller presented in this work, the damping is set to 1 to ensure aperiodic behavior, and the natural frequency is set to 2.5. An example of the reference trajectories is shown in figure 3.2.2. The error PD controller computing the pseudo controls can now be designed by the following:

$$\begin{aligned}\mathbf{a}_{Ref} &= \omega_0^2 \cdot (\mathbf{p}_{Ref} - \mathbf{p}_{com}) - 2\zeta \cdot \omega_0 \cdot \mathbf{v}_{Ref} \\ \mathbf{u} &= (\mathbf{v} - \mathbf{v}_{Ref}) \cdot \mathbf{k}_d + (\mathbf{p} - \mathbf{p}_{Ref}) \cdot \mathbf{k}_p + \mathbf{a}_{Ref}\end{aligned}\tag{3.2.2}$$

Where \mathbf{p} is the estimated position, \mathbf{p}_{com} are the position commands, \mathbf{v} is the estimated velocity and \mathbf{k}_d and \mathbf{k}_p are the proportional and differential gains for the PD controller.

At this point, one advantage of the system becomes obvious: Commanding a step signal for the position, a trajectory for acceleration, speed and position is computed. The trajectory for the acceleration is directly commanded to the vehicle as feed forward commands. Theoretically, in an ideal world without disturbances and an exact model of the UAV, this would be sufficient to keep the MAV on the desired trajectory. Since this is usually not the case, the PD controller compensates for the deviations of speed and position from the desired trajectory. For way-point following, this results in high accelerations at the beginning, constant velocity during cruise flight and finally slowing down towards the desired final position.

However, the position control method presented above has limitations, namely the attitude controller on the LLP is a black box, and this results to not knowing the internal dynamics and thus cannot take full advantage of the vehicle's capabilities. Still this proposed approach works well for most applications like hovering or way-point following, but may produce wrong results for dynamic trajectory flights, [58]. Furthermore, from equations 3.1.2 and 3.2.1, the inversion from the acceleration pseudo commands \mathbf{u} to actual vehicle commands is not straight forward, and potentially costly on micro-controller hardware.

To address the latter, a solution based on [17] is implemented, where a two-loop controller design is combined with a nonlinear dynamic inversion up to the angular rates. In addition the four inputs of thrust and angular acceleration were addressed solely by using their derivatives and their flat outputs, which was able, due to that a quad-rotor is differentially flat only on four outputs, namely position and yaw, [11]. The remaining outputs of roll and pitch do not explicitly appear and are just modeled functions of the desired accelerations. Hence there were no reason for these two to be included in the controlled states, but rather, they were just need to be known, which we perfectly achieved with this work proposed state estimation framework.

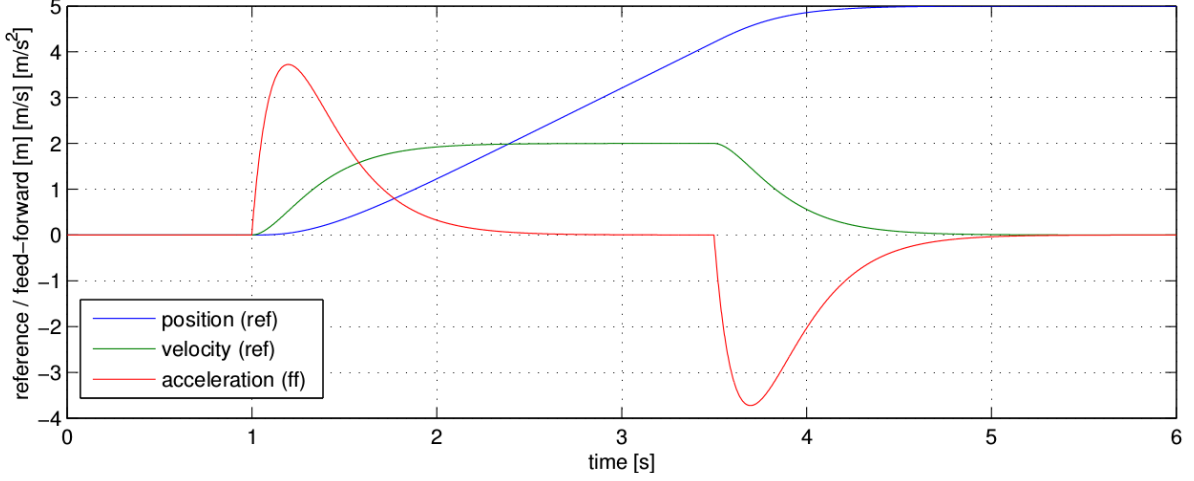


Figure 3.2.2: Response of the reference model (equation 3.2.2) on an input step of $5m$, in this case a change in the desired position. Note that the model outputs a negative acceleration after $t = 3.5s$ to slow the vehicle down in advance in order to arrive at the desired position fast, but without overshoot. By transforming the input step to a ramp, we limited the velocity. Acceleration was limited by setting ω_0 . Setting bounds in this way, the reference models stability was not influenced, [2].

Therefore, in addition with the previous mentioned PD-controller, a feedback linearization based on [56] was applied to our desired outputs, e.g. position p yaw ψ , this time up to angular rate level in contrast to the PD only position controller, resulting to the final position controller implemented in this thesis UAV. Equations 3.1.2 and 3.2.1 still hold here. For the yaw angle ψ , this was implemented using a simple P(I)- controller generating $\dot{\psi}$. From state estimation, we get estimates for position p and for velocity v . From accelerometers, accelerations are measured, expressed in IMU frame. These are bias-corrected and expressed with respect to world coordinates from the state estimation approach presented in the following sections. Using the measured accelerations, rather than the model based ones, disturbances, i.e. wind gusts and air drag were directly measured and are compensated for.

Using the conversion $\hat{\mathcal{G}} = z^T \cdot j$, we computed the desired vehicle commands. The inversion works fine for the angular rates ω , but this inversion also computes a desired change of thrust, i.e. a change of rotor speed. However, this entails some problems, since thrust is a direct function of the rotor speeds, but the only available system input. A straightforward solution would be, to obtain a thrust command, a discrete integration of \dot{T} over the control period. However, this is a poor approximation, which empirically resulted in poor performance for height control.

Therefore, as a more proper solution, a linear state feedback error controller \mathcal{K} for every single axis, that turns errors in p , v and a with respect to the desired reference into pseudo jerk, j commands, is implemented with pole placement and LQR techniques. If we hold the UAV around the hovering point, the acceleration is caused by a thrust in z_ω axis and the jerk is caused by a ω_x, ω_y acting in x_ω, y_ω axis. These two inputs are combined as soon as the UAV is not leveling anymore. Then, we obtain for world coordinates, the 3D pseudo control commands \mathbf{u}_a for acceleration and \mathbf{u}_j for jerk, with x_{ref} to be the reference for the controller:

$$\mathbf{x}_{ref}(t) = \begin{bmatrix} p_{ref}^T \\ v_{ref}^T \\ a_{ref}^T \end{bmatrix} \quad (3.2.3)$$

$$\mathbf{u}_{err} = \mathcal{K} \cdot (x_{ref} - x)$$

For a simple hovering, this is a static position p_{ref} , with the velocity v_{ref} and the acceleration a_{ref} set to null. A reference

path leading to a way-point could be generated by applying a reference model, as it is described in [17]. However, setting reasonable limits on states and control inputs, becomes more and more difficult with this approach. Furthermore, smooth trajectories through multiple intermediate points are not possible, since the vehicle has to stop at each of these intermediate points. This is best addressed using a path planing algorithm, for either planning a single segment for flying using simple way-points, or a multi-segment version for more complex trajectories through multiple intermediate points, but this would go beyond the scope of this work¹. Using an analytical solution of the desired trajectory as presented in [56], feed forward signals \mathbf{u}_{feed} (with $\mathbf{u}_{feed,a} = a(t)$ add jerk $\mathbf{u}_{feed,j} = j(t)$) for the controller are computed. This avoids lag that would incur when solely using feedback control. The final control signal is:

$$\mathbf{u} = \mathbf{u}_{feed} + \mathbf{u}_{ref} \quad (3.2.4)$$

This concludes the design of the outer loop from the two-loop control design. We have shown how to directly command and implemented angular velocities in an outer control loop. Before we describe the inner loop, where a simple angular rate controller, which provided by [7], a schematic with the full system overview is following.

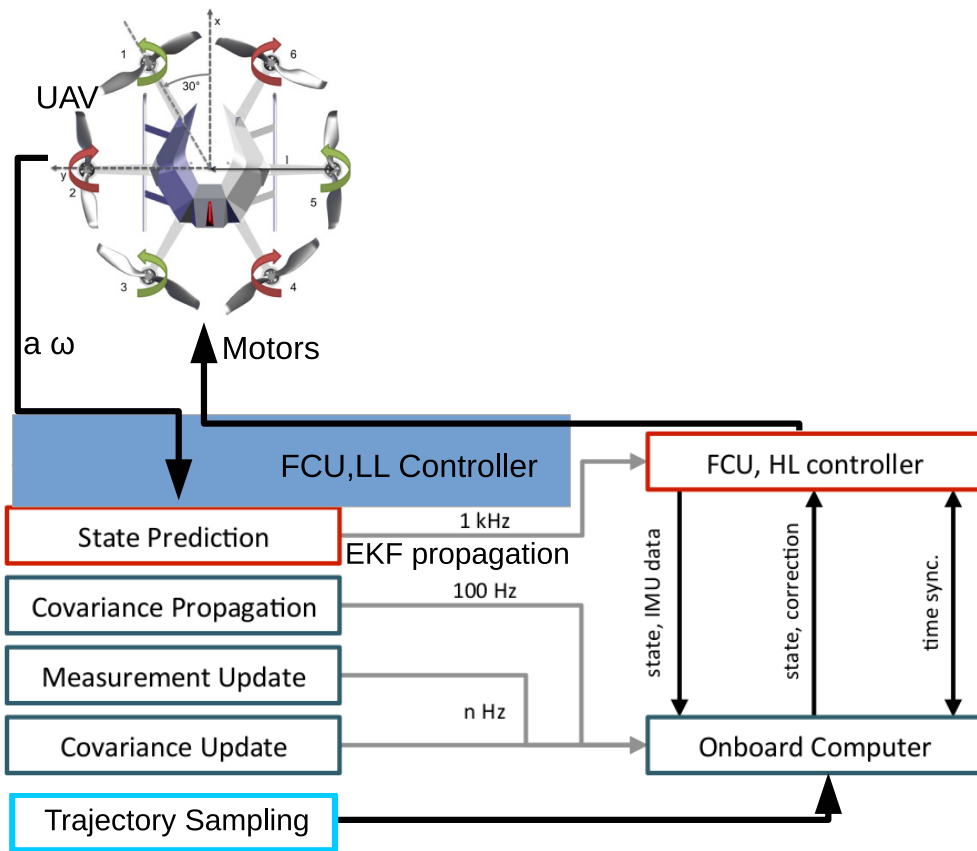


Figure 3.2.3: System overview: the computationally demanding tasks are executed on the onboard computer, while the less computationally complex but time critical parts are executed on the high-level micro processor at a rate of 1 kHz. The EKF propagation part (see Section 3.2.1) provides a current state estimate and IMU bias corrections to the controller, which obtains its reference (Ref) and feed forward (FF) signals from the trajectory generator.

Figure 3.2.3 describes the different dual-loop approach used in this thesis from the most common approach based on a position-velocity-attitude single loop. The dual loop approach increases the bandwidth and hence allows for aggressive maneuvers. Additionally, allows for a better separation of the position control problem from vehicle-specific parameters

¹A simplified approach is used in this work to circumvent this limitation, namely to send the next way-point before the vehicle slows down, and would miss the current via-point just slightly.

and the position loop can be easily implemented on any multi-rotor UAV simple controller, able to provide gyro readings, without the need of attitude estimation.

3.2.2 UAV Attitude Control - Inner Loop

For the angular rate control loop (Attitude) on ω , a simple proportional controller for each axis is used, that commands angular accelerations $\dot{\omega}$. According to inner dynamics of the UAV, described in equation 3.2.5, the $\dot{\omega}$ is converted into rotor speed commands. As we already noted, this inner Loop control is already implemented in LLP by UAV provider as presented in [1], [2]. Moreover this section briefly presents this simply proportional control and we further proposed a system parameter identification, which is required for easily implemented this attitude loop, in a variety of UAVs.

$$F_{rotor} = k_r \cdot \mathbf{r}^2$$

$$\mathbf{r} = \begin{bmatrix} \dot{\omega} \\ T \end{bmatrix} = [r_1^2 \quad r_2^2 \quad r_3^2 \quad r_4^2 \quad r_5^2 \quad r_6^2]$$

$$M = k_m \cdot F_{rotor} \tag{3.2.5}$$

$$\mathbf{u}_i = \mathcal{J}^{-1} \cdot \mathbf{K} \cdot T_r \cdot r$$

$$\mathcal{J}^{-1} = \begin{bmatrix} M & 0 \\ 0 & m \end{bmatrix}$$

where r is the rotational velocity of a rotor and k_n, k_m rotor constants. The $\dot{\omega}$ is converted into rotor speed commands in equations 3.2.5. A torque M based on F_{rotor} is created around its z -axis. The moment of UAV inertia matrix M , has only diagonal entries, while m is the UAV mass. In equations 3.2.5, the forces and torques are expressed, according to the geometry of the hex-rotor, with x - forward, y -left, z -up body coordinate system, rotor turning directions in figure 3.2.3, with l_{boom} the boom length with respect to the center of gravity. The T_r in equations 3.2.5, is the transformation allocation matrix for a common hex-rotor and $\mathbf{K} = \text{diag}([l \cdot K_r \quad l \cdot K_r \quad K_r \cdot K_m \quad K_r])$ is the parameter matrix, where we already factored out all vehicle specific parameters. Solving equations 3.2.5 for r , the rotor speeds $r_i, i = 1, \dots, 6$ are acquired:

$$r = T_r^\dagger \cdot k_r^{-1} \cdot \mathcal{J} \cdot \mathbf{u}_i \tag{3.2.6}$$

The operations in equation 3.2.6 are critical to accomplish fast computations. This is achieved, as the matrices k_r^{-1} and \mathcal{J} are diagonal, the pseudo-inverse T_r^\dagger is the most energy efficient solution as it minimizes the Euclidean norm of r , and since the UAV specific parameters are factored out beforehand, the T_r^\dagger is stored and computed only once. Furthermore every entry just affects a single row of T_r^\dagger .

As we already mentioned, the final part of inner-loop controller designing is the parameter system estimation. Only a few parameters, mainly l and m can easily and accurately be measured. In contrast the matrix \mathcal{J} and the rotor constants K_m and K_r are subject to errors. To make matters worse, if parameters are measured separately, errors might accumulate and hence return erroneous parameter values. Since k_r^{-1}, \mathcal{J} are diagonal, and the pseudo-inverse T_r^\dagger is pre-computed, only one parameter in each row has to be identified more precisely, and accordingly:

$$\mathbf{K} = \text{diag}([K_{\omega_x} \quad K_{\omega_y} \quad K_{\omega_z} \quad K_T]) \tag{3.2.7}$$

$$\mathbf{u}_i = \text{diag}(\mathbf{K}) \cdot T_r \cdot r$$

These parameters can be estimated by contacting multiple experiments, e.g. manually-controlled flights exhibiting some excitation to the UAV. All that is needed are acceleration measurements in the body z -axis, angular rate measurements

from the gyros and the rotor speeds. As state estimation is available and run in parallel in our fusion framework, even gyro and acceleration biases are also compensated. We do not provide an in depth study of system identification parameters here, as this would go beyond the scope of this work. Nevertheless, values of \mathbf{K} were determined in this thesis as follows, which may serve as a good starting point, but are subject to vary, depending on the vehicle setup: $\mathbf{K} = [5.54 \cdot 10^{-5} \quad 5.01 \cdot 10^{-5} \quad 2.76 \cdot 10^{-6} \quad 6.56 \cdot 10^{-5}]$ based on [2] and provided by UAV manufacturer. The values for the x- and y- axes are not the same, for the reason that UAV used in this thesis, the UAV mass of the battery is more centered around the x-axis of the helicopter, causing different moment of inertia.

3.3 Sensor Fusion

The controllers presented in the last section employ simple, but reliable and well tested control structures, and rely on accurate information on the UAV state. In Section 3.1, we learned that at least the quantities position, velocity and attitude are necessary for the attitude command-based controller. To comprehend with UAVs fast dynamics, state estimates are required to be provided at a sufficiently high rate with as little delay as possible.

In the following chapters, we discuss proper sensor fusion and (inter-) sensor calibration of the full sensor suite. In a first step, we focus in particular on our basic setup consisting of a single camera and an IMU as the only sensors for locally stable UAV navigation. In this work, we propose a basic fusion system, which calibrates itself online while flying. Thus, computationally complex methods are unsuitable. Instead, the non-linear observability analysis, proposed in [3], reveals that the vision algorithm is decoupled by treating the arbitrarily scaled 3D camera velocity or 6Dof pose as measurements. The state estimator which is also responsible for the inter-sensor calibration has constant complexity, since these measurements have constant size, [3].

In a second step, we add further sensors for drift compensation. State-of-the-art techniques mainly focuses either on the calibration of the sensors or on the pose estimation assuming known calibration. Furthermore, most of these approaches are poorly scalable and can thus barely be used for motion estimation in large real-time environments. This thesis proposed approach is based on an indirect Extended Kalman Filter (EKF) sensor-fusion approach. For fusing the IMU data with the visual input, alternative methods to consider are a least squares approach and Unscented Kalman Filter (UKF), but the UKF is too costly for on-board processing. Promising results based on least squares approaches are shown in in [59], where the comparison of the integrated inertial values and the visual pose acquired by using the former least squares optimization. However these least squares approaches are not robust enough, mainly because of the least squares approach sensitivity to outliers. In addition, least squares approaches do not estimate the robot's velocity and position in order to control it efficiently.

Furthermore, a set of carefully selected additional sensors are used to increase the autonomy of a UAV, hence improve robustness of state estimation and navigation. However, each additional sensor adds to the payload. As a rule of thumb, every 10 grams require 1W of motor power in hover mode for a small UAV. As in any multi-sensor system, the (inter-)sensor calibration is crucial to the robustness of our estimation processes. While it is assumed that the intrinsic camera parameters to be known and fixed, the inter-sensor calibration parameters describing the 6 DoF pose between the IMU and the camera and additional sensors are unknown. There exist various methods in the literature to calibrate these unknowns. Particular attention has been paid to calibrate the IMU and camera calibration parameters [26], [27]. However, these approaches usually address off-line calibration exhibiting complexity of at least $O(M^2)$ for M features observed by the camera, [3].

In Addition, the proposed in this thesis EKF approach is independent of the underlying vision algorithm which estimates the camera pose. This has the advantage that the algorithm operates at a constant computational complexity. This allows us to treat the visual framework as a black box and thus the approach is modular and widely applicable to existing (monocular) solutions. It is used with any 6 DoF pose estimation algorithm such as visual odometry or visual SLAM in monocular or stereo setups, depth and ICP based range finder approaches (Kinect, 3D laser scanners, etc), or external estimators like Vicon, AR-Toolkit etc. In particular, we are independent of the computationally costly vision based EKF-SLAM for camera pose estimation. This results to a very versatile EKF solution to control a robot efficiently and at high frequency in metric position and velocity, as shown in figure 3.3.1. A similar to this thesis EKF approach was proposed in [3], [60]. Alternative solution to the above monocular visual-inertial approaches is a vision only approach. However, a major issue

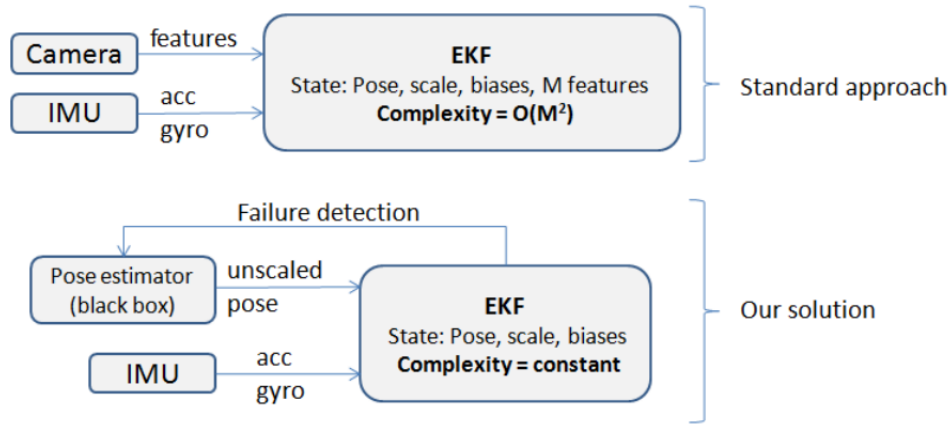


Figure 3.3.1: top: Standard (tightly design) EKF-SLAM approach exhibits computational complexity of at least $O(M^2)$ with M the number of features. bottom: In contrast, the proposed approach (loosely coupled design) runs at constant computational complexity. Moreover, by treating the pose estimation part as a black box, but still able to detect failures and drifts of the black box, our approach is independent of the underlying pose estimation method. The whole framework inherits thus the complexity of the chosen pose estimator, [3].

of vision based UAV navigation using one camera as the only exteroceptive sensor is the recovery of the metric error between the desired vehicle pose and the actual pose. Monocular approaches inherently suffer from the lack of absolute scale. Apart from having known visual landmarks one can think of a variety of options to retrieve the absolute scale. While infrared-deep sensors are too sensitive to sun light, ultra sound sensors lack in their range in which they reliably can measure and the distance laser rangefinders are too heavy. Other choices include the addition of a second camera (stereo-vision), but this solution loses its range measurement ability for scenes far away with respect to the stereo baseline. This leaves only the choice of fusion visual measurements with additional sensors with additional sensors, e.g. a pressure sensor or an inertial sensor (IMU).

However, the first option is not very reliable indoors (sudden pressure changes when closing doors, activating air conditioning etc), and thus the IMU remains the obvious choice. Therefore, in this thesis we employ an Inertial Measurement Unit (IMU) comprising of a 3-axes gyroscope measuring angular velocities, and 3-axis linear acceleration sensor as the basic sensors of our state estimation framework. This sensor selection is not a restriction since almost all airborne vehicles are able to carry an IMU. IMUs are based to Micro Electro Mechanical System (MEMS) and are cheap, small and lightweight such that they are applicable on almost any robotic platform. Compared to high-end expensive IMUs (and significantly heavier), measurements from less expensive and affordable inertial sensors typically used on-board UAVs, are corrupted by a time-varying bias, resulting for instance from changes of temperature. As a result, solely time-discrete integration (dead-reckoning) of these sensors makes a steadily accurate estimation of the absolute pose of the vehicle nearly impossible.

As already mentioned, this is remedied by the combined use of IMU and a 6 DoF pose estimation (Camera). While an IMU provides fast, but drifting measurements in the order of up to 1 kHz, the 6 Dof sensors provide absolute (or at least slowly drifting) measurements – but usually at a much lower rate in the range of 1-50 Hz.

3.3.1 Extended Kalman Filter Framework

The EKF framework presented in this thesis, consists of two main steps: (a) the prediction (“time update”) step and (b) the update (“measurement correction”) step. During time update, the states and the uncertainty of a process are predicted using the time-varying process model of a system, to which we will refer to as *state prediction and covariance propagation* respectively. During the measurement correction, the measurement from a sensor is applied to a sensor model, which is then used to correct the state and to update the uncertainty of the state estimate. We will refer to these steps as *state update and covariance update*. This algorithmic separation is used mainly to distribute the computational load to different

processing units on the UAV. We give an overview of the underlying structure of the implemented EKF framework and then discuss measurement updates that are computed from readings of additional sensors used in this thesis. First we present a overview of EKF selected model and designing general directions and in the following sections the main parts of the implemented EKF are described.

The process model (state prediction) used for this thesis is the IMU model driven, where IMU measurements, comprising of linear acceleration and angular velocity, are used to drive the process model of the EKF. Essentially, acceleration readings are integrated to velocities, which are further integrated to position. Angular velocity readings are integrated to orientation. If we have selected the IMU measurements as EKF measurement updates, the whole EKF procedure has to be computed for every single IMU measurement. As these measurements arrive at rates of up to 1-2 kHz, this would soon get computationally too expensive. In addition, the IMU based model allows for a better separation of this EKF framework tasks with respect to computational complexity, allowing us to further optimize for high update rate and low delays.

Moreover, the UAV process model is not used as it implements as model inputs the torques and thrusts originating from each rotor or the total thrust and angular momentum that rotors cause. This requires additional vector-states in the process model formulation, in order to apply the integration procedure for snap, jerk, acceleration, velocity and position. This is not robustly enough for constrained system implementations, since EKF update and especially covariance prediction have to run at high frequencies and involve dense matrix multiplications and IMU measurements, as EKF measurement updates.

In addition, if the model based approach is applied on a UAV, this approach relies on accurate system identification. Thus system identification implies that mass, moment of inertia and rotor constants are required, to obtain the UAV thrust and torque. To make things more complicated, mass and moment inertia may change due to different sensors or payloads, and constants of rotors that typically are used, may change over time. This would require again additional (non-linear) EKF states for estimation of these model parameters to be integrated. In contrast, the IMU based process model approach would simply measure the current linear acceleration and angular velocity, which are further integrated to position and orientation.

Below we assume that the inertial measurements contain a certain bias \mathbf{b} and white Gaussian noise n . Thus, for the real angular velocities $\boldsymbol{\omega}$ and the real accelerations \mathbf{a} in the IMU frame we have:

$$\boldsymbol{\omega} = \boldsymbol{\omega}_{mes} - \mathbf{b}_{\boldsymbol{\omega}} - \mathbf{n}_{\boldsymbol{\omega}} \quad (3.3.1)$$

$$\mathbf{a} = \mathbf{a}_{mes} - \mathbf{b}_a - \mathbf{n}_a$$

where the subscript *mes* denotes the measured value. The dynamics of the non-static biases \mathbf{b} are modeled as a random process:

$$\dot{\mathbf{b}}_{\boldsymbol{\omega}} = \mathbf{n}_{b_{\boldsymbol{\omega}}} \quad (3.3.2)$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{b_a}$$

For a detailed analysis of the characteristics in the continuous time and in discrete time domain, we refer to the work in [3], [61]. The state of the filter is composed of the position p_I^w of the IMU, its velocity v_I^w and its attitude quaternion q_I^w in the world frame w , describing a rotation from the world frame w into the IMU frame I . We also estimate the gyro and acceleration biases $\mathbf{b}_{\boldsymbol{\omega}}$ and \mathbf{b}_a , as well as the visual scale factor λ . These two bias states are essential to be incorporated in the state and to be continuously estimated, since those states drift over time and thus cannot be calibrated offline.

The calibration states are the rotation from the IMU frame into the camera frame q_I^c , and the position of the camera center in the IMU frame p_I^c . Additionally, the drifts between the black boxed visual frame v and the fixed world frame w are included. The rotational drifts are reflected in q_w^v and the translational ones in p_w^v . The gravity vector is not included in the state space, since our world reference frame is aligned with it and thus known. The gravity vector has to be estimated because the reference frame was the drifting and unaligned vision frame, [26]. Our entire state yields a 31-elements state vector x :

$$\mathbf{x} = [p_I^{wT} \quad v_I^{wT} \quad q_I^{wT} \quad b_\omega^T \quad b_a^T \quad \lambda \quad p_I^c \quad q_I^c \quad p_v^w \quad q_v^w] \quad (3.3.3)$$

The following differential equations govern the state. Converting quaternion q_I^w in Hamilton notation², q_{-I}^w :

$$\begin{aligned} \dot{p}_I^w &= v_I^w \\ \dot{v}_I^w &= A_I^w \cdot (a_{mes} - b_a - n_a) - g \\ \dot{q}_{-I}^w &= \frac{1}{2} \cdot q_{-I}^w \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{mes} - b_\omega - n_\omega \end{bmatrix} \\ \dot{b}_\omega &= n_{b_\omega} \\ \dot{b}_a &= n_{b_a} \\ \dot{\lambda} &= 0 \\ \dot{p}_I^c &= 0 \\ \dot{q}_I^c &= 0 \\ \dot{p}_v^w &= 0 \\ \dot{q}_v^w &= 0 \end{aligned} \quad (3.3.4)$$

With g as the gravity vector expressed in the world frame and A_I^w the rotation matrix computed from q_{-I}^w . Here the zero motion model is applied, assuming that the scale and visual frame V drift spatially and not temporally. Thus, as we employ a visual framework to calculate the camera pose, the covariance of the visual scale λ and the visual frame drift states p_v^w, q_v^w are to be augmented upon frame creation. The non-linear observability analysis presented in [3] reveals that the bias states are always observable as long as there exists a measurement update that contains either a position or velocity measurement. This is the case for the sensors setups, and is thus an important precondition for or modular sensor fusion setup.

3.3.2 EKF Error State Representation

A very important decision in the implementation of the EKF framework, in conjunction with IMU used systems, is the use of the indirect instead of the direct form, also referred to as the error- state and the total state formulation, respectively. As the name indicates, in the total state (direct) formulation, total states such as orientation are among the variables in the filter, and the measurements are sensors outputs. In the error state (indirect) form, the errors in orientation are among the estimated variables, and each measurement presented to the filter is the difference between the sensors and the external source data. There are some serious drawbacks to the direct filter implementation. Being in the system loop and using the total state representation, the filter would have to maintain explicit, accurate awareness of the vehicle's angular motion i.e. incorporate a dynamic model, as well as attempt to suppress noisy and erroneous data at a relatively high frequency. Another disadvantage of the direct filter design is that if an error occurs in the filter estimation the entire navigation algorithm will fail. Instead the error filter in case of a failure can continue to provide estimates by acting as an integrator on the sensors data. This makes the direct filter a tool that would be ideal for fault detection and identification purposes. Instead, the implemented error-state EKF filter estimates the errors in the navigation and attitude information using the difference between the IMU system and external sources of data (visual). The IMU itself is able to follow the high frequency motions of the vehicle very accurately, and there is no need to model these dynamics explicitly in the filter. In addition, the dynamics upon which this error filter is based are a set of inertial system error propagation equations which are low frequency and very adequately represented as linear. Because the implemented EKF is out of the loop and is based on low frequency dynamics, its sampling rate is much lower than that of a direct filter. In fact, this error filter is developed with a sample period (of the external source) of the order of minutes. For the above mentioned reasons, the error state formulation is used in essentially all terrestrial navigation system aided by inertial (IMU) data, [64], and therefore is this thesis EKF design selection as we already mentioned.

²The equations defining rotations look slightly different here, since we thoroughly use the Hamilton quaternion notation ([62]), as opposed to the NASA JPL [63] convention used in [61] and Weiss (2012), [3]. However, the final result for the error quantities is the same.

In the above described state representation, section 3.3.1, quaternions are used as attitude description. As discussed above, it is more efficient in such cases to implement the EKF state as error (indirect) state, thus we have the option to represent the error and its covariance in terms of an arithmetic difference or with the aid of an error quaternion. If an arithmetic difference error representation is used, as attitude quaternion is converted to have unit length for handling the quaternion in its minimal representation, effectively a corresponding singular covariance matrix results. This leads to increased numerical instability in error quaternion conversion. Instead of the arithmetic difference, if the error between the estimate \hat{q} of a quaternion and its true value \bar{q} is defined as a small rotation $\delta\bar{q}$, this attitude quaternion is converted to have unit length for handling the quaternion in its minimal representation with increased numerical stability. This rotation is assumed to be small, which allows to apply the small angle approximation with the error angle vector $\delta\theta$, [61].

$$\begin{aligned}\bar{q} &= \hat{q} \otimes \delta\bar{q} \\ \delta\bar{q} &\approx \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta^T \end{bmatrix}\end{aligned}\quad (3.3.5)$$

Similarly, the error for rotation matrices is expressed as:

$$\begin{aligned}A &= \hat{A} \cdot \Delta A \\ \Delta A &\approx I_3 + [\delta\theta \times]\end{aligned}\quad (3.3.6)$$

Then, we redefine state vector 3.3.3 as a the 28-elements error state vector, with an estimate \hat{x} to its quantity x , i.e. $\tilde{x} = x - \hat{x}$:

$$\tilde{\mathbf{x}} = [\delta p_I^w \quad \delta v_I^w \quad \delta \theta_I^w \quad \delta b_\omega^T \quad \delta b_a^T \quad \delta \lambda \quad \delta p_I^c \quad \delta q_I^c \quad \delta p_v^w \quad \delta q_v^w]\quad (3.3.7)$$

Finally, we obtain the differential equations governing the error state, according to indirect EKF implementation in [61], [31] but using Hamilton quaternion notation:

$$\begin{aligned}\delta \dot{p}_I^w &= \delta v_I^w \\ \delta \dot{v}_I^w &= \hat{A}_I^w \cdot [(a_{mes} - \hat{b}_a) \times] \cdot \delta \theta_I^w - \hat{A}_I^w \cdot \delta b_a - \hat{A}_I^w \cdot n_a \\ \delta \dot{\theta}_I^w &= -[(\omega_{mes} - \hat{b}_a) \times] \cdot \delta \theta_I^w - \delta b_\omega - n_\omega \\ \delta \dot{b}_\omega &= n_{b_\omega} \\ \delta \dot{b}_a &= n_{b_a} \\ \delta \dot{\lambda} &= 0 \\ \delta \dot{p}_I^c &= 0 \\ \delta \dot{q}_I^c &= 0 \\ \delta \dot{p}_v^w &= 0 \\ \delta \dot{q}_v^w &= 0\end{aligned}\quad (3.3.8)$$

This is summarized to the linearized continuous-time error state equation with respect to error state vector x , with $\hat{\omega} = \omega_{mes} - \hat{b}_\omega$, $\hat{a} = a_{mes} - \hat{b}_a$ and with n being the estimate error state noise vector $n = [n_a^T \quad n_{b_a}^T \quad n_\omega^T \quad n_{b_\omega}^T]$ around a continuous (subscript ‘‘cont’’) $\hat{\mathbf{x}}_{cont}$, :

$$\dot{\tilde{\mathbf{x}}} = \frac{\partial \tilde{\mathbf{x}}}{\partial \tilde{\mathbf{x}}} \Big|_{\hat{\mathbf{x}}_{cont}} \cdot \tilde{\mathbf{x}} + \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{n}} \Big|_{\hat{\mathbf{x}}_{cont}} \cdot \mathbf{n} = \Phi_{cont} \cdot \tilde{\mathbf{x}} + \Gamma_{cont} \cdot \mathbf{n}\quad (3.3.9)$$

To increase the speed of the algorithm calculations, the Φ , Γ are assumed to be constant over the integration time step between two consecutive state propagation steps. For discretization, i.e. obtaining the discrete (subscript ‘‘disc’’) time version of the system propagation matrix Φ :

$$\Phi_{disc} = \exp(\Phi_{cont} \Delta t) = I_{disc} + \Phi_{cont} \Delta t + \frac{1}{2!} \Phi_{cont}^2 \Delta t^2 + \dots\quad (3.3.10)$$

This allows us to express Φ_{disc} exactly without approximation and to use sparse matrix operations later in the implementation of the filter:

$$\Phi_{disc} = \begin{bmatrix} I_{3_{dist}} & \Delta t & A & B & M & 0_{3 \times 13} \\ 0_3 & I_{3_{dist}} & C & D & L & 0_{3 \times 13} \\ 0_3 & 0_3 & E & F & 0_3 & 0_{3 \times 13} \\ 0_3 & 0_3 & 0_3 & I_{3_{dist}} & L & 0_{3 \times 13} \\ 0_3 & 0_3 & 0_3 & 0_3 & I_{3_{dist}} & 0_{3 \times 13} \\ 0_{3 \times 13} & 0_{3 \times 13} & 0_{3 \times 13} & 0_{3 \times 13} & 0_{3 \times 13} & I_{13_{dist}} \end{bmatrix} \quad (3.3.11)$$

Now for obtaining matrix blocks compact solution A, B, C, D, E, F , using the l'Hopital rule and the small angle approximation and in addition computing the L, M :

$$\begin{aligned} \mathbf{L} &= -A_{\hat{q}_l^w}^T \frac{\Delta t^2}{2} \\ \mathbf{M} &= -A_{\hat{q}_l^w}^T \Delta t \\ \mathbf{A} &= -A_{\hat{q}_l^w}^T [\hat{a} \times] \left(\frac{\Delta t^2}{2} - \frac{\Delta t^3}{3!} [\omega \times] + \frac{\Delta t^4}{4!} [\omega \times]^2 \right) \\ \mathbf{B} &= -A_{\hat{q}_l^w}^T [\hat{a} \times] \left(\frac{\Delta t^3}{3!} - \frac{\Delta t^4}{4!} [\omega \times] + \frac{\Delta t^5}{5!} [\omega \times]^2 \right) \\ \mathbf{C} &= -A_{\hat{q}_l^w}^T [\hat{a} \times] \left(\Delta t - \frac{\Delta t^2}{2!} [\omega \times] + \frac{\Delta t^3}{3!} [\omega \times]^2 \right) \\ \mathbf{D} &= -A \\ \mathbf{E} &= -I_{disc} - \Delta t [\omega \times] + \frac{\Delta t^2}{2!} [\omega \times]^2 \\ \mathbf{F} &= -\Delta t [\omega \times] + \frac{\Delta t^2}{2!} [\omega \times]^2 + \frac{\Delta t^3}{3!} [\omega \times]^3 \\ \mathbf{J}_\bullet &= -A_{\hat{q}_l^w}^T \end{aligned} \quad (3.3.12)$$

and Γ , using equation 3.3.9:

$$\Gamma_{cont} = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ J_\Gamma & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & -I_{3_{dist}} & 0_3 \\ 0_3 & 0_3 & 0_3 & I_{3_{dist}} \\ 0_3 & I_{3_{dist}} & 0_3 & 0_3 \\ 0_{3 \times 13} & 0_{3 \times 13} & 0_{3 \times 13} & 0_{3 \times 13} \end{bmatrix} \quad (3.3.13)$$

The discrete time system noise covariance matrix Q_{disc} as suggested in [65] and with Q_{cont} the continuous time system noise covariance matrix:

$$Q_{cont} = \text{diag}(\sigma_{n_a}^2, \sigma_{n_{b_a}}^2, \sigma_{n_\omega}^2, \sigma_{n_{b_\omega}}^2) \quad (3.3.14)$$

$$Q_{disc} = \int_{\Delta t} \Phi_{disc}(\tau) \cdot \Gamma_{cont} \cdot Q_{cont} \cdot \Gamma_{cont}^T \cdot \Phi_{disc}^T(\tau) d\tau$$

With the computation of the discretized error state propagation and the error process noise covariance matrices, the propagation steps for the implemented EKF prediction step are as follows:

EKF Prediction Module

1. Propagate the state variables according to equations 3.3.4 using the first order integration quaternion transform (equivalent to solving the first order \dot{q}_{-I}^w quaternion differential equation, as described in [61] and making the assumption of a linear evolution of ω in the integration interval.
2. Compute the error state variables according to equations 3.3.8, using equations 3.3.5, 3.3.6, 3.3.7.
3. Calculate Φ_{disc} , Q_{disc} .
4. Compute the propagated state covariance matrix according to the regular EKF equation:

$$\mathbf{P}_{k+1|k} = \Phi_{disc} \mathbf{P}_{k|k} \Phi_{disc}^T + Q_{disc} \quad (3.3.15)$$

3.3.3 EKF Measurement Model

While body accelerations and angular velocities are used for the proposed in this thesis EKF state propagation discussed above, making the IMU an indispensable sensor for the system as the basic propagation node. However to enable robust UAV flights, we need additional sensing modalities measurements, to account for the temporal drift of the sensor used in the basic node. The mentioned measurements are used for the filter update. In particular, here we focus on a standard sensor type, with 6 DoF pose sensor measurements, that are acquired from a monocular camera or motion capture system (Vicon). Usually the first problem that occurs with the addition of the extra sensing measurements is that its pose center does not coincide with the origin of the IMU. That said a pose (displacement and rotation) calibration is required. Obtaining this calibration manually is tedious, and especially complicated and error prone for the rotation of the additional sensor with respect to the IMU. The calibration for both 6 DoF or even 3DoF (e.g. GPS) consists of the displacement vectors p_I^s and rotation q_I^{-s} , with respect to the IMU frame and to the measurement sensor frame. Figure 3.3.2 depicts the setup with the coordinate frames, sensors and state variables.

In addition, the position measurement for both sensor types may be scaled, in case for monocular vision can detect the direction of motion, but the magnitude only up to a scaling factor λ . The non-linear observability analysis in [3], [26], [27], [66] reveals that all states are observable including the calibration states as long as the IMU's accelerometers and gyroscopes are excited in at least two axes.

In real scenarios, pose (attitude and position) measurement from the mentioned sensors, may possibly drift spatially. As additional states included by the measurements sensors do not have any dynamics in state propagation, the calibration does not change over time. On the other hand, scale drifts due to the scale propagation error, usually occurring in vision based 6 DoF pose measurements are spatial. Scale changes only when the visual-sensor moves and new features get into the field of view. Visual pose estimation algorithms and their derivatives usually define their frame of reference at an arbitrary position in the world frame. This means that all measurements are expressed with respect to the visual algorithm's reference frame. Obtaining a p_s^w position measurement from a 6 DoF sensor, with \mathbf{A}_I^w as the IMU's attitude and \mathbf{n}_p the position noise, we have the following measurement model implemented:

$$\mathbf{z}_p = \mathbf{p}_s^w = (\mathbf{p}_I^w + \mathbf{A}_I^w \cdot \mathbf{p}_I^s) \cdot \lambda + \mathbf{n}_p \quad (3.3.16)$$

The above model is expressed in terms of estimated quantities and error quantities:

$$\begin{aligned} \mathbf{p} &= \hat{\mathbf{p}} + \Delta \mathbf{p} \\ \mathbf{A} &= \hat{\mathbf{A}} \cdot \Delta \mathbf{A} \approx \hat{\mathbf{A}} \cdot (I_3 + [\delta \theta \times]) \end{aligned} \quad (3.3.17)$$

We define the position error as:

$$\begin{aligned} \tilde{\mathbf{z}}_p &= \mathbf{z}_p - \hat{\mathbf{z}}_p = (\mathbf{p}_I^w + \mathbf{A}_I^w \cdot \mathbf{p}_I^s) \cdot \lambda + \mathbf{n}_p - (\hat{\mathbf{p}}_I^w + \hat{\mathbf{A}}_I^w \cdot \hat{\mathbf{p}}_I^s) \cdot \hat{\lambda} \\ &= (\Delta \mathbf{p}_I^w + \hat{\mathbf{A}}_I^w \cdot \Delta \mathbf{p}_I^s) \cdot \hat{\lambda} - \hat{\mathbf{A}}_I^w \cdot [\hat{\mathbf{p}}_I^s \times] \cdot \delta \theta \cdot \hat{\lambda} + \hat{\mathbf{p}}_I^w \cdot \Delta \lambda + \hat{\mathbf{A}}_I^w \cdot \hat{\mathbf{p}}_I^s \cdot \Delta \lambda \end{aligned} \quad (3.3.18)$$

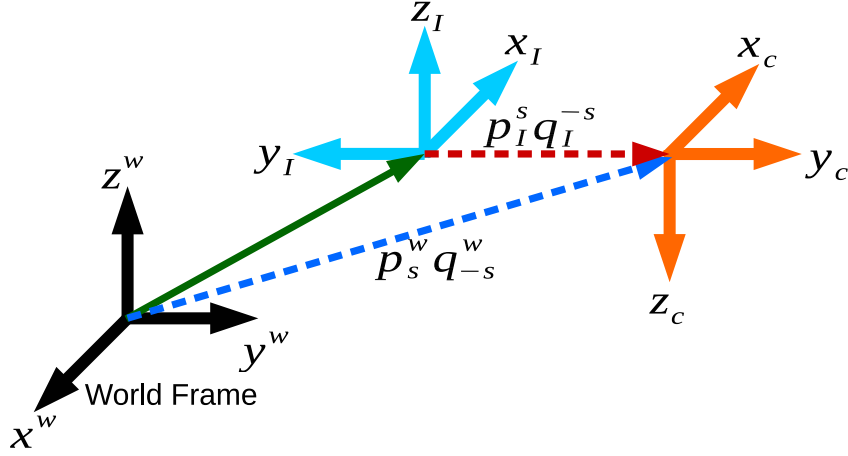


Figure 3.3.2: Setup depicting the UAV body with one sensor. For UAV navigation, we are interested in obtaining an estimate of p_I^w and q_{-I}^w denoted by the green arrow. To accomplish this, we compute auxiliary states denoted by the red arrow and measurements p_s^w and q_{-s}^w denoted by the blue arrow. Depending on the type of sensor, only one of the measurements may be available, for example 6 DoF pose sensors measures both position and attitude, in contrast 3 DoF pose sensors, only return the position. The additional states do not have any dynamics during the propagation phase.

For the rotation measurement, we apply again the notion of an error quaternion. The 6 DoF pose algorithm yields the rotation from the world frame to the sensor frame q_{-s}^w with δq_n as small measurement noise:

$$\mathbf{z}_q = q_{-s}^w = q_{-I}^w \otimes q_I^{-s} \otimes \delta q_n \quad (3.3.19)$$

and the estimated rotation error:

$$\tilde{\mathbf{z}}_q = \mathbf{z}_q - \hat{\mathbf{z}}_q = \delta \theta = \hat{\mathbf{A}}_I^{s^T} \cdot \delta \theta_I^w + \delta \theta_I^s + \delta \theta_n \quad (3.3.20)$$

which is linearized using the general approach the $\tilde{\mathbf{z}} = \mathbf{H}\tilde{\mathbf{x}}$.

Finally the measurements are stacked together as :

$$\begin{bmatrix} \tilde{\mathbf{z}}_p \\ \tilde{\mathbf{z}}_q \end{bmatrix} = \begin{bmatrix} H_p \\ 0_{3 \times 6} H_I^w 0_{3 \times 10} H_I^{-s} \end{bmatrix} \quad (3.3.21)$$

With equations 3.3.18, 3.3.20, the EKF Jacobians \mathbf{H}_p , \mathbf{H}_q , \mathbf{V}_p , \mathbf{V}_q for position $\tilde{\mathbf{z}}_p$ and attitude $\tilde{\mathbf{z}}_q$ are computed. The Jacobians are required for the EKF measurement and covariance update:

$$\begin{aligned}
\mathbf{H}_p &= \frac{\partial \tilde{z}_p}{\partial \tilde{\mathbf{x}}} |_{\hat{\mathbf{x}}} \\
\mathbf{H}_q &= \frac{\partial \tilde{z}_q}{\partial \tilde{\mathbf{x}}} |_{\hat{\mathbf{x}}} \\
\mathbf{V}_p &= \frac{\partial \tilde{z}_p}{\partial \tilde{n}_p} |_{\hat{\mathbf{x}}} \\
\mathbf{V}_q &= \frac{\partial \tilde{z}_p}{\partial \tilde{\delta \theta}} |_{\hat{\mathbf{x}}}
\end{aligned} \tag{3.3.22}$$

With the definition of the matrices above, we now proceed with the proposed in this framework EKF update steps:

EKF Update Module

1. Compute the position residual $\tilde{\mathbf{z}}_p = \mathbf{z}_p - \hat{\mathbf{z}}_p$ from equation 3.3.18.
2. Compute the attitude residual $\tilde{\mathbf{z}}_q$ from equation 3.3.20.
3. Compute Jacobians from equation 3.3.22.
4. Compute the EKF Innovation $\mathbf{S} = \mathbf{H} \cdot \mathbf{P} \cdot \mathbf{H}^T + \mathbf{R}_V$ with $\mathbf{R}_V = \mathbf{V} \cdot \mathbf{R} \cdot \mathbf{V}^T$ and \mathbf{R} is the measurement covariance matrix acquired from the sensor.
5. Compute the inverse Innovation matrix $\mathbf{Inv}(\mathbf{S}) = \mathbf{S}^{-1}$.
6. Compute the EKF gain using the inverse Innovation matrix $\mathbf{K}_{\text{gain}} = \mathbf{H}^T \cdot \mathbf{P} \cdot \mathbf{Inv}(\mathbf{S})$.
7. Compute the filter correction $\hat{\tilde{\mathbf{x}}} = \mathbf{K}_{\text{gain}} \cdot \tilde{\mathbf{z}}$
8. From 3.3.5 and ensuring unit length, correct the quaternions state with a small δq .
9. Correct the updated state variables in our state \mathbf{x} using $\hat{\tilde{\mathbf{x}}}$.
10. Finally update the state covariance:

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I}_{\text{dist}} - \mathbf{K}_{\text{gain}} \cdot \mathbf{H}) \cdot \mathbf{P}_{k+1|k} \cdot (\mathbf{I}_{\text{dist}} - \mathbf{K}_{\text{gain}} \cdot \mathbf{H})^T + \mathbf{K}_{\text{gain}} \cdot \mathbf{R}_V \cdot \mathbf{K}_{\text{gain}}^T \tag{3.3.23}$$

As already mentioned, the non-linear observability analysis shows that all states are observable including the inter-sensor calibration state p_i^j and the absolute yaw angle of the UAV and this result is only valid as long as there is an excitation in at least two linear directions. From intuition, this is obvious since the double-integration IMU accelerometers must align with the measured position of the measurement sensor, leading to observability of the yaw angle.

3.3.4 EKF Multi-Sensor Model

In previous sections, we discussed the EKF basic UAV sensor set-up consisting of a 6DoF and an IMU to facilitate vision-based navigation for UAVs. Since the IMU senses gravity but the camera pose suffers from 6DoF spatial drift, one might assume that our basic approach only recovers the UAV state up to a global yaw and global 3D position. For many tasks, neglecting the visual position and yaw drift, is reliable if an appropriate path planner is applied.

However, this basic IMU-visual sensor configuration, is sufficient for locally stable navigation. Thus we already have a full system running in real-time and on-board, able to estimate the UAV state, calibrate the inter-sensor states and estimate the

visual roll, pitch, and scale drift, and the IMU biases. Multi-sensor systems for navigation have three main requirements to the pose estimation. First, it should be scalable to the environment in question. Second, it should be extendable with additional (drift free or slowly drifting) sensors. Third, the algorithm should account for any misalignment of the sensors occurring during navigation.

All previous requirements are met by considering each sensor as a separate unit to estimate the UAV pose or part of it, and fusing each output in a statistically optimal way. Such type of system is called loosely - coupled. This allows us to employ these pose estimators as a *black-box*. The presented EKF implementation taking into consideration the sensor frame tackles the first requirement. The actual UAV position is defined by the IMU, expressed in the world frame, and the drift of the sensor pose estimate is already modeled as a rotational and translational offset between the sensor and the world reference frame. Further sensors are added and referred to the world frame.

In this thesis the main focus of a multi-sensor model, is the deployment of two different 6 DoF monocular visual sensors. Each sensor position is not in par with either UAV IMU or between them. Vision algorithms tend to drift due to the accumulation of small errors. While a slow position drift does not affect (local) navigation, the rotation is more problematic: A wrong estimate of attitude leads to a wrong alignment of the world z-axis and thus to partial integration of g . This drift is observable for the roll and pitch axes, as already mentioned, and therefore are compensated by adding the visual sensor reference frame \mathbf{p}_v^w and \mathbf{q}_v^w to the state, as we present in equation 3.3.3.

Before we proceed with the dual-sensor set-up, we will address the issue of multiple measurements for single on-board UAV visual sensor. In fact, the state presented in equation 3.3.3 is already taken this into consideration. But the EKF measurement model in equations 3.3.16 and 3.3.20 is valid only for the case that the position of the sensor is measured with respect to the origin, as it would be the case for an external e.g. motion capture system measuring the UAV state. For on-board systems, the measurement reference frame is moving from the point of view of the camera. We define the position measurement \mathbf{p}_s^v and the orientation \mathbf{q}_{-s}^v , and the new measurement model for position become:

$$\mathbf{z}_p = \mathbf{p}_s^v = (\mathbf{A}_I^s \cdot \mathbf{A}_I^w \cdot (\mathbf{p}_v^w - \mathbf{p}_I^w) - \mathbf{A}_I^s \cdot \mathbf{p}_I^s) \cdot \lambda + \mathbf{n}_p \quad (3.3.24)$$

Similarly, the orientation measurement \mathbf{z}_q is:

$$\mathbf{z}_q = \mathbf{q}_{-s}^v = \mathbf{q}_{-I}^{s*} \otimes \mathbf{q}_{-I}^{w*} \otimes \mathbf{q}_{-v}^w \otimes \delta q_n \quad (3.3.25)$$

Computation of the error quantities, Jacobians and corrections follow the methods presented in the previous section. For the case that we have an additional algorithm working with the same camera's images, for instance a tracking known marker(s) on a landing pad, we can simply add the measurement frame of that specific algorithm to the state vector. An additional camera calibration state is not necessary, since it was already included by adding the camera to the sensor setup. Essentially, given a sensor is observable, this leads to the conclusion that we need calibration states with respect to the IMU for each sensor, and measurement frame states for each algorithm, that provides a measurement. This setup is shown in Figure 3.3.3. We only need to find a valid transformation chain consisting of our desired world to IMU transformation, a calibration, a measurement and a measurement frame with respect to the world frame.

Besides using multiple measurements process or algorithms on board the UAV, additional sensors are added. In this thesis a dual visual 6 DoF pose estimation sensor (camera) is used together with the IMU. Additionally a pressure sensor is implemented and will be presented in the next section. As suggested in sections 3.3.1 and 3.3.2 our EKF implementation has a basic form based on the presented propagation model using the dynamics acquired by the IMU, and consisting of the basic state \mathbf{x} , equation 3.3.3. This needs to be augmented with measurements (update part) in 3.3.3. As a result, using additional auxiliary states \mathbf{x}_{aux} for measurement updates, the second sensor is added without affecting the basic module. The only limitation to be considered is that the auxiliary error state $\tilde{\mathbf{x}}_{aux}$ and the basic error state $\tilde{\mathbf{x}}$ must not depend on each other during propagation. The same applies for the error state noise vectors.

If this is fulfilled and as observability is met³, we can implement updates for the additional sensors and algorithms, as described above. To add another sensor/algorithm to the system, it is only necessary to define the initialization values of the corresponding states, the measurement Jacobians \mathbf{H}_p , \mathbf{H}_q , \mathbf{V}_p , \mathbf{V}_q and compute the residual from equation 3.3.18. The

³An important note to be considered on observability, states that once we have the global position and yaw of one of the measurement frames, the remaining ones become observable as well. In practice, this can be obtained by setting one of these frames of reference to a fixed position and yaw by a measurement with zero uncertainty. As an example, a fix position or yaw of a known marker that is attached to a landing pad can be used.

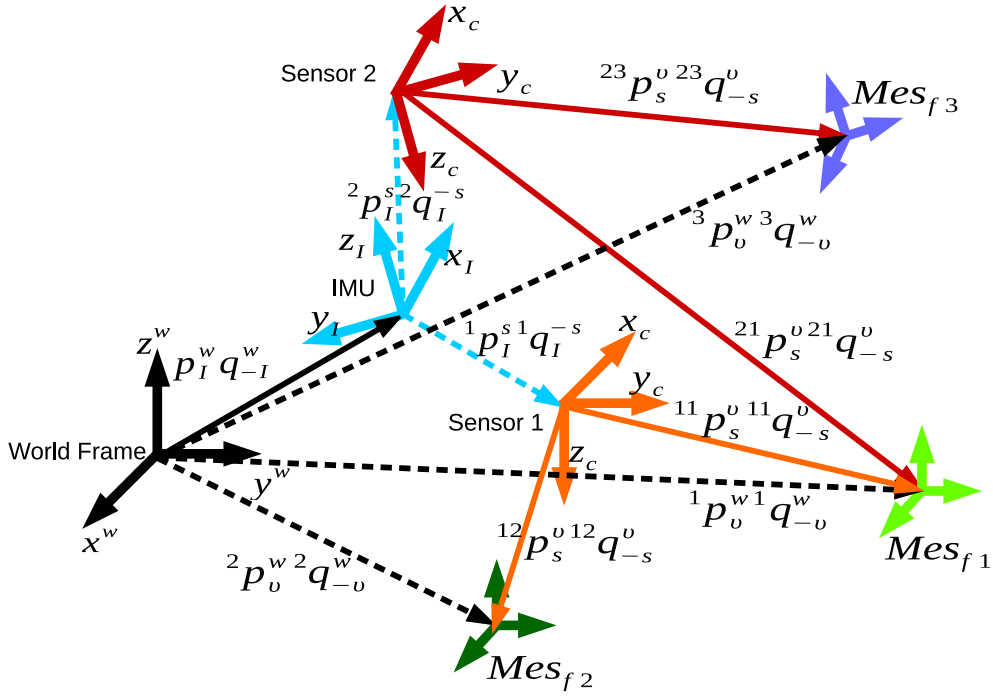


Figure 3.3.3: Full coordinate frame setup for multiple sensors, and with multiple measurements, e.g. two vision algorithms Mes_{f1} (bright green) and Mes_{f2} (dark green), having different reference frames, can use the images from the same sensor Sensor 1 (orange), which has a calibration with respect to the IMU p_I^s and q_I^{-s} , (cyan dash line). In contrast with system in figure 3.3.2, the measurements vectors point from the sensor to its measurement frame, since this frame is moving from the point of view of the sensor. With multi-sensor setup we can combine sensors or algorithms arbitrarily. For instance, a known visual feature, the Mes_{f1} (bright green) measurement reference frame, may be seen by both sensors.

augmented state for the dual visual sensor format and a UAV pressure sensor, yields a 47-elements state vector \mathbf{x}_{aux} (same procedure holds for the error state).

$$\mathbf{x}_{aux} = [p_I^{wT} \quad v_I^{wT} \quad q_I^{wT} \quad b_\omega^T \quad b_a^T \quad \lambda \quad p_I^c \quad q_I^c \quad p_v^w \quad q_v^w \quad 2\lambda \quad 2p_I^c \quad 2q_I^c \quad 2p_v^w \quad 2q_v^w \quad b_{p_{press}}] \quad (3.3.26)$$

with 2λ , $2p_I^c$, $2q_I^c$, $2p_v^w$, $2q_v^w$, for the second visual pose sensor and $b_{p_{press}}$ the bias for the pressure sensor. Similarly, we present only the additional differential equations to these in equation 3.3.4, (same procedure holds for the error error dynamics), for the state \mathbf{x}_{aux} :

$$\begin{aligned} \dot{2}\lambda &= 0 \\ \dot{2p_I^c} &= 0 \\ \dot{2q_I^c} &= 0 \\ \dot{2p_v^w} &= 0 \\ \dot{2q_v^w} &= 0 \\ \dot{b}_{p_{press}} &= 0 \end{aligned} \quad (3.3.27)$$

The proposed UAV pressure sensor is able to estimate the height. In order to achieve this, the bias $b_{p_{press}}$ need to be account. Setting the changes in pressures as follows:

$$\tilde{x}_{press} = \Delta b_{p_{press}} \quad (3.3.28)$$

For the pressure measurements the following model applies, with n_{press} as the measurements noise modeled with zero mean, and Gaussian white noise, p_{press} the measured altitude, \tilde{z}_{press} the error in z - position.

$$\tilde{z}_{press} = p_{press} - b_{p_{press}} + n_{press} \quad (3.3.29)$$

$$\tilde{z}_{press} = z_{press} - \hat{z}_{press}$$

The error \tilde{z}_{press} is linearized using the Jacobian H_{press} of the pressure measurement with:

$$\tilde{z}_{press} = H_{press}\tilde{x}_{press} + n\Delta b_{p_{press}} \quad (3.3.30)$$

3.3.5 False Pose Estimation

Unlike tightly coupled approaches, with our solution, we can treat the visual part, i.e. the visual pose estimation as a black box. Of course, one desires to detect failures and drifts of the black box to ensure ongoing functionality of the whole framework. The scale drift is handled by the scale estimate in real time. In section 3.3.4, we considered the rotation between the inertial world frame and the frame of the visual framework q_{-v}^w as constant during a filter step. At each filter step n we can measure this rotation as

$$q_{-v}^w = q_{-I}^{w*}(n) \otimes q_{-I}^{s*}(n) \otimes q_{-v}^s(n) \quad (3.3.31)$$

Since the drift is slow compared to the filter and measurement frequency, we can smooth a sequence of measurements of $q_{-v}^w(n)$. We found the median filter as the vision part is better modeled with non-zero mean outlier jumps. Therefore, the estimation of the rotation between inertial and visual frame, applying a window of size K and $i = n - K$ is:

$$\hat{q}_{-v}^w = F_{median}(q_{-v}^w)(i) \quad (3.3.32)$$

Due to the fact that the drift is slow, we can identify abrupt jumps in the measured orientation q_{-v}^w with respect to the smoothed estimate \hat{q}_{-v}^w as failures of the visual pose estimation. We implement this in the EKF filter, where as soon as a measurement q_{-v}^w lies outside the $2.5 \cdot \sigma$ error bounds of the past N estimates. When a failure period occurs neither q_{-v}^w nor the biases and scale are updated since the measured data is corrupted. We update the IMU orientation, the velocity and position to bound the error during long dropouts. In addition, gyroscope integration is fairly robust also over longer periods of time, in contrast low cost accelerometers suffer from large drifts and should not be integrated over longer periods of time without updates. In contrast, gyroscope integration is fairly robust also over longer periods of time.

Furthermore, we increase the visual measurement noise for the position much less than for the attitude. The increasing of the measurement noise during failure sequences bounds the filter error yet trusts the simple integration to a large extent. Analyzing the rotation for wrong measurements, this is less sensitive to loop closure jumps. Usually these jumps affect greatly the position but less the attitude.

3.3.6 EKF Design Model

The multi-sensor setup described in section 3.3.4 requires a modular software design, capable of adapting to different and multiple sensors. Software implementation consists of a basic module, taking care of IMU sensor handling, as well as performing state prediction and covariance propagation. In case of different and multiple measurement modules, a scheduler measurement manager is added, which is responsible of the initialization and handling the interactions between the sensors and their corresponding states. Whenever a new IMU reading becomes available, the scheduler creates a measurement state-object consisting of the measurements Jacobian, measurements covariance R^3 , the residual and a time-stamp of the measurement. These computations are the same for all measurements, just the dimensions of their matrices and vectors have to be compatible.

These objects are passed to the basic module, which computes the necessary update steps for the whole state, based on the measurement data objects. To apply this on a working system operating in indoor and outdoor scenarios, a splitting of the

EKF tasks and distributing them among the available hardware, according to the computation capabilities and real-time constraints, is crucial. This splitting not only optimizes for high frame-rates and low latency, providing real time estimates to the controller of the UAV, but also relaxes some real-time constraints for the low-end on-board computer. Distributing task is not always optimal in real-time systems as new scenarios and conditions are becoming available. An efficient strategy is used based on the critical importance of each task. Lower importance tasks are handed based on the available system resources, task completion time, task periodicity and task resources demand.

State prediction, as the most time-critical part for controlling the UAV, should be executed at high rates with the least possible delay, to allow for fast responses to disturbances, or to enable dynamic flights. Since IMU data is available with almost no delay at a rate of 1 kHz at micro-controller of the UAV, control part is implemented there. This leaves enough time for the more complex parts to be computed on the on-board computer with a non-real-time operating system. The UAV micro-controller sends its state predictions to the on board computer over a high speed serial link, while the on board computer sends back state corrections every time a measurement update arrives. Crucial for this to work is accurate time synchronization between UAV micro-controller and on board computer. This is implemented in a lightweight NTP-like approach as is described in [31]. The current state, predicted by the micro-controller, is exchanged only at a rate of 100 Hz. This means that we predict the covariance at 100 Hz while the state is predicted at 1 kHz on the UAV micro-controller. This is due to bandwidth limitations of the data link between micro-controller and the on-board computer, and due to the relatively high computation cost of propagating a $N \times N$ covariance matrix.

In addition, an efficient method to handle the measurement delays is crucial for robust and accurate state estimation. The measurement update cannot be performed when other processing is not finished. Instead it has to be aligned in time with the state prediction. Keeping a state-buffer of the past states is a solution to the delay, it facilitates the application of the obtained measurements at the exact time in the past they were taken. After performing the update step, the corrected state in the past is propagated again to the present time, where the state on the micro-controller is then corrected accordingly. In cases where no state is available at the measurement time, we interpolate the respective proprioceptive measurements to get the best available linearization point.

In a similar way we can handle measurement delays for the multi-sensor case. In multi sensor measurements fusion, it is typical that some sensor feeds might be missed for some time or become unavailable during the mission. As before, the closest state (in the past) is retrieved from the state-buffer, which is updated using the information from the visual-SLAM system. In addition, we have to keep a measurement buffer of measurement updates that have been performed. Subsequently, the measurement buffer is queried recursively for later measurements, between which both the state and covariance are predicted accordingly. Delayed sensor measurements are applied by querying the closest state in the buffer, and any subsequent measurements are re-applied to the updated state. This time we need to re-apply all EKF steps from the time of the measurement to the present time, using all measurements and system inputs that have occurred in that period. This is due to the different time that each sensor measurements are applied to the EKF model. As a result, when a delayed update measurement z_d is applied after a fast one z_f , we have to apply the full EKF procedure between the z_d and z_f time implementation again, requiring the predictions and updates occurring later to be replayed.

Formulating the visual update as already mentioned has draw-backs, when using absolute measurements from a visual-odometry (or SIAM performing systems) sensor framework, the latest estimate of the sensor scale is used to scale the whole odometry path, not taking into account intermediate changes in scale. In addition to that visual odometry- SLAM systems provide a pose measurement which denotes a relative measurement between time-instants. This is handled using these relative measurements between time-instants in EKF-framework. In order to account for relative measurements, authors in [23], propose to add a stochastic clone of the state for each sensor providing relative measurements as well as the respective errors for the landmarks used to compute the relative measurements. Since we want to use the sensors in a loosely-coupled manner abstracting the internal algorithms (e.g. V-SLAM) we don't include the measurement errors for landmarks relating both states in our state vector. A different approach, partially used in this thesis, is based on the framework presented in [60]. The modularity of this framework allows seamless handling of sensor signals while employing a state buffering scheme augmented with Iterated EKF (IEKF) updates to allow for efficient re-linearization of the prediction to get near optimal linearization points for relative state updates. To keep the computational complexity low and because in general not all measurements in a multi-sensor setup denote relative quantities this approach does not in general clone every state. We do not provide an in depth study of the implementation of relative measurements here, as this would go beyond the scope of this work.

Chapter 4

Horizon Line Estimation

To operate an Unmanned Air Vehicle (UAV) in a realistic environment, it is crucial to monitor and control parameters and environment features associated with the system state. Information provided by vision-based sensors is of utmost importance for this task, as evidenced in a number of studies [67], [68], [69], [70]. Furthermore horizon line can be used as an additional, or a back up visual cue for autonomous systems navigation, localization and visual SLAM algorithms and path planning, [71], where the conventional localization techniques would not work, such as extraterrestrial spaces and regions with no or very limited GPS and generally lack of good structural features and landmarks, [4].

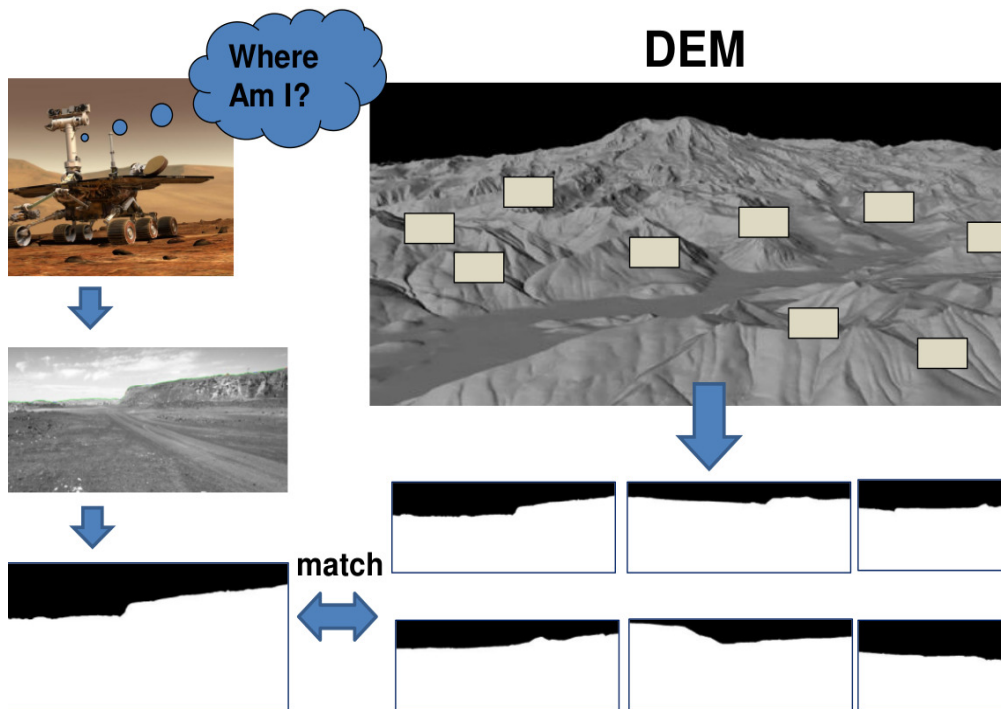


Figure 4.0.1: Rover localization, [4].

Fundamentally, flight stability and control requires measurement of the UAV's angular orientation. While for larger aircraft this is typically estimated through the integration of the aircraft's angular rates or accelerations, a vision-based system can directly measure the aircraft's orientation with respect to the ground. More specifically, of particular interest is the detection of the so-called Horizon Line (HL), that in turn gives rise to two critical parameters regarding UAV's angular orientation, namely roll angle ϕ_{roll} and pitch angle θ_{pitch} . The latter are directly involved in tasks, i.e autonomous systems stability, smooth UAV navigation and obstacle avoidance [68], [54], [72], [1].

HL detection algorithms usually rely on plausible assumptions, such as HL forms a linear boundary, lies in the upper half of the image, sky and non-sky regions are equally probable, and the higher brightness and prominent texture of the upper part of the image (sky region) compared to the lower part (non-sky region). Methodologically, machine learning approaches, i.e. the decision trees approach proposed by G.C.H.E. de Croon et al., [73] for sky segmentation, and particularly deep learning [74], have led to state-of-the-art performance, at the cost of assigning significant computational resources during the training phase. Moreover, such methods do not scale smoothly in cases of images that contain unlearned features and/or patterns. Zhai et al. [74], employed deep Convolutional Neural Networks to derive multiple vanishing points in the image, which are used in turn to optimally compute HL. Alternative probabilistic approaches that rely on real-time feature inferencing offer robustness in the presence of irregular features, but significantly increase execution times.

Regardless of the employed methodology, most HL detection approaches fall into two main categories: (a) edge detection approaches, and (b) image segmentation into sky and non-sky regions. Zafarifar et al. [75], were among the first to formulate a robust edge detection based scheme to identify HL. By employing the Hough transform prominent lines are extracted and the *strongest* one is declared as HL. This method was found to be more accurate with respect to the angular error in [76]. This work compared the Hough HL segmentation with four algorithms designed to find a horizontal line separating sea and sky in marine images in real-life conditions. One of the selected algorithms was based on regional covariances in luminance images, the second one used maximal local edge detection and the least-squares optimization method, the third one employed median filtering in small neighborhoods and linear regression and the fourth one was based on regional edge magnitudes and the least-squares method. The highest accuracy with respect to the position of the line was achieved by the regional covariance method but only with a half pixel mean deviation than Hough HL approach. Lie et al. [70], compute an image edge on a non-refined edge map which is subsequently transformed to a multi-stage graph. Dynamic Programming (DP) is then utilized to detect HL as the shortest path in the graph. This approach is based on the assumption that there exists a consistent edge boundary segmenting the sky and non-sky regions. This assumption, however, is not always true due to various environmental factors (e.g., clouds) and edge gaps. Even with the proposed gap filling strategy in [70], all edge gaps cannot not be filled, [77]. Moreover, DP methods assume that consistent edges appearing in the upper half of the image belong to the horizon line. To enforce this assumption, they induce biases towards shortest paths in this region. This assumption is not always valid as there might be strong edges in this region due to environmental factors (e.g., clouds). By enforcing this constraint, consistent edge segments due to clouds could become part of the extracted horizon line since there is no way to discriminate between cloud edges and horizon edges. An alternative, edge-based DP approach was proposed by [77], where a classifier is used to refine the edge map by removing non-horizon edges. The refined edge map is then used to form a multi-stage graph where dynamic programming is applied to extract the horizon line.

A number of approaches in the second category focus on extracting the boundary between sky and non-sky image areas. Accordingly, segmentation into sky and non-sky regions is attempted based on various descriptors such as texture, color, SIFT and Histogram of Gradients. Ahamd et al. [77] proposed an edge-less machine learning approach based on dynamic programming on the resultant dense classification map rather than on the edge map, for classifying each image pixel based on the above descriptors. Edge-less methods suffer more from miss-classifications compared to edge-based methods. This is because every pixel is classified in the case of edge-less methods while only edge pixels are classified in the case of edge-based methods. Even if DP techniques mentioned here are able to compute a non-linear sky segmentation, this is influenced by the sky-ground region separation. This can lead, in cases of partial obscured HL by obstacles, to false detect obstacles points as HL points.

In [5], Omar Oreifej et al. detect HL based on maximum a posteriori estimation applied on multiple candidate horizon lines. The authors in [68] utilize optical flow to robustly extract edges prior to performing the Hough transformation. Moreover, they compute ϕ_{roll} and θ_{pitch} based on the detected HL and under the assumption of infinite horizon distance. Boroujeni et al. [78], employed k -means and intensity-based clustering to segment the image in the regions of interest. Todorovic and Nechyba [67], relied on multi-resolution linear discriminant analysis for image segmentation and, additionally, derive ϕ_{roll} and θ_{pitch} for UAV navigation.

The goal of the mentioned segmentation is almost always to obtain an estimate of the horizon line, which conveys information on the pitch and roll of the air vehicle. As a consequence, most studies described here, [5], [67] and [68] contain an underlying assumption that the MAV is flying quite high in the sky, with few to no obstacles protruding from the horizon line. The goal of segmentation in this thesis is different, namely, it is to detect HL, when the camera is not high in the sky

and HL is partially obscured by obstacles.

Therefore, in this chapter, we propose a novel HL detector solution, that employs Particle Swarm Optimization (PSO) [79], to assess candidate horizon lines. The latter are derived via Canny edge detection followed by the application of the Hough transformation. The main novelty of the current work lies in the optimization step where adequate image descriptors are utilized to evaluate alternative HL solutions. This is accomplished via the PSO's objective function which is formulated as a variant of the Bag of Words (BOW) encoding of the image content. The image descriptors used in the latter are $L*a*b$ color features, texture features and PHOW-SIFT features. Accordingly, rich image-based information is considered to robustly compute HL, whereas the PSO scheme facilitates efficient computation times.

The proposed HL detection framework consists of two main modules: (a) computation of candidate HLs, and (b) HL derivation. The first module comprises edge detection followed by Hough transform, whereas the latter employs PSO, and an optimized HL solution is acquired. A detailed technical presentation is in order. The chapter is organized as follows. Section 4.1 establishes the computation of candidate HLs for the proposed framework. In Section 4.2 the optimized HL detection approach is presented. The Bag of Word concept is introduced in 4.3. We conclude the chapter in section 4.4, by describing on how the extracting HL may be used to derive UAV roll and pitch angles.

4.1 Computation of Candidate HLs

A block diagram illustrating the processing steps in the current module is depicted in Figure 4.1.1. Initially, the input image is converted to grayscale and, subsequently, a Canny edge detector [80] is applied, with edge thresholds determined from the Otsu method [81]. Edges are identified as local maxima in the direction of the intensity gradient, where in our case the corresponding gradient is calculated using the derivative of a Gaussian filter. The edges points give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top, thus a thin line is formed as output, a process known as non-maximal suppression. The ridge pixels are then thresholded by a hysteresis algorithm, which is based on using the two Otsu thresholds O_{t_1} , O_{t_2} . Ridge pixels with values greater than O_{t_2} , are said to be the dominant edge pixels, and a strong indications for edge. Ridge pixels with values between O_{t_1} and O_{t_2} threshold, are the weak indications for edge pixels, thus an edge linking is performed by incorporating into edges only these weak pixels, that are 8-connected with a strong edge pixel.

The computed edge map image is used as input to a line detector based on the Hough transform as in [82], where the image space (x,y) is transformed to the (ρ,θ) parametric space according to:

$$\rho = x\cos(\theta) + y\sin(\theta) \quad (4.1.1)$$

where ρ stands for the perpendicular distance of the line from the origin and θ for the angle subtended by the line with respect to the x -axis. Consequently, a line in the image space corresponds to a point in Hough space and vice versa. Next, Hough accumulator cells are populated with one vote each time a non-background point in the image is detected. The bin with the highest number of votes represents a Hough peak, or in other words a potential line in the input image, figure 4.1.2.

Accordingly, we formulate the set of candidate lines, \mathbb{S}_{HL} , as prominent points in the Hough parametric space. More specifically, three subsets of points are constructed, \mathbb{S}_{SP} , $\mathbb{S}_{P_{sel}}$, and \mathbb{S}_R , with their union representing \mathbb{S}_{HL} . \mathbb{S}_{SP} denotes the most probable Hough peak points P_H , according to their cumulative distribution (CDF), \mathbb{S}_{RP} is similarly the set of selection peaks from all the peaks, excluding P_H points. \mathbb{S}_{RP} peaks are chosen, similarly based on the peaks CDF, resulting to a subset, which points is represented with random peaks from all the peaks distribution. The \mathbb{S}_R subset consists of randomly selected points from the remaining Hough space $(H - \{P_H\})$, where α is a threshold that determines the sizes of the first set:

$$\begin{aligned} \mathbb{S}_{SP} &: \forall CDF(P_H(\rho, \theta)) \geq \max(CDF(P_H(\rho, \theta))) - \alpha \\ \mathbb{S}_{P_{sel}} &: \text{select}(CDF(P_H(\rho, \theta))) \\ \mathbb{S}_R &: \text{rand}(H(\rho, \theta)) \end{aligned}$$

The rationale for formulating candidate HLs as the union of the above three subsets \mathbb{S}_{SP} , $\mathbb{S}_{P_{sel}}$ and \mathbb{S}_R lies in the nature of

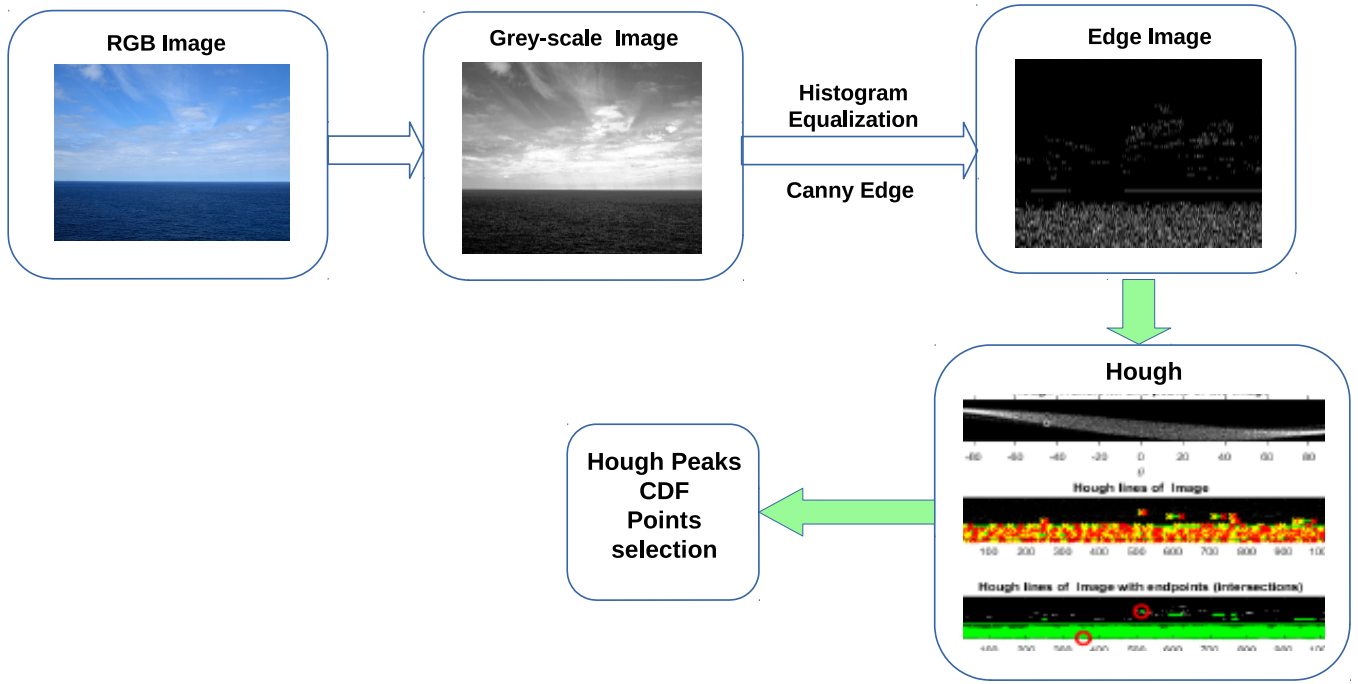


Figure 4.1.1: Computation of candidate Horizon Lines. The input is an *RGB* image. A Canny Edge detector is applied, with a Hough transform following. The Output is the set of candidate HLs. HLs subsets, particularly \mathbb{S}_{SP} , $\mathbb{S}_{P_{sel}}$ and \mathbb{S}_{RP} , are produced based on P_H CDF distribution and Hough space points.

the adopted optimization method, namely PSO. In particular, \mathbb{S}_{SP} feeds as candidate HL solutions to the PSO lines with high image contrast and hence high probability of being the sought HL. Given that in many cases the actual HL might be a weak image line or having mid probability, according to P_H CDF distribution, $\mathbb{S}_{P_{sel}}$ is utilized. In addition, the use of P_H only Hough points, in the \mathbb{S}_{SP} and $\mathbb{S}_{P_{sel}}$ subsets, are explicitly employed to expedite PSO convergence and cater for fast execution times. Finally, in cases that, the HL is defined by a non-peak point, the subset of the \mathbb{S}_R is used to provide randomly selected Hough points to PSO and thus facilitate successful convergence to the correct solution, so that the swarm can effectively explore the parameter space, without getting stuck to local minima.

4.2 HL Derivation

A schematic representation of the second module is shown in Figure 4.1.3 which actually comprises PSO. The latter is a stochastic evolutionary algorithm, since it incorporates concepts, such as populations, generations and rules of evolution for the particles. The population is essentially a set of points in the parameter space of the objective function to be optimized. The particles evolve in parts or generations. After some iterations and evolutions the PSO leads to a global behavior, in terms of a globally optimal or near optimal solution, discovered in a defined multidimensional search space and regarding a defined global function, which has to be minimized.

Therefore, PSO particles, which correspond to a set of proposed solutions, move around a specific bounded solution-space, searching and evolving to converge to the best solution which optimizes an objective function. Accordingly, the HL returned by PSO is an optimized solution, with respect to the image-based criteria that are expressed by the objective function. All particles are initialized from \mathbb{S}_{HL} and the PSO variant developed by Oikonomidis et al. [83] is utilized, which is based on the canonical PSO, (*C-PSO*). The main assets of *C-PSO* are: (a) depends on only a few parameters, (b) the computation of the objective function derivatives are not required and (c) a low number of evaluations of the objective function is used for *C-PSO* convergence. In PSO, each particle stores its current position in a vector x_k and its current

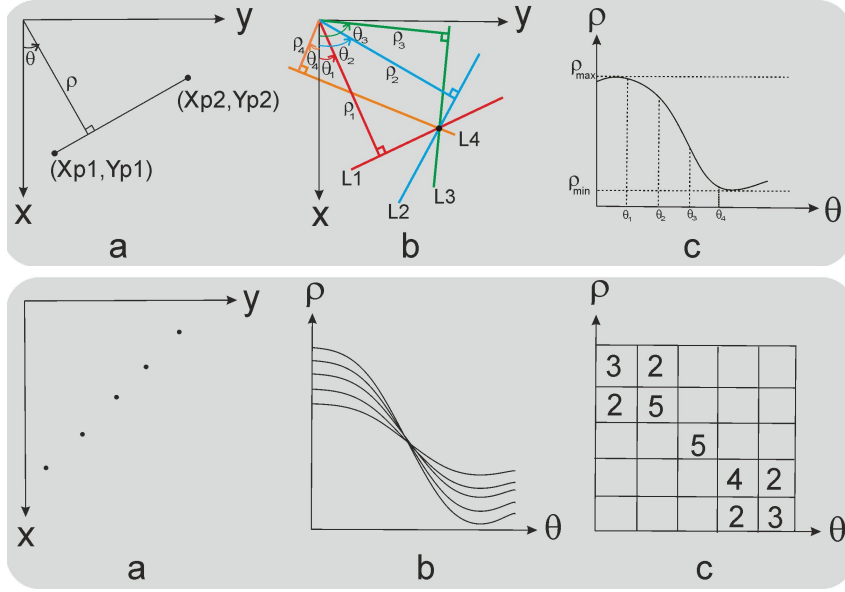


Figure 4.1.2: Hough Transform with polar variables ρ as the radius and θ as the angle. The (x,y) are the 2D space coordinates. Basic Principle of Hough. Upper image: (a) shows representation on equation of straight line, (b) illustrates intersection of many lines to a point, (c) Transformation of point in image space to polar space. Bottom image: Working of Hough as feature extractor. (a) Shows a set of points in image space, (b) illustrates convergence of points in image space to sinusoidal waves in polar space, (c) represents the accumulator space. The limits of the Hough transform are $[-D, D]$ for ρ , where D is the diagonal of the input image and θ lies in $[-89^\circ, 90^\circ]$.

velocity in a vector u_k , where k is the current PSO generation. The position at which the particle achieved the best score of the objective function, in the current generation k , is stored in vector P_k (local optimum). The best position encountered across all particles of the swarm, in the current generation k , is declared as the global optimum in vector G_k . Then every particle is acknowledge of the global optimum and taking into consideration its own local optimum, and updates its velocity and position vector in every generation. The update equations for every generation t correction in each particle velocity and position are:

$$\begin{aligned}
 u_t &= K(u_{t-1} + c_1 r_1 (P_k - x_{t-1}) + c_2 r_2 (G_k - x_{t-1})) \\
 u_t &= u_{t-1} + u_t
 \end{aligned}
 \tag{4.2.1}$$

where K is a constant constriction factor, c_1 is the cognitive component, c_2 is the social component and r_1 and r_2 are random samples of a uniform distribution in range $[0 : 1]$, [83]. For each dimension in search space, a boundary constraint is defined. In first iteration each particle are spread all over the search space with a random function and the velocity is set to be zero. If during an evolution of a particle, a particle is forced to move outside the boundaries a handling policy is required.

Given that each PSO particle represents an image line, it segments the image into two regions. For each of the two regions, three feature-histograms are formed based on SIFT, Color, and Texture features. To facilitate computational efficiency, feature-histograms are not computed for each possible image region but are readily derived from corresponding histograms precomputed for the entire image. The latter are computed only once and give rise to histograms for specific image regions by means of the Bag of Words (BOW) scheme, outlined in the next section for the sake of clarity of presentation. The above computed histograms for each image region are used to formulate the objective function F_O of the PSO.

Firstly, a histograms comparison was implemented by computing the union (Un) and intersection (Int) between the two image histograms. Then the F_{Oa} is found according to equation 4.2.2.

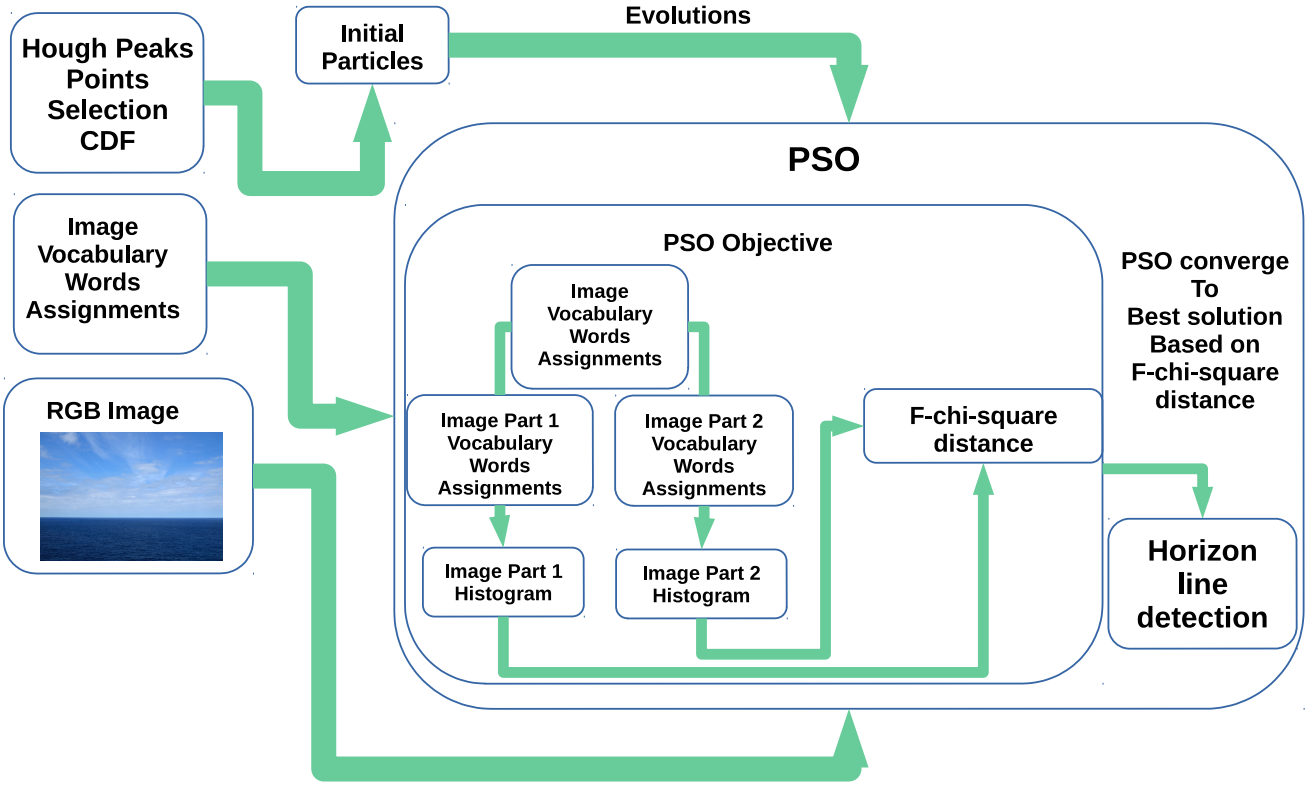


Figure 4.1.3: Schematic description of HL derivation. PSO objective is based on χ -square distance, computed by the two image parts.

$$F_{Oa} = \min \prod_{n=1}^3 \left(\frac{Un_n}{Int_n} \right) \quad (4.2.2)$$

where the three Un_n and Int_n refer to each descriptor used (Sift, color, texture) and the F_{Oa} is calculated on each PSO iteration for every particle and represents a score. However this objective was found not robust enough, e.g especially in figures with similar texture and color features. Additionally, in some image cases, the F_{Oa} score was the same for different HL. For that, the corresponding histograms across the two regions are also compared in terms of the χ^2 distance, χ^2 . F_{Ob} is calculated as the product of the three χ^2 distances:

$$F_{Ob} = \min \prod_{n=1}^3 (1 - \chi_n^2) \quad (4.2.3)$$

where the three χ_n^2 distances refer to each descriptor used (SIFT, color, texture). Similarly the F_{Ob} is calculated on each PSO iteration for every particle. F_{Ob} was found to be more robust, than F_{Oa} , yet all features had the same influence, even in the images where some features are less accurate, i.e. low texture images, or only a few Sift features were detected. To counter this, weights are included and the F_O solution was obtained:

$$F_O = \min \sum_{n=1}^3 (1 - w_n \cdot \chi_n^2) \quad (4.2.4)$$

where the three w_n refer to each descriptor weight used (SIFT, color, texture). Equivalently, F_O is calculated on each PSO iteration for every particle and represents a score. The weights in this thesis were computed with empiric methods, mainly using an image dataset. More rigorous approaches are build on non-linear principal components analysis (PCA) and Machine- Deep learning techniques, a complete description of which, is beyond the scope of this work. Essentially, each particle ρ , θ corresponds to a line that separates the image into two regions. χ_n^2 measures the distance between the BOW representations of the two regions. Thus, the $1 - \chi_n^2$ measures the similarity of these regions. Given the above, the objective function F_O measures the similarity of the two regions with respect to all three descriptors. Overall, the PSO-based minimization of the F_0 function results in the optimal Hough point (ρ^*, θ^*) that separates the image into two maximally dissimilar regions.

4.3 Bag of Words - BOW

Bag of Words (BOW) is a method for image recognition and classification. Similar to terms in a text document, an image has local interest features or key-points depended as salient image patches (small regions) that contain rich local information of the image, figure 4.3.1.

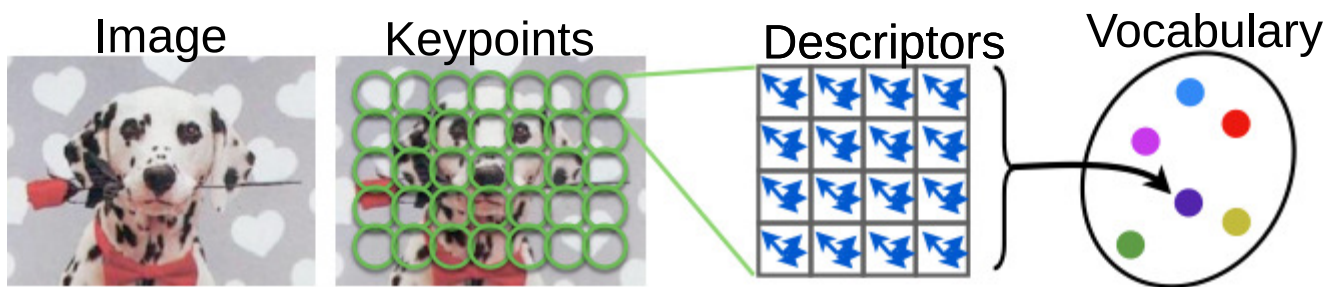


Figure 4.3.1: A key-points BOW approach. Denoted by small circles (blobs), key-points are usually around the corners and edges in image objects, such as the edges of the map and around people’s faces. A SIFT descriptor can automatically detect regions/points of interest and compute local descriptors over those regions/points. Then, the descriptors are quantized into visual words to form the visual vocabulary.

Images can be represented by sets of key-points of feature descriptors, but the sets may vary in cardinality and lack of meaningful ordering. This creates difficulties for learning methods (e.g. classifiers) that require feature vectors of fixed dimensions. To solve this problem, in BOW approach the vector quantization (VQ) technique is used, which clusters the key-points descriptors, using the K-means algorithm, by the index of the cluster to which it belongs.

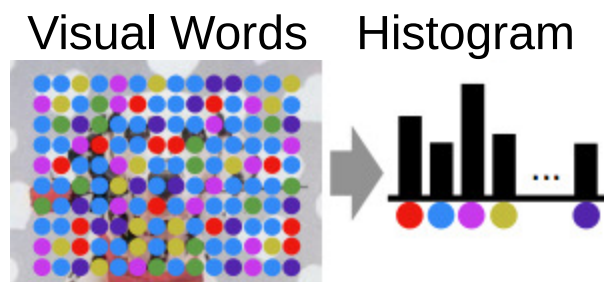


Figure 4.3.2: BOW visual words. The occurrences of each specific visual word in the image are computed, constructing the BOW histogram, based on visual word frequencies.

We conceive each cluster as a visual word that represents a specific local pattern shared by the features in that cluster. Thus, the clustering process generates a visual-word vocabulary, figure 4.3.1, describing local patterns in images. The

number of clusters determines the size of the vocabulary, and mapping the key points to visual words, we can represent each image as a bag of visual words, illustrated in figure 4.3.2. The bag-of-words technique uses a previously built visual vocabulary to transform an image to a numerical vector that is subsequently employed to look up among the images stored in a database. Since the BOW presented in [84] is hierarchical, the vocabulary structure becomes a tree.

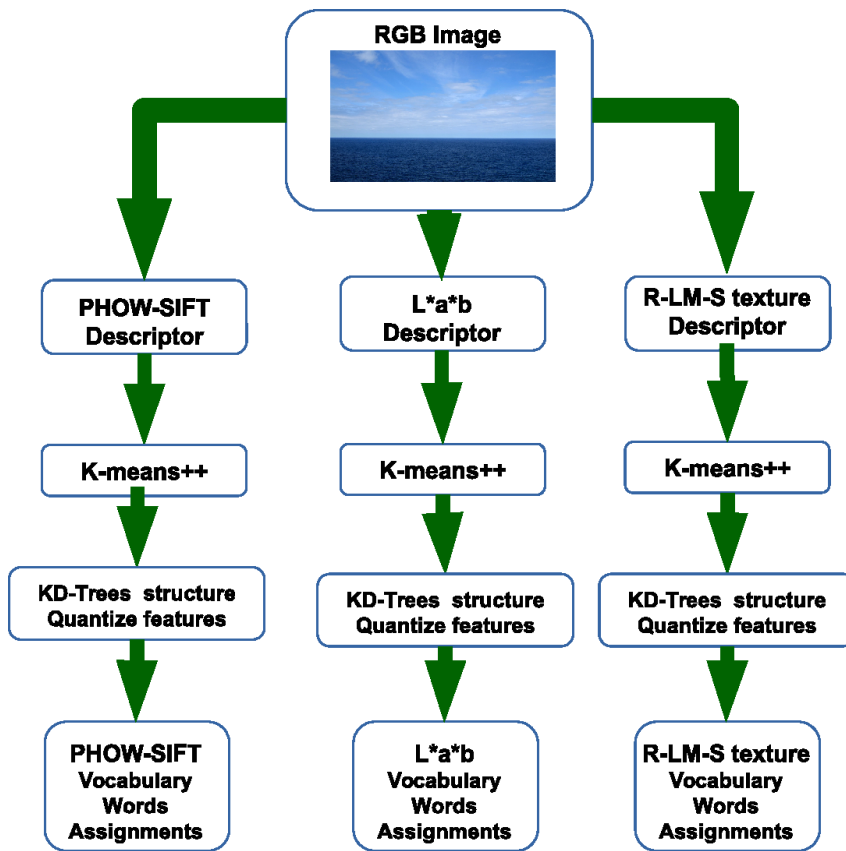


Figure 4.3.3: BOW formulation. Color- L^*a^*b , texture and SIFT-Phow descriptors, vocabulary and words are extracted from input image.

In our framework BOW, Figure 4.3.3, is employed for the sake of computational efficiency and is used for obtaining feature-histograms for any image region from the corresponding histogram of the entire image. As mentioned above, three such histograms are used in our framework based on appropriate local descriptors namely, SIFT features, L^*a^*b color features, and texture features.

SIFT is a blob detector, i.e. it doesn't have high response at corners in the image but in blob-like structures. In addition to the detection, the SIFT method also proposes a local descriptor based on the histogram of orientations of the feature neighborhood, providing detected features with a highly distinctive signature that allows matching them with other views of the same feature. The descriptor, besides the scale and orientation invariance, presents partial invariance to changes in illumination and points of view. For years, SIFT has been the most powerful of the feature detectors and descriptors due to its robustness and performance when matching features that undergo different transformations. However, its computational burden is significant, sometimes rendering it useless when dealing with real-time applications. Although some SIFT-based methods have been reported dealing with this issue [85], their performance is usually affected. Furthermore, SIFT descriptors suffer from the low features detection in the two image parts, particularly in low resolution compressed images. Therefore, in this thesis we substitute the SIFT descriptors in our framework with a dense SIFT approach, the PHOW descriptors, [86], which can quickly compute descriptors for densely sampled keypoints with identical size and orientation, with the main advantage of using a dense SIFT over SIFT is faster computation time. Additionally, the PHOW descriptors, based on SIFT features applied at several resolutions, which are robustly computed in the presence of brightness variations.

The reasons of selecting $L*a*b$ color features than the typical camera $R-G-B$ system are: (a) the $L*a*b$ system is device (camera) independent in contrary with the other formats, this makes the proposed algorithm able to run in different visual sensor and providing consistence results, (b) the $L*a*b$ are more clearly separate gray-scale information (entirely represented as L^*) from color information (represented using a^* and b^*), this feature is allowing us to chose of using only color information, forming an color descriptor unaffected by light intensity variations and weather changes. Furthermore, $L*a*b$ was designed so the Euclidean distance in $L*a*b$ -space corresponds reasonably well with the perceived differences between the colors, resulting to a perceptually uniform color representation. As a corollary, L^* values are linearly related to the human perception of brightness.

The latter descriptors, the texture, are extracted via the combination of three different filter banks: Leung-Malik (LM) [87], Schmid (S) [88], and the Root Filter Set (R) [89]. The Leung-Malik (LM) Filter Bank set is a multi-scale, multi-orientation filter bank with 48 filters. The Schmid (S) Filter Bank consists of 13 rotationally invariant filters and the Maximum Response (MR) Filter Banks, each of the reduced MR sets is derived from a common Root Filter Set (RFS) which consists of 38 filters and is very similar to LM.

Subsequently, the vocabulary of visual words for each different image feature descriptor is constructed, a process often mentioned as BOW encoding. In our case, the visual vocabulary is computed from a number of words (k -means centers) by running the k -means algorithm on features. The optimal use of k -means is based on the number of clusters k , since for different images, the number k of clusters may vary. More specifically, for optimal selection of k , the Silhouette method [90] is employed. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette values ranges from -1 to 1 , where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. In the case that, most of the points have a high value, the clustering configuration is the appropriate, else if many points have a low or negative value, then the clustering configuration may have too many or too few clusters. The silhouette can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance.

Alongside the number of clusters selection, an initialization procedure for k -means is required. The simplest method for k -means initialization, e.g. selection of the k clusters centers are performed by a sample of k points from the input data, which are chosen randomly. This method is not robust enough, especially in cases of complex, in k -means space, data points representation. Accordingly, the k -means centers are initialized as a greedy pick of k data points that are maximally different according to the `kmeans++` approach in [91].

The overall BOW process is significantly accelerated by using randomized KD-Trees data structures (the FLANN library [92] is used in our implementation). A KD-Tree is a hierarchical structure built by partitioning the data recursively along the dimension of maximum variance. At each iteration the variance of each column is computed and the data is split into two parts on the column with maximum variance. The splitting threshold can be selected to be the mean or the median. KD-Trees structure also, enables fast medium and large scale nearest neighbor (NN) queries among high dimensional data points such as those produced by Sift and Phow. The KD-Tree query uses a best-bin first search heuristic, a branch-and-bound technique that maintains an estimate of the smallest distance from the query point to any of the data points down all of the open paths and supports two important operations: approximate NN search and k -NN search.

Finally as we already mentioned, BOW quantizes features using the vocabulary for finding the nearest k -means centers, based on the Euclidean distance metric for each descriptor, thus forming the visual words. The described process for BOW formulation is presented in Figure 4.3.3.

4.4 Derivation of Roll and Pitch Angles

Having computed HL, the UAV's roll and pitch angles are readily available from a sequence of images acquired from the UAV camera as in [67].

The roll angle ϕ_{roll} can be directly determined from the HL since it corresponds to the angle formed by the slope of the HL (see Figure 4.4.1). Since it is orthogonal to the camera's rotation axis, the HL rotates exactly the same as the roll angle of the UAV camera and is invariant to all other motions, such as UAV translation, pitch and yaw rotations. This is true because under these motions no image motion is induced for the points at infinity and the HL slope remains constant [68]. Accordingly, in this work we first compute the horizon slope $m_{Horizon}$, where θ^* is the Hough parametric space estimate

angle (equation 4.1.1) as:

$$m_{Horizon} = -\frac{\cos(\theta^*)}{\sin(\theta^*)} \quad (4.4.1)$$

Then, ϕ_{roll} can be derived as:

$$\phi_{roll} = \tan^{-1}(m_{Horizon}) \quad (4.4.2)$$

On the other hand, the relation between the pitch angle θ_{pitch} and the horizon line is less straightforward. Previous works such as [93], [94], [95], [67] estimate the pitch percentage which is correlated with the actual pitch. However, for navigation concepts, we need the exact pitch angle of the UAV. Hence, we extend the previous work and find the pitch using the intrinsic parameters f_u, f_v, c_u, c_v of the camera and applying relative projection equations from sections 2.1 and 2.2. Let us refer to the point of intersection of the HL and the principal axis of a level camera as P , and its image as p . If the camera is initially leveled with the ground (i.e. $\theta_{pitch} = 0, \phi_{roll} = 0$), then P will always project at the principal point p_0 as shown in figure 4.4.1. Here there is no need of taking in consideration the height h of the UAV, as P previous projection assumption is still hold. This can also be seen from the perspective projection equation. Hence, when the UAV undergoes pitch movement, the imaged point p moves vertically in the image plane. Therefore, the vertical distance between the horizon line and the principal point only depends on the pitch angle as illustrated in figure 4.4.1.

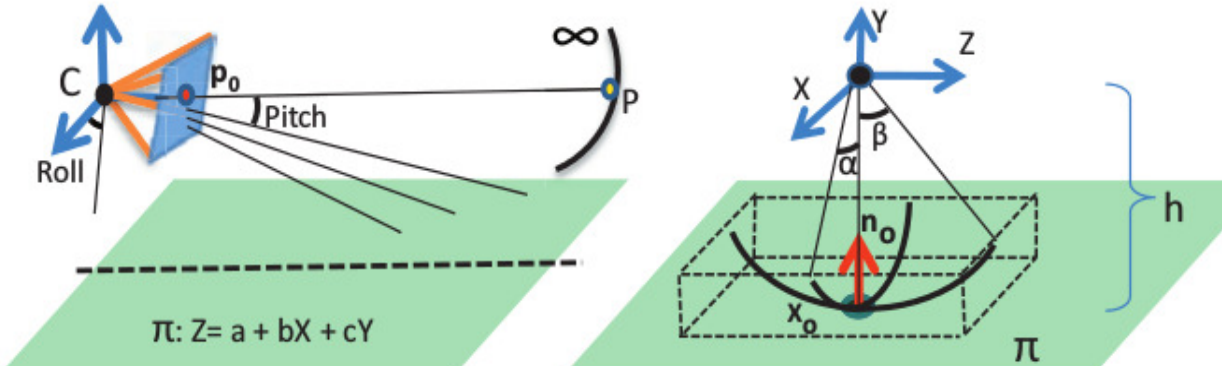


Figure 4.4.1: The point at infinity in the direction of the principal axis P always projects on the principal point p_0 if $\theta_{pitch} = 0$ holds. Additionally, rotation of the unit normal n_0 around the coordinate system is shown. The ground equation at any instance can be described by rotating the canonical unit normal $n_0 = (0, 1, 0)^T$ around the camera coordinate system α (ϕ_{roll}) and β (θ_{pitch}), where $Z = a + bX + cY$ is the equation of the ground plane in the camera coordinate system, where $a = -h/(\sin(\theta_{pitch}) \cos(\phi_{roll}))$, $b = \tan(\phi_{roll})/\sin(\theta_{pitch})$ and $c = -1/\tan \theta_{pitch}$, [5].

Additionally, pitch angle θ_{pitch} computation is based on the assumption that HL is captured from the UAV camera in infinite distance [68], [72]. Accordingly, θ_{pitch} is calculated using the current and the initial HL positions as in [5].

$$\theta_{pitch} = \tan^{-1} \left(\frac{\frac{\rho_i^*}{\sin \theta_i^*} - \frac{\rho_0^*}{\sin \theta_0^*}}{f} \right) \quad (4.4.3)$$

where f is the camera focal length and where ρ^* is the Hough parametric space estimate perpendicular distance (equation 4.1.1). The described ϕ_{roll} and θ_{pitch} derivations and relevant geometry are shown schematically in figure 4.4.2.

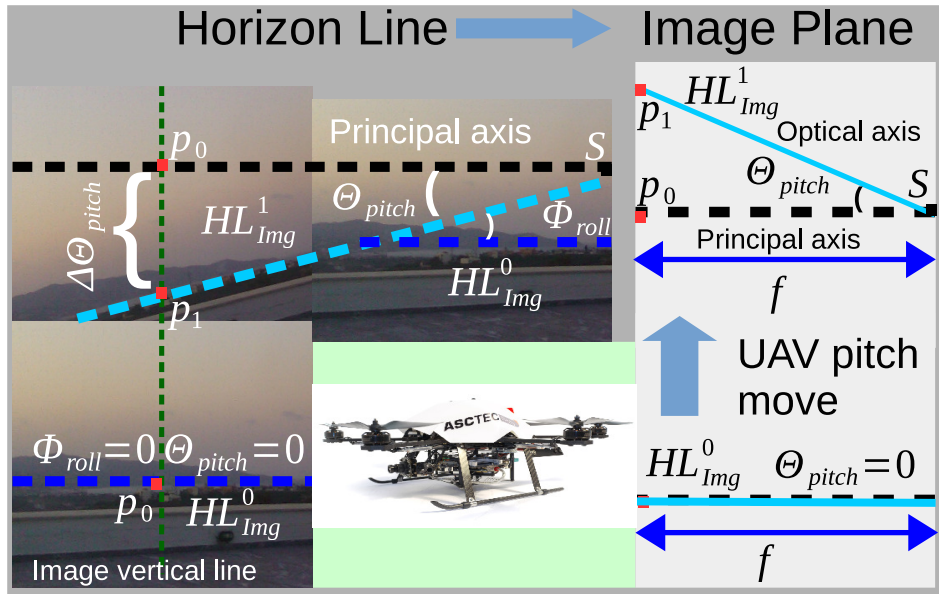


Figure 4.4.2: Pitch and roll angles detection. The ϕ_{roll} angle is readily computed from the horizon angle. The θ_{pitch} angle is the angle between the camera optical axis (cyan line) and the principal axis (black dashed line) in camera image plane. In addition in Horizon line plane, the roll angle ϕ_{roll} is the angle between the HL^0_{Img} (UAV in ground level) and HL^1_{Img} horizon lines. The pitch θ_{pitch} is the angle between the HL^0_{Img} and the line joining the camera center S with point p_1 , where the position of point p_1 is determined by the intersection of horizon line HL^1_{Img} with the image vertical line (green dashed line), which passed through the p_0 principal point. The p_0 is detected by HL^0_{Img} line, when camera is initially leveled with the ground (i.e. $\theta_{pitch} = 0$, $\phi_{roll} = 0$).

Chapter 5

UAV Visual Navigation

For an autonomous mobile robot to navigate in an environment shared with human, it needs to have a representation of its workplace. The ideal representation depends on the scope of the robot's task. If the task of the robot is navigation, for instance to deliver a package, a useful environment representation is a map of landmarks that the robot is capable to detect with its on-board sensors. If this map is not existent a priori, the robot has to incrementally create a consistent map of landmarks while simultaneously determining its location within this map. This problem of simultaneous localization and mapping (SLAM) has been a cornerstone in robotics research. With the advent of probabilistic approaches in robotics, the problem could be formulated and solved in various forms. Today, SLAM algorithms are a standard module on mobile robots in research labs, which have been implemented using different sensor modalities, such as laser range finders, sonars, depth-cameras, or standard cameras. Using cameras for localization and mapping is appealing as these sensors are small, inexpensive, power efficient, and ubiquitous.

One of the most difficult challenges in autonomous navigation is the localization problem. Solutions to the latter for UAVs can be divided into two categories: (i) those that provide absolute positioning, and (ii) those that provide positioning via incremental localization (also called dead-reckoning) or simultaneous localization and mapping. The absolute position localization provides an estimate of the robot's pose independently on the previous robot's states. Examples of absolute localization systems are GPS, or a motion capture system and geodetic total station. In this chapter we will introduce the related work in both areas. However given the focus of this thesis is visual navigation, this chapter is mostly dedicated to exploring this solution.

The major goal in this thesis is to enable real time autonomous UAV missions, namely, to address the demand of smoothly real time navigation on computationally and payload constrained UAVs. Therefore, in this chapter, we propose a UAV navigation solution, that employs the control strategy and the multi-sensor EKF fusion framework, presented in chapter 3. The latter are implemented a PD controller, along with an IMU, a dual visual sensory system and a full filter state, that includes sensors 6 DoF pose, UAV x , y and z velocity, position, pitch roll and yaw angles and bias (see equation 3.3.26). However, control structures rely on accurate information on the UAV state, which can be handling with the combine use, as already mention, of a 6 Dof pose estimation though a visual sensor and an IMU. While an IMU provides fast, but drifting measurements, the 6 Dof visual sensors provide absolute (or at least slowly drifting) measurements – but usually at a much lower rate. These camera measurements are accomplished via the proposed real time SLAM framework, described in this chapter.

Furthermore, computational advantages are emerged from the fact that the proposed here SLAM approach is a key-frame based algorithm similar to structure from motion (SFM) methods in robotic vision. A key-frame represents a distinct camera frame associated with a 6 DoF pose and corresponding 3D features. The sparsely placed key-frames, set with their associated features, represent the map in which the camera is localized at every camera frame. A batch optimization algorithm ensures the consistency of all feature positions and key-frame poses. In a previously mapped area, only the tracking task for the current camera frame within the current map has to be performed, thus, only the addition of new key-frames to the map (i.e. exploring new environment) is computationally critical.

Hence, the proposed navigation framework, illustrated in figure 5.0.1 consists of two main modules: (a) computation of 6 DoF visual poses from multiple sensor and algorithms, e.g. HL, and (b) the sensors measurements EKF integration. The first module comprises the map based localization approach, namely the RT-PTAM SLAM, which is formulated as

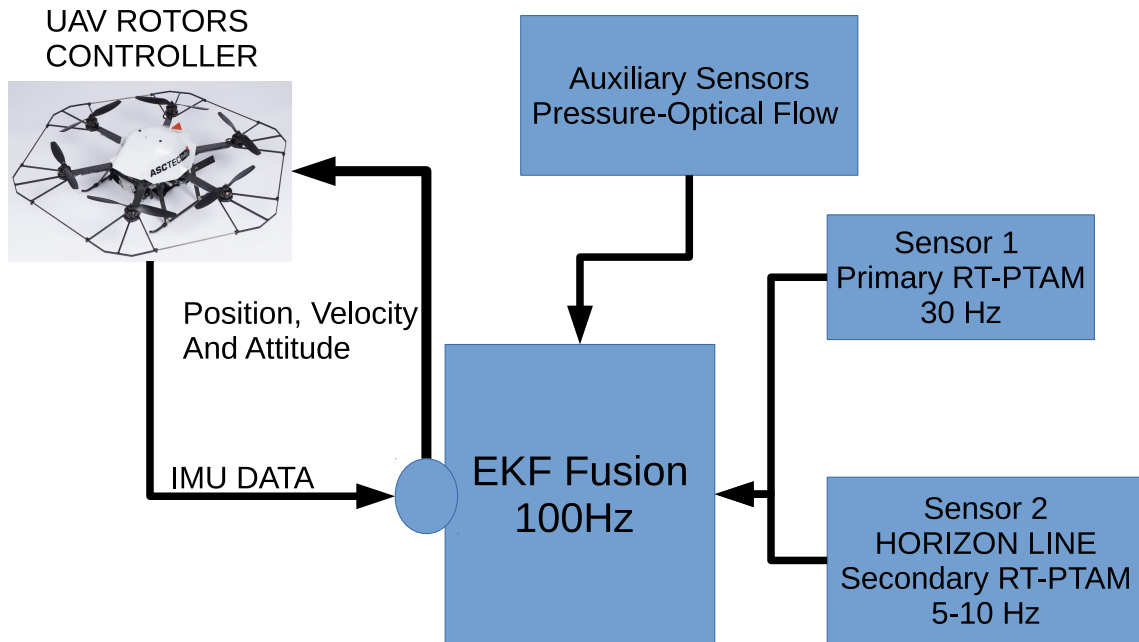


Figure 5.0.1: Navigation framework with multiple sensors. Sensor 1: A monocular visual sensor and the primary RT-PTAM is used. Sensor 2: Second monocular visual sensor, where a combined Horizon Line detection and RT-PTAM is utilized. Auxiliary sensors: May include a pressure sensor (presented in section 3.3.4, where we additionally mentioned that it is relatively easy to include additional sensors if available), or an optical flow module. The EKF is then responsible for the sensors' measurements fusion and is updating the state estimation to provide the UAV micro-controller with the updated state and velocities, enabling real-time navigation.

optimized PTAM visual SLAM, presented in [37]. The RT-PTAM SLAM reduces PTAM SLAM computational demands and produces more robust maps, in order to enable real-time on-board UAV computation. In addition, a secondary 6 DoF visual pose measurement is collected by a secondary visual sensor, which is employed to detect the HL and derive the UAV pitch and roll angle as presented in chapter 4, and at the same time a secondary RT-PTAM SLAM is used to obtain the additional DoF, that are demanding for full EKF state update. The latter navigation module employs and fuses the sensors' measurements in the EKF. The updated EKF state is then used to control the UAV, as we discuss in chapter 3. The chapter is organized as follows. Section 5.1, introduces the general localization approaches. In section 5.2, the map-based approaches are presented, namely, visual odometry and the visual SLAM (VSLAM) approaches, where the camera is deployed as a pose estimator. The main RT-PTAM SLAM framework is established in section 5.3, consisting of the core RT-PTAM in section 5.3.1, the RT-PTAM key-frame handling presented in section 5.3.2, and the RT-PTAM map optimization in section 5.3.3. In section 5.4, the Horizon line fusion to navigation framework is addressed. We conclude the chapter in section 5.5, where we present the EKF and sensors' measurements handling, that provide us the request from the controllers 6 DoF pose, for enabling UAV real-time navigation on computationally and payload-constrained UAVs.

5.1 Localization Approaches

The most common radio-based navigation solution for autonomous vehicles is the Global Positioning System (GPS). The GPS system consists of a network of satellites, which are equipped with an atomic clock, that is synchronized with the clocks on-board other satellites and the ground. The satellites continuously broadcast the current position of the satellite and a ground station can use the satellite signals to compute the distance between satellite and ground station. There has been much success both in research and industrial work with UAVs using GPS-based autonomous navigation [96], [97] and [98]. Other radio navigation solutions are also possible with the growth in coverage of wireless local area networks

(WLAN) in recent years, particularly radio localization using WLAN systems, [99], [100].

In contrast to GPS scheme, which is built for navigation, in typical WLAN solutions, precisely timestamped signals that include the position are not transmitted. This interpretation, in most of WLAN approaches, has led to utilize the WLAN signal strength, namely the Received Signal Strength Indicator (RSSI), rather than the usual concept of time of flight to determine the distance between the base station and the receiver device. In [101], the authors present a novel indoor localization, that is based on RSSI signal feature extraction and learning, where stacked autoencoders, a prominent example of a deep learning architecture, and random projections, a universal feature extraction approach are considered. Main drawbacks in WLAN navigation methods are the precision, reliability and the overall coverage of GPS solution. Additionally GPS methods is affected by two sources of interference, e.g. atmospheric conditions both in the ionosphere and the troposphere can perturb the very weak signal sent by the orbiting satellites. These interferences, for UAVs multi-path navigation, have much larger effect, that is, when some or all of the signals do not travel directly to the receiver but instead are reflected off surrounding environmental features such as mountains and tall buildings. This can further reduce the accuracy of GPS. The accuracy is improved in differential GPS approaches, where the atmospheric error is corrected, even so the sensitivity to multi-path interference and lack of coverage indoors, makes GPS navigation only applicable for large open area navigation. On the other hand WLAN methods in many occasions, suffer from lack of straight line of sight between base station and receiver and the availability of WLAN transmitters outdoors. Hence the RSSI signal transformation to distance becomes not relatively obvious, as the propagation of reflected WLAN signals is dependent on many factors some of which include the properties of the occluding material and the frequency and signal strength of the base station [102].

Another approach to indoors autonomous navigation, is the use of high speed motion capture systems such as Vicon. These vision based absolute localization approaches usually rely on recognition of the special pattern attached to the robot, e.g., April Tags, [103]. Furthermore these systems use a set of high speed infra-red cameras coupled with infra-red emitters to precisely track reflective markers which can be attached to any object and provide sub-millimeter precision at very high update rate 100-2000 Hz [104]. Nevertheless, there are several limits of using such a system for UAV localization and navigation. Primarily these systems have reduced scalability and coverage, meaning that are able to only track a fixed number of UAVs and cover a rather limited and indoors only area, since for accurate localization the UAV has to be within the view of at least 3 cameras, which increases significantly the solution cost.

All previous discussed navigation approaches, provide absolute position information with respect to some fixed reference frame. Another approach is the incremental localization, which creates a fixed coordinate system on-the-fly using observations from sensors while simultaneously localizing within that coordinate system. The incremental localization has prior knowledge of the UAV starting pose. One of the common used incremental localization methods is the odometry and SLAM or map based approaches, presented in next section. These solutions can most readily be categorized by the type of sensors used, particularly, 6 DoF vision sensors and laser range-finders approaches.

Laser range finders provide direct measurements of the distance between the sensor and environment and distance can be calculated based on the time of flight. The use of laser range-finder readings to solve the SLAM problem has been extensively studied with respect to ground-based robots. Regrettably these ground-based approaches do not directly translate to UAVs equipped with laser range-finders, and present challenges, including the 3D motion of the UAVs, the lack of wheel or Humanoid walking odometry and the availability of limited UAV on-board computational resources. Thus the current laser range-finder technology has limited applicability on-board UAVs, as while they provide highly accurate readings, the laser units are often heavy and consume enough power to make them impractical to use.

5.2 Map base Localization

In this section, we present the map based approach, where we use the visual sensor as a pose estimator. While the most accurate solution to offline structure from motion (SFM, section 2.3) problems is undoubtedly to extract as much correspondence information as possible and perform batch optimization, sequential methods suitable for live video streams must approximate this to fit within fixed computational bounds. Two quite different approaches to real-time visual SLAM have proven successful, but they compromised the problem in different ways, the filtering SLAM methods, [105] and more recently [106], which summarize the information gained over time with a probability distribution and bundle adjustment

(SFM-BA) SLAM methods, i.e. PTAM- SLAM. SFM tackled problems of 3D scene representation from small sets of images, and projective geometry and optimization have been the prevalent methods to derive relevant solutions. In filtering approaches, on the other hand, the classic problem is to estimate the motion of a moving robot in real-time as it continuously observes and maps its unknown environment with sensors which may or may not include cameras. In addition Bundle adjustment in SFM, or the EKF and variants in SLAM, all manipulate the same types of matrices representing Gaussian means and covariances.

There are several important differences between the visual SFM-BA frameworks and the standard filter-based approach. SFM-BA algorithm does not use an EKF-based state estimation and does not consider any uncertainties, for the pose of the visual sensor or for the location of the features. This considerably reduces computational complexity, however, the lack of modeling uncertainties is compensated by the use of a large amount of features and the global and local batch optimization. In contrast, the filter-based SLAM approaches pose and feature locations are estimated jointly within the state of a EKF, and as EKF state includes the visual sensor pose, with each new image the complete state of the filter must be updated to achieve reliable tracking. However, given that the state contains sensor pose and feature positions this update is costly and scales fast with the number of feature points. In addition, not only does this limit the maximum size of the map, and it is also computationally inefficient considering the limited new information based on features, that is gained by processing every frame. Hence, the algorithms in SFM-BA approaches, are using a fixed and sometimes limit area for feature matches, are still able to track efficiently the point features and to close loops up to a certain extent.

Furthermore, many SFM-BA approaches, separate the tasks of camera pose tracking and map building. This allows them to use each frame for the computationally less expensive process of real-time tracking and then build a map from only those images, which provide new information, these are referred to as the key-frames. This makes the SFM-BA extremely fast and reliable, and the built map very accurate; as demonstrated in [44], SFM-BA SLAM algorithms can outperform filter-based SLAM. This removes the real-time constraints on the map building components allowing the use of more costly but more precise techniques for map optimization. The extent to which the map is optimized is, what differentiates key-frame based Visual odometry from Visual SLAM methods.

Alongside with the previous discussion, the real-time visual SLAM approaches are additionally based on geometry and scene perception, and a slight distinction is defined as single and stereo visual SLAM or odometry. A passive stereo camera consists of two separate cameras mounted side-by-side at a fixed, known distance. Stereo cameras replicate human binocular vision and allow the recovery of depth data from an observed scene. The main advantage of stereo cameras in terms of visual navigation is their ability to recover scene depth from a single pair of images. This makes it possible to recover both the 3D structure of a scene, as well as the camera position in metric units, which enables real time collision avoidance. An alternative stereo visual sensor is the use of an active stereo, based on structured light. In such systems the second visual sensor is replaced by a patterned-light projector. The visual sensor then observes the scene and afterwards, can compare the projected pattern to compute the depth.

The major drawback to stereo vision systems for navigation, is the reliance on the fixed baseline between the two cameras. As depth is estimated from point disparity between the two images the range in which depth can be calculated is fixed by the baseline between the cameras. Points at longer range will exhibit little or no disparity between the two cameras meaning depth values cannot be computed. Hence, this solution loses its range measurement ability for scenes far away, with respect to the stereo baseline, particularly, in height attitude UAV navigation. Additionally, one of the drawbacks of passive stereo sensor are the need for highly textured scenes to facilitate point matching and triangulation, in contrast active stereo vision systems is able to operate in low texture environments. However active stereo systems typically make use of infra-red pattern projectors in conjunction with an infra-red camera, as this simplifies detection of the projected pattern in the captured image. This limits the use of such sensors to indoor environments, where there are typically fewer sources of infra-red light.

While the scene depth is a welcome asset, along with the previous mentions issues, stereo systems, are added computational complexity in UAVs constrained computational capabilities and are introduced real time synchronization faults between stereo camera pair, which is consisted from dual single visual sensors set-up with fix baseline and additional sensors systems, e.g. IMU. The latter can be reduce or eliminate with a embedded and synchronized passive stereo device. However, this increases significantly the solution cost, and based on the needing for UAV outdoors navigation, which totally exclude active stereo approaches, in this thesis a monocular visual sensor system is selected. Using a single visual sensor, is a feasible alternative to stereo or laser based system, due to the significant, smaller size, less weight and reduced

power consumption requirements. However, in contrast to all the previous sensors mentioned, a monocular camera has no means of reliably obtaining depth directly from a single image. This can be obtained by matching and triangulating common features from multiple views (i.e. moving the camera). Additionally, in the monocular sensor system used in this thesis, scene metric and scaling, absolute height and position recovery is only possible via the use of additional sensors, e.g. IMU, or via recognition of objects of known size in the environment.

Before we proceed with Visual Slam algorithms and eventually the proposed RT-PTAM SLAM, an introduction to the localization framework of Visual Odometry is presented in section 5.2.1.

5.2.1 VISUAL Odometry (VO)

In previous work on monocular vision-based navigation for UAVs, Visual Odometry (VO) is a sufficiently address subject. VO common approaches make use of real time or "natural" geometric features, namely not artificial features (Visual Markers), and the system can operate in unprepared environments. As already mentioned, in VO or SFM, the aim is to track the motion of the sensor, by continuously computing the relative motion between frames using feature correspondences. Typically VO uses a visual sensor mounted to the robot, while processing consecutive image frames to estimate a change of the camera's pose. Each change of the visual sensor's pose between pairs of consecutive frames is estimated by detecting correspondences between these frames.

Visual Odometry has been shown to work well for ground based vehicles and robots where the constrained motion of the vehicles serves to simplify the problem. However, in UAV similar assumptions may be made to simplify the problem, for example a UAV, using motion from a down-facing camera to compute its translational velocity. An already mention problem of monocular camera that arises, is the camera motion at different heights is scaled, particularly this is meaning, the higher the UAV flies the slower, it seems to move in camera. Another issue is the rotational movement of the UAV while translating, since the calculated visual motion includes the movement induced by both the translation and the UAV rotation. Additionally, if the UAV changes height between frames this would also affect motion calculation. These effects can be compensated for, if both the relative rotation and relative height is known or measured by other sensors, such as sonar and IMU.

In a planar scene, the transformation between the two images Im_1 and Im_2 is described by a homography transform matrix T_{hom} , where D_{height} is the relative difference in height above the plane, with $H_{relative}$ the relative height and $R_{rotation}$ is the rotation matrix between the two frames. t is the relative transformation and G is the normal vector of the ground plane:

$$T_{hom} = R_{rotation} + \frac{1}{D_{height}tG^T} \quad (5.2.1)$$

The UAV velocity, using the motion metric M and focal length of the camera f is computed by:

$$v = H_{relative} \times \frac{M}{f} \quad (5.2.2)$$

We can then use the known values and homography T_{hom} to transform image Im_2 to have the same rotation and height as image Im_1 and calculate the translation from the visual sensor motion, [107]. However the combination of camera and IMU means the rotation for both frames is known, leaving on the relative translation unknown. The addition of an IMU can also be used to simplify the problem in other ways, computing the essential matrix $ES \in \mathbb{R}^{3 \times 3}$, where $[t]$ is a skew symmetric translation matrix:

$$ES = [t]_{\times} R_{rotation} \quad (5.2.3)$$

which describes the transformation between the two frames, [108].

As the essential matrix is a 3×3 it appears that there are 9 unknowns, however to compute the ES a 8-point linear algorithm algorithm is sufficient, since we can arbitrarily scale the essential matrix ES , and still satisfy the epipolar ¹ constraint, meaning we can only compute ES to scale and this leaves 8 unknowns. In addition, with the combine use of camera and IMU, the rotation angles for both frames is known, leaving only the relative translation unknown, and in this

¹In particularly, here this means that the corresponding feature point in the second image is guaranteed to be found along the epipolar line, thus this limits the search space for point correspondences to a single line.

case the essential matrix computation, is required a 3-points algorithm , [109]. Then, the approach to estimate the camera movement is generalized to be applied to more than two images, by reconstructing the 3D structure of the local scene, by feature point triangulation and minimizing the re-projection error over the i closest frames, using a non-linear least squares approach. The main drawback, when we only compute the relative pose between two images to obtain the cumulative trajectory of the UAV, is small errors in the relative pose estimates accumulate over time and lead to a large error in the overall UAV trajectory [40]. This approach, known to as, the window Visual Odometry and is very closely related to Visual SLAM and often share the same components. However the main difference, is that VO focus is to compute a consistent trajectory of the visual sensor in real time as opposed to Visual SLAM, which aims to jointly reconstruct the camera trajectory and scene structure in a globally consistent manner.

5.2.2 VISUAL SLAM (VSLAM)

Visual SLAM (VSLAM) systems take advantage of the rich information about the captured environment contained in image data to provide a reliable localization of a mobile robot. In VO approaches we are only concerned with visual sensor trajectory and as such, map optimization approaches tend to be limited to a small local area, particularly in the most recent i -frames. In a VSLAM system we are also concerned with constructing a consistent map. The extent to which the map is optimized is what differentiates the mentioned window- Visual Odometry from VSLAM. One of the limitations of the VO frameworks is the lack of large loop detection and closure. During VSLAM mapping loop detection refers to the problem of recognizing, when the UAV has returned to a previously mapped area.

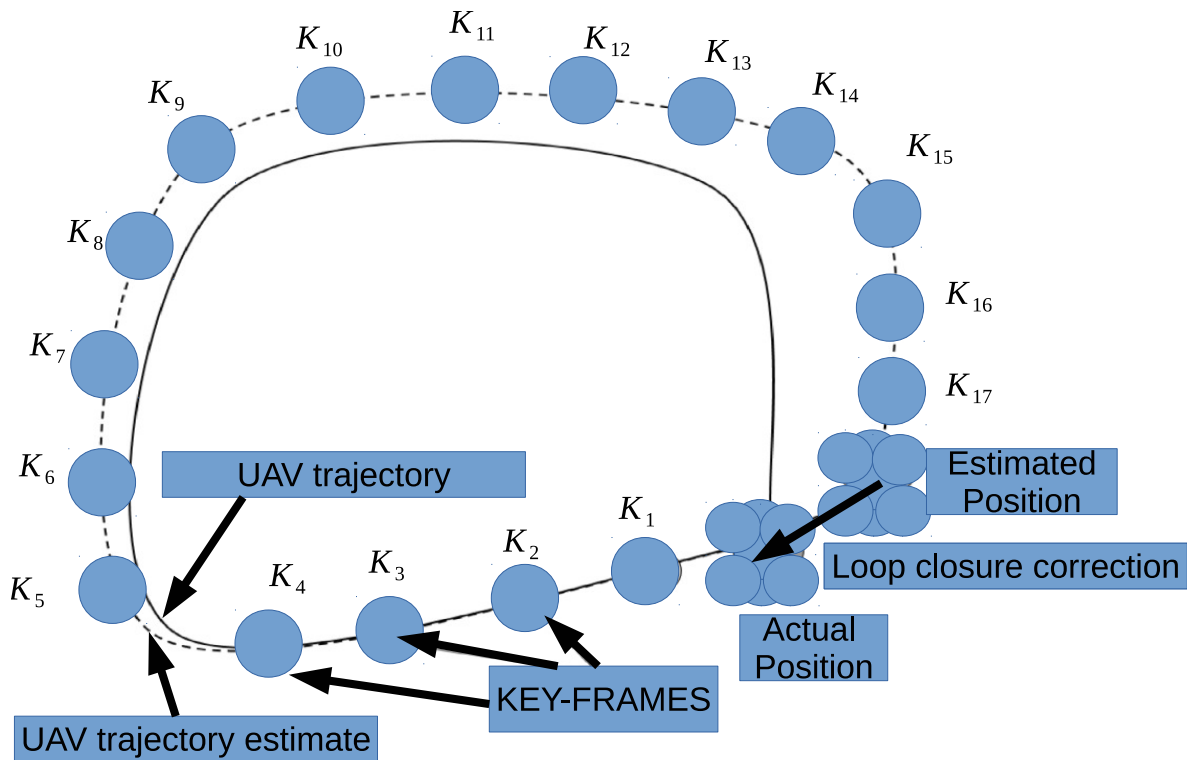


Figure 5.2.1: A simple example of a loop closure situation. The UAV’s trajectory is shown as a solid line and it’s own trajectory estimate is given as a dashed line. In this example, a visual sensor equipped in a UAV follows the trajectory shown building a map of visual features, the stored sensor poses or key-frames are indicated as circles. The accumulated errors in the UAV’s position estimate, give rise to a drift for both the current position estimation and the key-frame position, with respect to the ground truth (solid line). To correct this the system must be able to compute the transformation (the loop closure correction) required to realign the map. This correction can then be back propagated throughout the map to correct all key-frame positions.

In figure 5.2.1 the loop closure detection (also called visual place recognition) is presented, a well studied problem in robotic vision. The most widely used loop closure approaches in VSLAM methods, are the image matching approach, where image features are utilized to determine the similarity between two images. In this thesis we proposed, an efficient direct image matching approach based on key-frame SLAM approach, described in next sections. The image matching approaches, has been shown to work well, where the density of frames is high and the relative viewpoint between the current frame and the closest recent frame is similar. Before we discuss our proposed V-SLAM, an introduction to VSLAM approaches is following.

VSLAM systems can be divided by the way the image data are processed to full and online, sparse and dense, and direct and indirect. Full VSLAM estimate the whole robot's path and are capable to recalculate the path when a loop closure is detected. Whereas, online VSLAM seek to estimate only the current camera pose. The advantage of full VSLAM, is that they can improve quality of the localization by means of improving the underlying map of features, whenever the robot revisits the same place. On the other hand, online VSLAM usually have lower hardware requirements, as it is supposed these VSLAM are running on-line and on-board of the robot computational resources.

In addition, the amount of information taken from an image can be utilized to further classify a VSLAM system as sparse or dense. VSLAM systems that classified as sparse, use just a small set of image pixels to construct a coarse map of the environment and calculate the camera's pose. Examples of sparse VSLAMs, are the key-frame VSLAMs, i.e. the ORB-SLAM [110], which address the problem of place recognition using ORB features, like SIFT and SURF features. ORB features are invariant to rotation and scale making them useful for place recognition, but are computationally demanding. On the other hand, in dense VSLAMs, all or most of pixels in an image are employed to construct a detailed map of the environment and compute the camera's pose. The advantage of dense VSLAMs, are that their underlying map contains more details about the robot's surrounding and the map can be used also for other purposes, e.g., collision avoidance. However computational demands of dense methods, e.g. Large Scale Semi-Direct (LSD) SLAM, [111], are really high and these methods usually require image processing on a graphic processing unit (GPU).

Therefore, in previous mentioned LSD SLAM an optimization is introduced, where instead of computing a depth map for the entire image, image regions are selected with large gradients which provide the most accurate depth information. This hybrid approach is akin to using both point and line features for tracking. This allowed their system to operate in real-time on a CPU and still build semi-dense maps of the environment. The main limitation of the LSD-SLAM approach is the limited map optimization, as the system does not compute descriptors for image regions, it is not possible to re-project map features and optimize the map via re-projection minimization. A relative comparison presented in [110] between ORB SLAM, PTAM SLAM and LSD SLAM, in which the localization accuracy of LSD SLAM was shown to be poorer than both ORB SLAM as well as PTAM. This is largely attributed to the lack of structural refinement, i.e. Bundle Adjustment present in both PTAM and ORB SLAM.

The last type of categorizing SLAM systems, is based on the way of using the image data, which can be done directly or indirectly. Direct SLAM methods make use of measurable image quantities (e.g. intensities) for each pixel in the image rather than extracting a sparse set of features (indirect methods). The indirect SLAM methods compute a special representation of the selected pixels and their neighborhood (image features), and then a depth of each individual feature is calculated. The advantage of direct methods, is that the whole image can be used for tracking rather than a sparse set of features and allowing the construction of more complete environment maps rather than sparse feature maps. A direct Slam is presented in [112], namely the Direct Tracking and Mapping (DTAM). In DTAM, tracking is done using whole image alignment using the depth-map. This approach is computationally very intensive and requires large scale GPU parallelisation to run in real-time.

To address this issue many semi-direct approaches, e.g. LSD SLAM and Semi Direct Visual Odometry (SVO) [113], have been developed. In SVO SLAM a direct image based alignment is applied, but instead of operating on a dense or semi-dense depth map, a sparse 3D map is built similar to the key-frames and features based methods. The addition of the direct-based tracking makes the tracking very robust even with very fast camera movements. However the inclusion of feature points means, that they get all the benefits of a bundle adjustment based mapping thread for map optimization. One important note however is, in order for SVO to operate in real time on-board and MAV the maximum number of key-frames is heavily restricted.

In the next section we describe in detail the key-frame based Visual SLAM proposed approach, and as real time UAV implementation is the focus of the work presented in this thesis, the following sections will additionally explore critical

optimizations in our SLAM approach in more detail.

5.3 RT-PTAM - VISUAL SLAM

The state of the art for Visual SLAM on UAV can be divided into two categories, off-board where the complete Visual SLAM system runs on a suitably powerful ground station computer and on-board where the full system runs on-board the UAV. Running a full monocular SLAM system on-board a MAV is a challenging task. One of the main limitations is bundle adjustment. The complexity for straightforward bundle adjustment is $O((k+n)^3)$ with n features and k key-frames and even sparse bundle adjustment has a cubically scaled computational complexity of $O(k^3 + kn)$, [37]. In key-frame SLAM the mapping thread regularly performs a local BA on a selected key-frame and its four closest neighbors based on euclidean distance, and only performs a global BA when the tracker is operating in previously mapped areas.

In this section, we present the proposed in this thesis VSLAM, namely the RT-PTAM SLAM, e.g. a map based approach, which provided us with the 6 Dof measurements for the EKF update module discussed in section 3.3.3. RT-PTAM SLAM approach, is based on the a visual SLAM algorithm PTAM, which is found to be both computationally less expensive and more robust than classic filter-based visual SLAM implementations, [44]. The RT-PTAM algorithm forms the basis for the UAV real-time SLAM in this thesis, discussed in the following sections, illustrated in figure 5.3.1.

5.3.1 RT-PTAM CORE

In summary, RT-PTAM approach is splinted the simultaneous localization and mapping task into two separate threads: the tracking thread and the mapping thread, presented in figure 5.3.1. The tracking thread is responsible for the tracking of salient features in the camera image, i.e., it compares the extracted point features with the stored map and thereby attempts to determine the pose of the camera. This is done with the following steps:

RT-PTAM SLAM Tracking Module

1. *A simple motion model C_t^m is applied to predict the new pose of the camera from the previous pose C_{t-1}^m , where M is a 4×4 camera motion, belongs to $SE(3)$ and can be minimally parametrized with a six vector μ using the exponential map, typically representing sensor motion translation and rotation.*

$$C_t^m = MC_{t-1}^m \quad (5.3.1)$$

2. *Then the stored map points are projected into the camera frame and corresponding features are searched. This step is often referred to as data association.*
3. *A coarse scale search is conducted for a small number (50) of features (the data association step) and the camera pose is refined based on the matched features.*
4. *A larger number of features (1000) is projected and searched for in the image.*
5. *Next, the algorithm refines the orientation and position of the camera such that the total error between the observed point features and the projection of the map points into the current frame is minimized.*
6. *A final pose estimate is computed based on all the matched features found.*

Thereby, the mapping thread uses a subset of all camera key-frames to build a 3D-point map of the surroundings, presented in figure 5.3.2. The key-frames are selected using heuristic criteria, based on a distance measure and on the visibility (i.e. overlap) of the last key-frame. After adding a new key-frame, a batch optimization is applied to refine both, the map points and the key-frame poses. This attempts to minimize the total error between the reprojected map points and the corresponding observations in the key-frames, particularly a bundle adjustment is used to optimize the map (see Section 2.3.4). This is a key aspect of RT-PTAM SLAM approach, by decoupling the tracking and mapping processes into separate

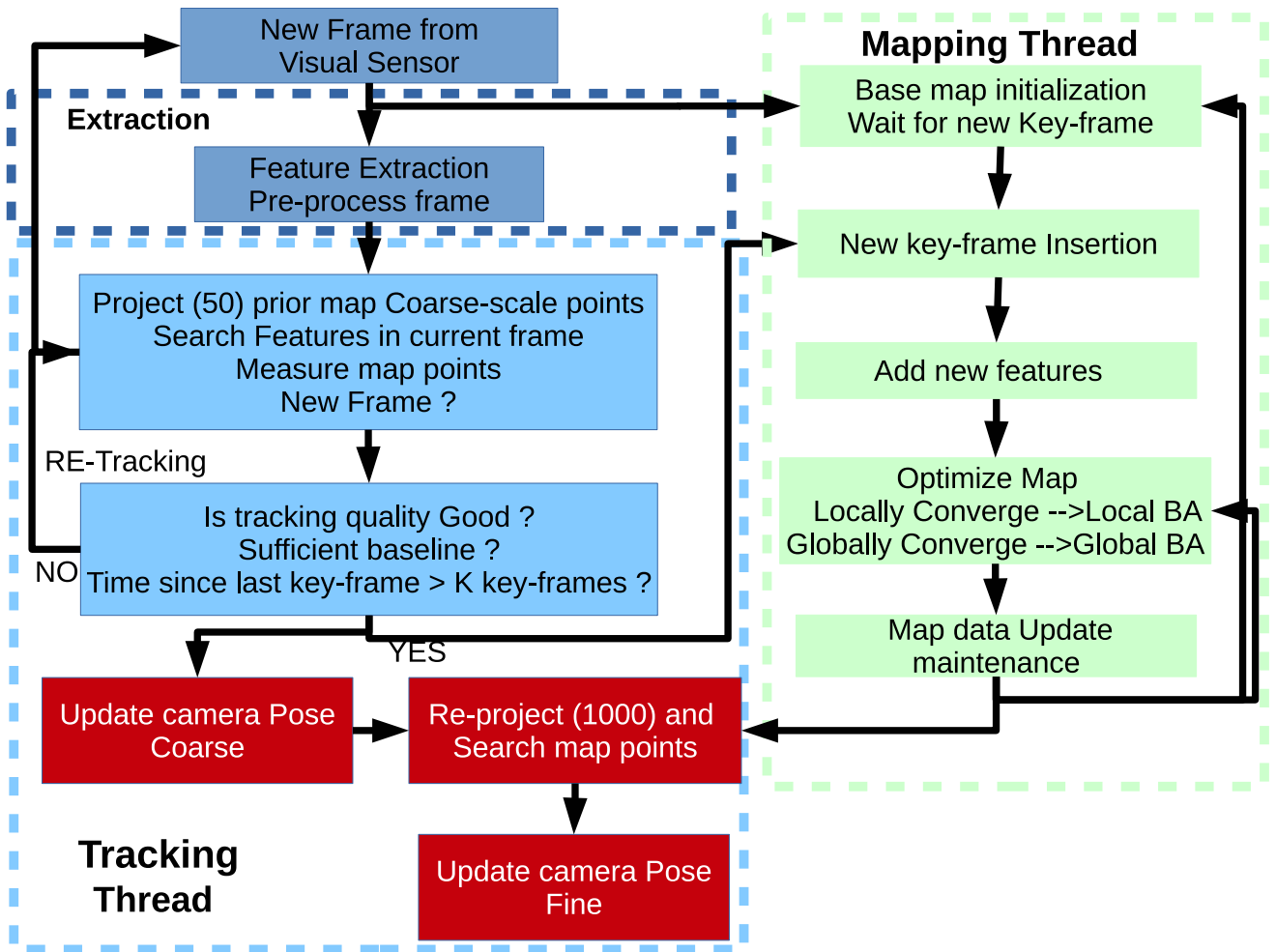


Figure 5.3.1: RT-PTAM flowchart, which illustrates its working procedure is consists by three threads, namely Extraction, Tracking and Mapping. A frame stream acquired by the visual sensor on-board a UAV, records the scenes of a relatively large region. The acquired image is processed to generate a pyramid key-frame, containing multiple levels of the frame at different resolutions. In order to be able to track the camera and map new points, a base map initialization phase is necessary. This is achieved using a standard five-point stereo algorithm between two key-frames, developed in [6]. The mapping phase of the RT-PTAM starts immediately after the map initialization process has completed and runs parallel along the tracking part. Mapping thread is responsible for providing the map, and optimizing the map using local and global BA with multiple loops runs, until the resulting map to be as detail and as accurate as possible. However, map optimization is most of the time a very slow procedure. In contrast, the tracking thread is implemented to be fast and robust, since its main task is the camera pose accurate computation, which is constantly needing in real time navigation.

threads this removes the real-time constraint on the mapping procedure imposed by the need to process every incoming frame.

The tracking and the mapping threads operate on the original image, as well as on down-sampled versions of it, or pyramidal levels (see section 2.5). On each key-frame creation, features extracted on different such levels are introduced to the map. A pyramidal level prediction procedure in the tracking thread searches the features stored in the map in the predicted pyramidal level of the current image. If a feature extracted at level 1 (first down-sampled image) and added to the map, it will be searched for in level 0 (original image) if the UAV moves away from it.

Therefore the main advantage of the thread splitting lies therein that both the mapping and the tracking thread can run at different frequencies, shown in figure 5.3.3 . Thus, the mapping thread is able to apply a much more powerful and

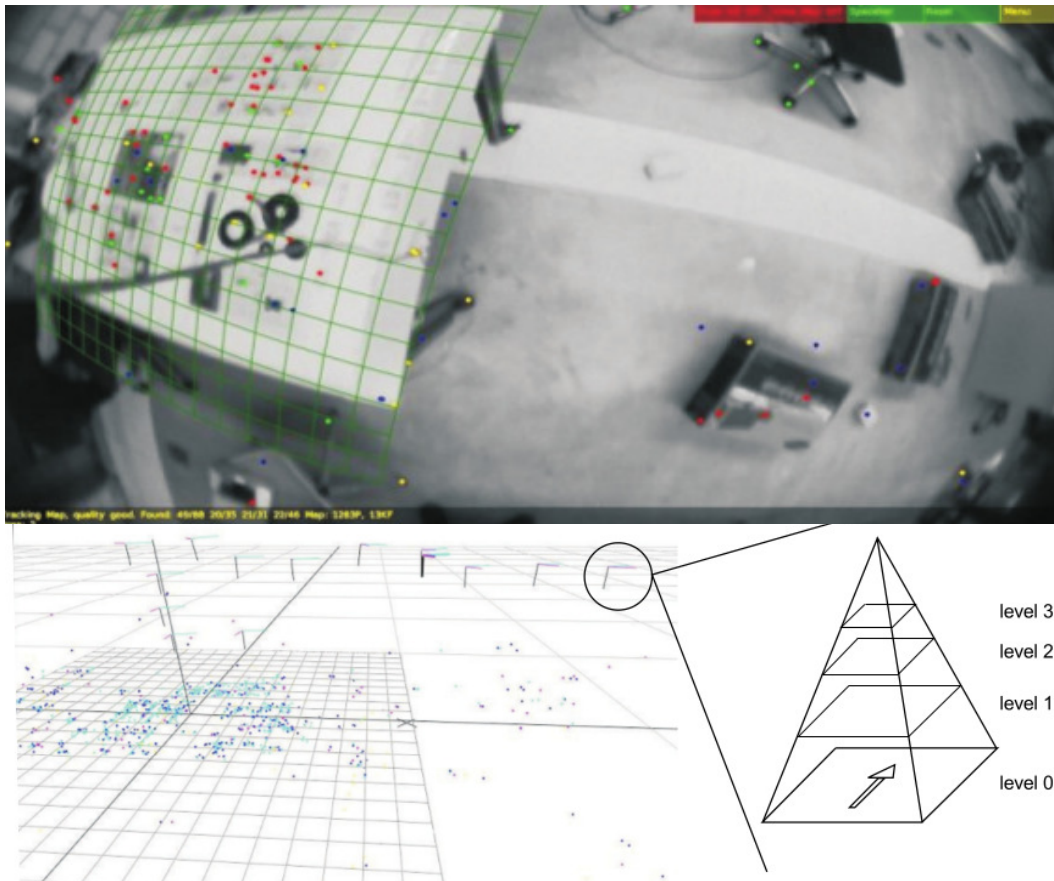


Figure 5.3.2: RT-PTAM SLAM scheme. The top picture is a RT-PTAM’s visual SLAM algorithm real-time visualization. The tracking of the FAST corners can be observed. This is used for the localization of the camera. In the bottom picture, the 3D map built by the mapping thread is shown. The three-axis coordinate frames represent the location where new key frames were added. Each key-frame consists of a 6 DoF camera pose, the camera image and its down-sampled pyramidal levels schematically depicted on the bottom right.

time-consuming algorithm to build its map. Simultaneously, the tracking thread can estimate the sensor pose at a higher frequency. This strongly improves the performance. Compared to frame-by-frame SLAM (such as filter based EKF-SLAM), the RT-PTAM algorithm substantially reduces computational complexity by not processing every single frame for the map. Particularly when using a camera with a wide field of view, consecutive images often contain redundant information. In addition, for example, when the camera is moving very slowly or if it stays at the same position, the mapping thread rarely evaluates the images and, thus, requires only very little computational power. This eliminates to a great extent redundant information processing during UAV slow movements or hovering. Additionally, a strength of RT-PTAM monocular SLAM algorithm is its robustness against partial camera occlusion. The large amount of features considered in each camera frame allows large occlusions while the pose estimate is still accurate enough to sustain stable UAV control. Furthermore, it is very easy to adapt and to optimize independently each of the threads to our specific needs on the flying platform. These are the main reasons we choose this SLAM algorithm. We describe our modifications in the following section.

However, this is still not enough to run initial RT-PTAM in real-time on-board a UAV. In order to analyze the behavior of the RT-PTAM SLAM we focus on, apart from the RT-PTAM speed, to the three types of behaviors we are interested in, when, using the estimated pose as input for a UAV controller, particularly, the scale drift, the map or pose drift, and the localization failures. A rotational map drift of the RT-PTAM SLAM algorithm is identified, and is non-negligible, so that the reference coordinate frame is no longer correctly aligned with gravity. Moreover rotations around the x and y axis are

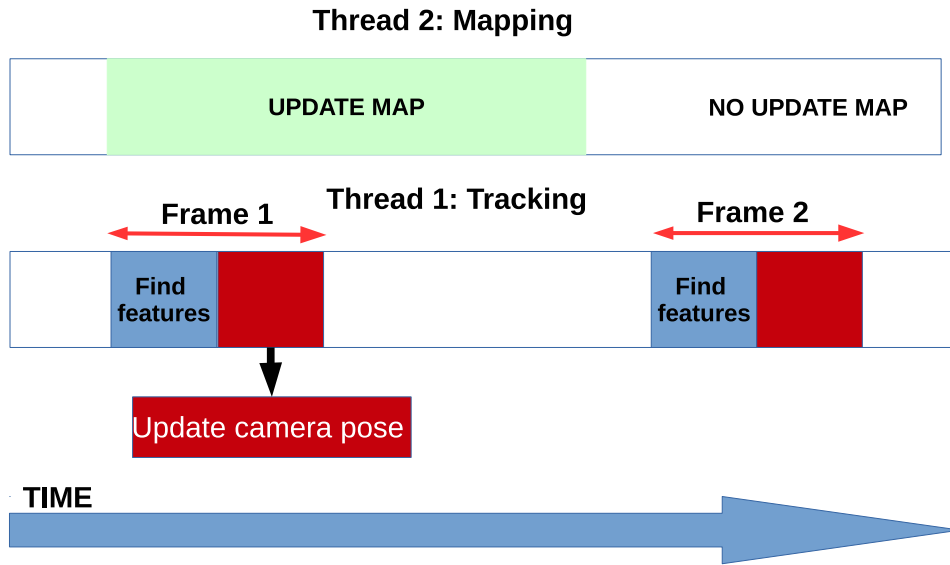


Figure 5.3.3: RT-PTAM SLAM mapping and tracking threads time representation. Mapping thread can take a great amount of time, in contrast tracking thread is more robust and computed in 20-30 Hz. Map is not updated in every frame.

error-based, as a result of not correctly decomposed into its x , y , and z values, the error between the reference value and the actual position, leading to UAV instability, if not considered.

Therefore we found that, if the movement occurs in the x -direction, the pitch-angle estimation is mostly affected by the drift. In the monocular SLAM framework RT-PTAM, these drifts can be minimized by applying navigation patterns with recurrent loop closures. A full SLAM system is, however, computationally too complex for large environments. In the following sections, we detail our modifications to leave the mentioned to PTAM SLAM “ [the] small AR workspaces ”. The changes enable on-board RT-PTAM operation at 20-30 Hz, and in larger, unknown and outdoors environments under difficult conditions.

5.3.2 RT-PTAM Key-frame Handling

The prior discussed topics and assumptions have established the base implementation of RT-PTAM algorithm. In this and the following section, we describe the suggested key-frame SLAM algorithmic modifications and optimizations, which are built in final RT-PTAM framework, in contrast with the base RT-PTAM. Similar modified key-frame SLAM based approaches, are used and proven to be more robust in [3], [114].

In base RT-PTAM, the map is defined as a set of key-frames together with their observed features. As the number of key-frames may grow quickly, especially in an navigation mission, we have to assume cubic complexity, as we already mentioned in previous section, relative to the key-frames number. This soon becomes computationally infeasible on constrained on-board hardware. In order to render the computational complexity constant, we set in RT-PTAM a maximum number of key-frames retained in the map. If this key-frames number is exceeded, we remove distant key frames from the RT-PTAM map (based on Euclidean distance) and the current UAV pose, along with the features associated to each removed key-frame. These have only small influence on the current estimate, shown in [115], [31]. If the maximum number of retained key-frames is infinite, then the RT-PTAM algorithm is equivalent to the core RT-PTAM. Conversely, if we set a maximum of 2 key-frames, we obtain a visual odometry framework. Values in between, approximate a sliding window RT-PTAM SLAM approach.

Therefore the larger the number of retained key-frames, the smaller the drifts, but also the larger the computational com-

plexity for the bundle adjustment step. This is a large improvement since usually the number of features is much bigger than the key-frames number. However, this error is found to be small, even with a relatively low number of five key-frames, and same results are proposed in [3]. Out of the six drifting states (6DoF visual sensor), drift of only two states, namely roll and pitch are observable in combination with the UAV IMU and our EKF framework. Definitely, these are the two states that would cause instability of the UAV, if their state drift was not compensated for, in contrast the other states, e.g. position and yaw may drift without affecting the flight stability. That is, drift in position and yaw only causes small deviations from the set point or the desired trajectory. This means that we can implement the RT-PTAM SLAM in the control loop at constant complexity and high update rates, while maintaining the local stability.

5.3.3 RT-PTAM Robust Map

This optimization aims at selecting high quality features for robust map generation and camera pose tracking. During outdoor UAV navigation, we experienced that self-similarity features of the environment is an omnipresent issue and are often high frequent, e.g. grass in rural areas and asphalt in urban areas, causing the tracker to fail tracking the current pose, and the mapper to fail re-localizing. Furthermore, the higher the resolution of the visual sensor, the more are these high frequent self-similar structures captured. Conversely, the low resolution cameras blur these structures very similar to a low pass filter. A similar effect have the down-sampled RT-PTAM pyramidal versions of a original image which are stored in the map's key-frames. Thus, features extracted at higher pyramidal image levels are more robust to high frequent scene self-similarity. Features in higher levels represent salient points in lower frequencies, whereas features in lower pyramidal levels represent salient points in high frequencies. Figure 5.3.4 depicts the situation schematically.

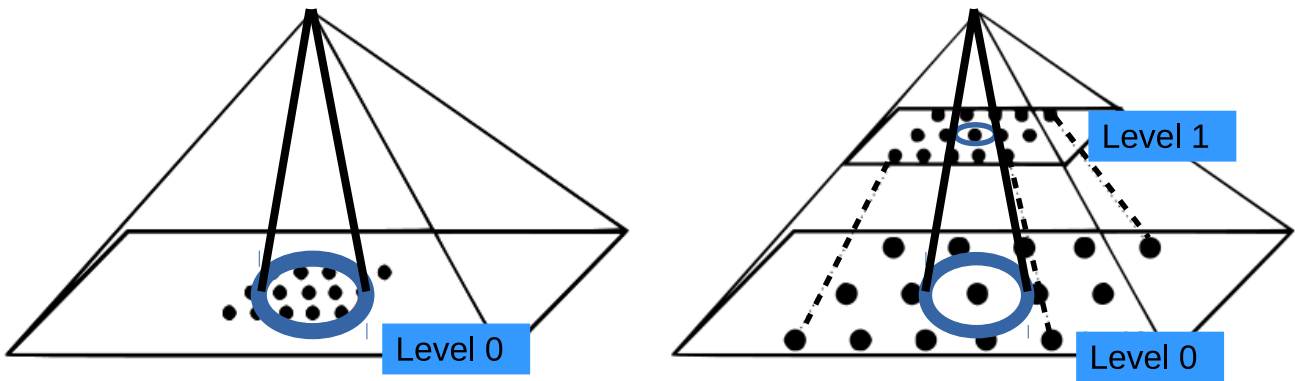


Figure 5.3.4: RT-PTAM and base RT-PTAM SLAM features detection. Right image: In base RT-PTAM the detected features of high-frequent, and self-similar structures are extracted in the lowest pyramidal level 0. Since most details are preserved in this level, the search radius calculated from the algorithm's motion model (blue circle) is covered several potential matches which leads to a wrong data-association. Left image: RT-PTAM, down-sampling the original image (i.e. higher pyramidal level 1) and acts as a low pass filter. Only features of low-frequent and self-similar structures are extracted and the motion model can disambiguate well between the different features.

In figure 5.3.4, the low frequent self-similarity is well handled by the RT-PTAM motion model, limiting the search radius for corresponding features (blue circle). In contrast, the search radius contains several matching candidates points on high frequent self-similar structure. Since the algorithm's motion model is not very precise, the feature search region is larger than the spatial image periodicity in the lowest pyramidal level 0. In figure 5.3.5 is evident that most were detected in level 0. On the remaining levels, the ratio is lower, since the smoothing involved in pyramid sub sampling, reduce the high frequencies unstable features. This inevitably leads to wrong matches. Furthermore, the results in unstable feature tracks over consecutive frames and can cause map-losses if this happens often enough.

Hence, we only include features extracted in the higher pyramidal levels for the map building process in RT-PTAM, and directly avoiding issues with high frequent self-similar structures, despite this, it is still important to extract features on

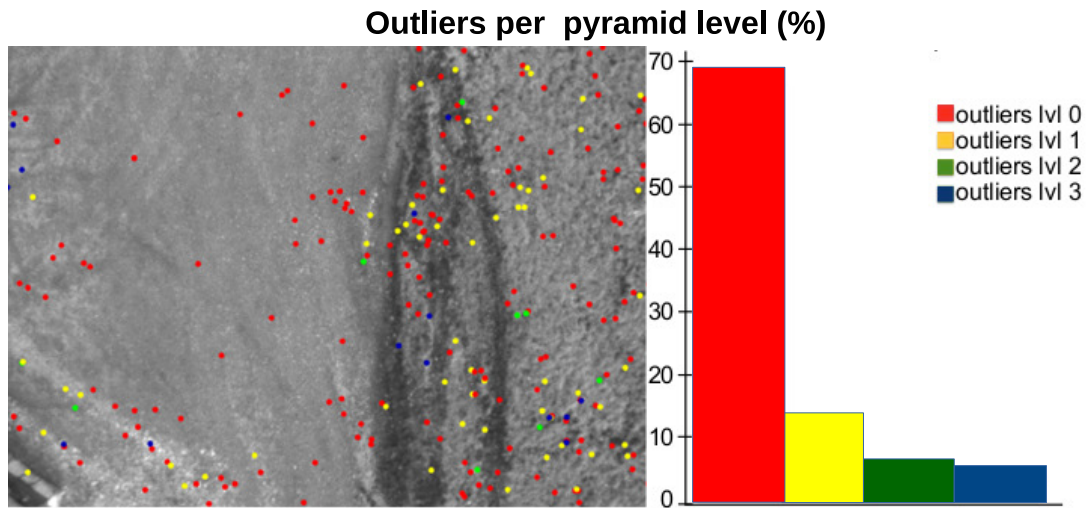


Figure 5.3.5: RT-PTAM feature detected for each pyramid level. Right image: shows the ratio of outliers that RT-PTAM reported after optimization, with respect to the number of features detected. Left image: Feature detection in a typical image taken by UAV onboard camera during outdoors navigation. The colors of the features points are based on the pyramid level that are detected [3].

level 0 during the tracking process, where the pose based on the actual feature track, is computed with respect to the closest key-frame. An example of pyramid level 0 features extraction importance, is shown in figure 5.3.6. To address this issue, we take advantage of the decoupled mapping and tracking processes of the RT-PTAM framework and we are still able to track map features expected in the lowest pyramidal level of the current camera frame. This improves tracking quality when moving away from a given feature and makes it possible to navigate over both grass and asphalt.

Moreover, to previous mentioned base RT-PTAM drawbacks (level 0 features), an adding issue, is that the FAST feature detector is implemented with a relatively small patch to decide whether its center pixel is a feature or not. Thus, the initial RT-PTAM framework will detect many features with high spatial frequency. In addition to latter, if we applied local non-maximum suppression in feature selection, i.e. picking the best feature in a local environment, on FAST descriptor, different features may be chosen in consecutive images as the “best” local feature. This results in added unstable feature tracks over key-frames and map-losses, particularly in key-frame pyramid level 0. This problem is slightly improved in final RT-PTAM framework, by using the AGAST feature detector as modified and improved FAST corner detector. In terms of speed and repeatability, AGAST outperforms FAST by implementing improved detector patterns and decision trees. In RT-PTAM, the AGAST variations that are built, are based on 12 pixel diamond pattern or 16 pixel circular pattern. therefore, in RT-PTAM final proposed algorithm, we avoid applying non-maxima suppression for extracting sufficient candidates for the matching procedure.

Furthermore, to enchained the results of RT-PTAM algorithm, besides the above mentioned modifications, we properly choose and utilized the visual sensor. A Down-looking wide angle pinhole sensors solution is used, with the advantage of detecting additional features and particularly peripheral features. We found that in this sensor, the peripheral features are more distant than centric features. Hence, peripheral features usually will be tracked on lower pyramidal levels than centric ones, making this an added case for RT-PTAM to extract features on level 0 during the tracking process. Then we deployed the primary RT-PTAM on monocular down-looking camera, where the camera position is found to increase the overlapping image portion of neighboring key-frames. The outcome of this is that we loosen the heuristics for adding key-frames to the map, and once the UAV has explored a certain region, no more new key-frames will be added within the region boundaries.

Conversely, when exploring new areas the global bundle adjustment can become very expensive, particularly in forward looking cameras. This increased computational complexity and may be reduced by limiting the number of key-frames to a few hundreds on our platform. In this thesis, we implement an alternative solution, in which, many key-frames are retained in RT-PTAM map and still we decreased the computational complexity by changing the bundle adjustment method

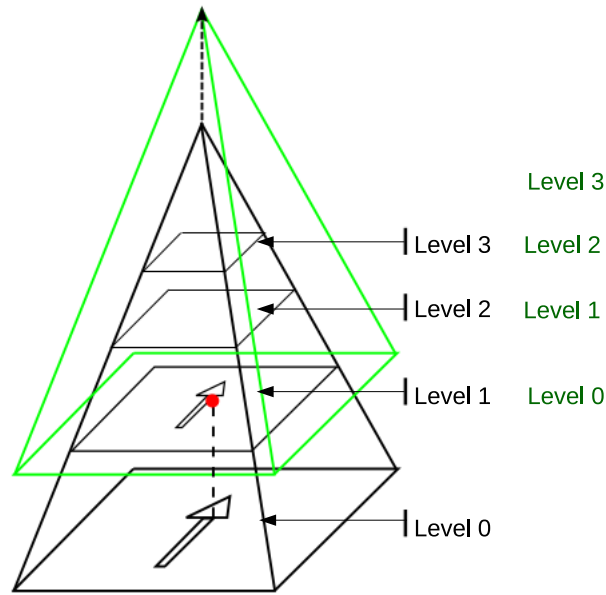


Figure 5.3.6: Example of level 0 features extracting significance. Pyramidal levels 1-3, that RT-PTAM uses for feature detection and map are corresponded to a half-sampled version of the original camera image level 0. Since features on level 0 are unstable, we do not include them into the RT-PTAM map. However RT-PTAM is still tracked them as the camera is moved away from the feature, in particularly a feature (red point) was prior detected on most recent key-frame level 1 (black pyramid), and since camera was moved upwards, the red point is reappeared on the current key-frame (green pyramid) level 0. If level 0 feature extraction is disabled, we are not be able to find this previously mapped point feature.

from local and global, to only local BA. Thus Local BA, only takes the current and nearest four key-frames (including their associated features) into account, during the optimization whereas the global-local method, optimizes all available key-frames and features. However, only using local bundle adjustment can significantly increase the visual sensor pose drifts. Therefore, in optimized RT-PTAM framework both the local only BA and the global-local BA approaches are implemented.

5.4 Horizon Line Navigation Fusion

In chapter 4, a novel Horizon Line detection scheme is described, where the extracted HL, may be used to derive UAV roll and pitch angles. However, a proper navigation framework request full UAV pose representation, e.g. x, y, z , roll ϕ , pitch θ , yaw ω and if is possible scale λ .

The horizon line was previously employed for stability, control and visual navigation of the UAV in [93], [94], [95], [67], [68], [5]. These previous works recovered partial motion parameters, the body rates in [68], or only aimed at providing stability measures for the UAV so that it maintains a level flight. In addition, the above mentioned works shown, detections algorithms of extreme attitudes, or UAV control movement that, achieved with predefined locations, [93]. Furthermore, Error detection in horizon estimation and basic feedback control and stabilization of the MAV was proposed in [95]. In [5], visual HL was computed and an ego-motion navigation was employed. However, the navigation algorithm in [5], was only applied in planar and high attitude navigation, where only 3 parameters need to be estimated (ϕ , θ and forward movement ρ), resulting to a 3 DoF system and excluding free and unconstrained UAV navigation.

Moreover, the previous mentioned methods, are vision only based and partial navigation approaches, since in most of these approaches IMU sensors were not included, and a monocular camera was used, monocular vision only navigation is scale ambiguous. Additionally, pure vision navigation is not robust enough, particularly due to environments existence with low

texture, the lack of high dynamic range sensor during high speed UAV motion, since the sensor's output rate is limited to 100 hz. State estimation based on only one visual sensor is dead-reckoning, which suffers from drifting over time. A partially solution can be loop detection and loop closure, a concept that is very similar to SFM and SLAM methods, discussed in previous sections.

Therefore, a navigation stack is proposed, where HL is applied along with this thesis proposed EKF fusion framework, a IMU sensor and the RT-PTAM framework. The HL pitch and roll angles, namely θ_{HL} and ϕ_{HL} respectively, are used as an sensor update in the EKF Framework. However the update sensor pose, required by the implemented EKF in section 3.3.1 is 6 DoF for a full pose representation. This enforced us to access the additional DoF, though a secondary RT-PTAM SLAM, deployed in the same sensor, where the HL is derived, producing a HL and RT-PTAM fuse framework. Hence a dual visual sensor pose is employed, with two RT-PTAM instances. The main assumption of combining the two frameworks, HL and RT-PTAM, is based on, that both are applied in the same visual sensor, and as an outcome the bias and sensor drifts are the same. In addition both algorithms measurements are providing in the EKF framework, with respect to same sensor frame.

5.4.1 Horizon Fault recovering

The HL is used in this thesis proposed navigation framework, if one of the implicit assumptions of the HL detection algorithm is hold, namely that there will always be a horizon in the images, from the forward looking camera on-board the UAV. However in some cases, we encounter times when no visible horizon appears in the image, if for example, a gust of wind forces the nose of the UAV too far up or down. Thus, an Horizon recovering procedure is used, based on [95]. This Horizon recovering information is additionally helpful in extreme condition of system failures, e.g. sensor failure to detect HL, or alternative sensors errors. There are two valuable sources of information, which we can based on to detect these types of extreme attitudes. The recent appearance of the sky and ground from previous time steps, and the recent location of the horizon line from previous time steps.

Therefore, we propose a workaround to this, if the HL was recently estimated to lie near the bottom of the image, it is logical that a subsequent image without a horizon line is most likely a view of the ground. We can use these two pieces of information to quantitatively determine if the horizon line exists in the image and if not, to determine whether we are looking at the sky or the ground. For any given hypothesized horizon line, we label color *RGB* pixels above the line as sky image part, and pixels below the line as ground image par. Using the covariance and means statistics (Σ_{sky} , Σ_{ground} , μ_{sky} , μ_{ground}) of these two part, we compute the appearance of the sky and ground over a recent time history of the UAV's flight. With each new frame, we comparing the sky and ground covariance model for the current frame with the computed, time-dependent statistical models for sky and ground. If the distributions on either side of the line in the current frame, both appear to be more similar to the known ground distribution, then it would appear that the UAV is pointing towards the ground. Conversely, if they both match the sky better, then we move downward. The statistical models are updated as follows, where the constant a controls how rapidly the models change over time:

$$\begin{aligned}
\Sigma_{sky(t)} &= a\Sigma_{sky(t)} + (1-a)\Sigma_{sky} \\
\Sigma_{ground(t)} &= a\Sigma_{ground(t)} + (1-a)\Sigma_{ground} \\
\mu_{sky(t)} &= a\mu_{sky(t)} + (1-a)\mu_{sky} \\
\mu_{ground(t)} &= a\mu_{ground(t)} + (1-a)\mu_{ground}
\end{aligned} \tag{5.4.1}$$

Then four distance metrics measures are used, D_1 distance to measure the similarity between the recent detected $\Sigma_{sky(t)}$ and previous Σ_{sky} , similar D_2 for recent $\Sigma_{ground(t)}$ and Σ_{ground} . Likewise, the values of D_3 and D_4 are the similarity measures between the current ground region and the sky and ground models from recent frames, respectively.

$$\begin{aligned}
D_1 &= (\boldsymbol{\mu}_{sky} - \boldsymbol{\mu}_{sky(t)}) \boldsymbol{\Sigma}_{sky(t)}^{-1} (\boldsymbol{\mu}_{sky} - \boldsymbol{\mu}_{sky(t)}) + (\boldsymbol{\mu}_{sky} - \boldsymbol{\mu}_{sky(t)}) \boldsymbol{\Sigma}_{sky}^{-1} (\boldsymbol{\mu}_{sky} - \boldsymbol{\mu}_{sky(t)}) \\
D_1 &= (\boldsymbol{\mu}_{sky} - \boldsymbol{\mu}_{ground(t)}) \boldsymbol{\Sigma}_{ground(t)}^{-1} (\boldsymbol{\mu}_{sky} - \boldsymbol{\mu}_{ground(t)}) + (\boldsymbol{\mu}_{sky} - \boldsymbol{\mu}_{ground(t)}) \boldsymbol{\Sigma}_{sky}^{-1} (\boldsymbol{\mu}_{sky} - \boldsymbol{\mu}_{ground(t)}) \\
D_1 &= (\boldsymbol{\mu}_{ground} - \boldsymbol{\mu}_{sky(t)}) \boldsymbol{\Sigma}_{sky(t)}^{-1} (\boldsymbol{\mu}_{ground} - \boldsymbol{\mu}_{sky(t)}) + (\boldsymbol{\mu}_{ground} - \boldsymbol{\mu}_{sky(t)}) \boldsymbol{\Sigma}_{ground}^{-1} (\boldsymbol{\mu}_{ground} - \boldsymbol{\mu}_{sky(t)}) \\
D_1 &= (\boldsymbol{\mu}_{ground} - \boldsymbol{\mu}_{ground(t)}) \boldsymbol{\Sigma}_{ground(t)}^{-1} (\boldsymbol{\mu}_{ground} - \boldsymbol{\mu}_{ground(t)}) + (\boldsymbol{\mu}_{ground} - \boldsymbol{\mu}_{ground(t)}) \boldsymbol{\Sigma}_{ground}^{-1} (\boldsymbol{\mu}_{ground} - \boldsymbol{\mu}_{ground(t)})
\end{aligned} \tag{5.4.2}$$

Then we applied the computed distance metrics as UAV control hypothesis:

UAV extreme attitude hypothesis

1. *UAV in Hovering Point, Detect initial HL.*
2. *if $D_1 < D_2$ and $D_3 > D_4$ then a valid HL is present.*
3. *if $D_1 > D_2$ and $D_3 > D_4$ then UAV is moving to the ground. Next step move to Hovering Point.*
4. *if $D_1 < D_2$ and $D_3 < D_4$ then UAV moving to the sky. Next step move to Hovering Point.*
5. *if $D_1 > D_2$ and $D_3 < D_4$ then UAV moving upside down. Next step move to Hovering Point.*

The determinations in the above table is combined with the past history of the horizon line to decide what action to take. If the current frame is determined to be normal by the validity test (case 1), then the horizon estimate is assumed to be accurate. However these navigation commands, are only applicable in extreme Horizon recovering situations as we already mentions. Smooth UAV navigation is providing only though the full EKF, RT-PTAM and Horizon line framework, since HL estimation is limited to 5-10 hz.

5.5 EKF Integration

The previous presented, RT-PTAM SLAM and HL approaches do not require any changes to the main EKF state. We implement the RT-PTAM and HL measurements as an update measurement pose in the EKF framework, with the camera measuring how the reference frame moves in its coordinate system (see section 3.3.4). The EKF measurements update are proceed, as we presented in section 3.3.1. However, we previous mentioned that a scaling factor λ is necessary as an additional pose state, to provide the estimate scale between metric units and the units of RT-PTAM and HL measurements. Since this arbitrary scaling factor cannot be measured from a single camera, we decided that this scaling factor is initially fixed to an arbitrary value on initialization of RT-PTAM. However, as the sensor moves, λ drifts slightly, and the EKF implementation proposed in this thesis, is compensated by continuously estimating λ within the full error EKF state in equation 3.3.7. The full state is computed in the following equation, where HL is a notation for HL and secondary RT-PTAM in state definitions:

$$\mathbf{x}_{aux} = [p_I^w \quad v_I^w \quad q_I^w \quad b_\omega^T \quad b_a^T \quad \lambda \quad p_I^c \quad q_I^c \quad p_v^w \quad q_v^w \quad HL\lambda \quad HLP_I^c \quad HLq_I^c \quad HLP_v^w \quad HLq_v^w] \tag{5.5.1}$$

,where p_I^c , q_I^c , p_v^w , q_v^w , $HL\lambda$, HLP_I^c , HLq_I^c , HLP_v^w and HLq_v^w are all subject to camera frame (c notation) IMU (I notation) frame, world (w notation) frame and visual sensor (v notation) frame and the relative transformations of them. The state definition and update equations for RT-PTAM are therefore, according to equation 3.3.25:

$$\mathbf{z}_p = \mathbf{p}_s^v = (\mathbf{A}_I^s \cdot \mathbf{A}_I^w \cdot (\mathbf{p}_v^w - \mathbf{p}_I^w) - \mathbf{A}_I^s \cdot \mathbf{p}_I^s) \cdot \lambda + \mathbf{n}_p \tag{5.5.2}$$

$$\mathbf{z}_q = q_{-s}^v = q_{-I}^{s*} \otimes q_{-I}^{w*} \otimes q_{-v}^w \otimes \delta q_n \quad (5.5.3)$$

Equations 5.5.2 and 5.5.3 are hold for the second HL and RT-PTAM visual pose. When a minimal setup (one RT-PTAM instance, one camera) is used, we omit the position state p_v^w . This position may drift, which cannot be avoided in the minimal setup. This drift is slow and does not affect flight stability. In contrast, orientation drift can cause flight instability, we need to include the orientation q_v^w of the vision frame as a state. Furthermore we store the last RT-PTAM pose and median scene depth key-frame, in order to keep RT-PTAM independent to other pose sensors, i.e. IMU. This is also important for automatic initialization after failure mode, where we propagate this stored information to the new initialized map. This way we minimize large jumps in scale and pose after re-initialization.

Chapter 6

Results

Having presented the theory and modular implementation for measurement updates in the preceding sections, we present a detailed evaluation of the navigation framework. In section 6.1 we briefly introduce the navigation system and the hardware set up. The state estimation is evaluated with respect to different noise levels, delay and update rates in this section. We benchmark our EKF framework in section 6.2 first with an “ideal” sensor (motion capture system) where we can control the properties (delay, noise, rate) of its measurements. We proceed presenting the experimental results for Horizon line detection in section 6.3. We conclude the chapter in section 6.4, where we present the results of experimental studies conducted using the full navigation framework.

6.1 System setup

While there are many commercial UAV platforms available for purchase, they are primarily geared towards GPS-based navigation in outdoor environments and thus feature sensor suites and on-board processors that are appropriate to those requirements. Of the platforms geared towards research the most popular are the Ascending Technologies [7] line of UAVs, such as the AscTec Firefly shown in Figure 6.1.1, which is the UAV model used in this thesis. Ascending technologies offer a range of configurations in terms of sensors and on-board computing which allows the UAV to be tailored to a specific research application.

Our setup consists of a Firefly UAV equipped with an Inertial Measurement Unit (IMU), dual monocular down-looking cameras and an on-board computer. Compared to previous UAVs models, the FireFly has an improved vibration damping system for the sensors (camera, Flight Control Unit (FCU)) and can tolerate failure of one rotor, presented in Figure 6.1.2. The FCU features two 60 MHz ARM 7 microprocessors, the Low-Level Processor (LLP) and the High-Level Processor (HLP), as well as an Inertial Measurement Unit (IMU) with linear acceleration sensors and gyroscopes measuring the angular velocity. It is furthermore equipped with a barometric pressure sensor, with a magnetic compass and a GPS sensor. The HLP unit is dedicated to custom code. The code to be executed on the FCU, that has been used during this dissertation, runs on the HLP, i.e, state estimation and control. On the other hand, LLP is a black box, which is in charge of all low level tasks that are required for manual flight operation and handles all hardware interfaces. LLP computes the attitude-data-fusion and flight-control algorithms at an update rate of 1 kHz. Furthermore LLP is used as a safety backup while performing experiments in flight. A distribution of the flight-control units (FCUs) main task between two microprocessors is shown in Figure 6.1.3. As Figure 6.1.3 illustrates, HLP is able to access the LLP on different control levels. On the highest level, it can send way-points. On an intermediate level it can send attitude, yaw-rate and thrust commands. On the lowest command level, it can send individual rotor speed commands to the LLP, which forwards those directly to the motor controllers.

However, tasks such as the state estimation framework from Section 3.3, or the visual navigation framework in Chapter 5, are computationally too demanding to be executed on micro controller hardware. These, are parts of the position control loop, which is very crucial for autonomous navigation with the UAV. Therefore, the firefly UAV is equipped with a on-board computer, which is able to process all critical tasks on-board. The computer is the firefly MasterMind, 1.86 GHz Core2Duo central processing unit, embedded on-board computers from Ascending Technologies. The computer board



Figure 6.1.1: Asctec firefly hex-rotor UAV.

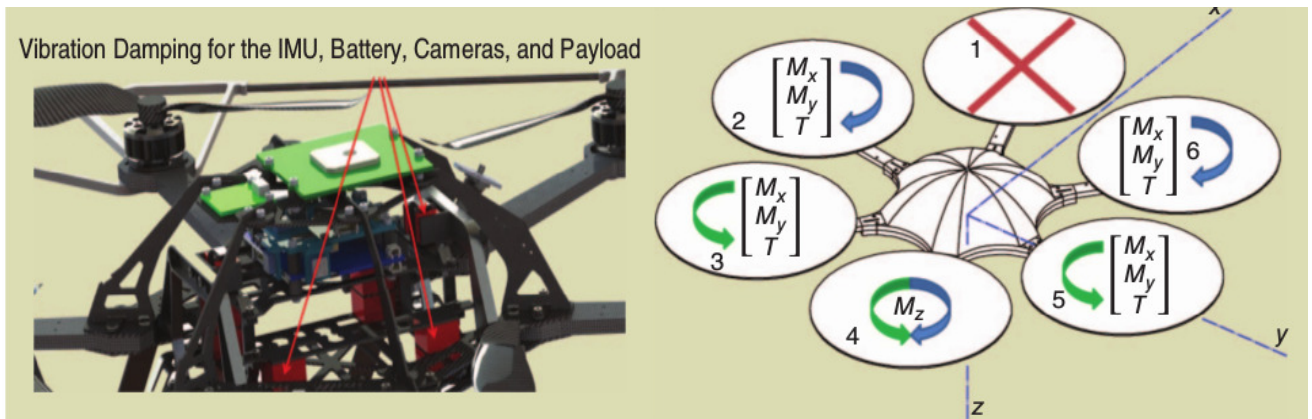


Figure 6.1.2: Asctec firefly internal system design. Left: The firefly CAD model illustrating the vibration damping between the two parts of the frame: the motors and the landing gear are connected to the outer UAV frame, and the inner frame to the IMU, battery, and payload. The silicon dampers are highlighted in red. Right: The redundancy against single-rotor failure is presented. Assuming that motor 1 is failing, motors 2, 3, 5, and 6 are controlled by the thrust command and the roll and pitch output. Motor 4, on the opposite side of the failing motor compensates and controls the yaw momentum by repeatedly changing its direction, [1], [7].

found to provide the computational power to run EKF and visual navigation on-board software, presented on this thesis. Additionally, the on-board computers run standard Ubuntu Linux operating system, and ROS is used as middleware, facilitating the development and deployment of new algorithms.

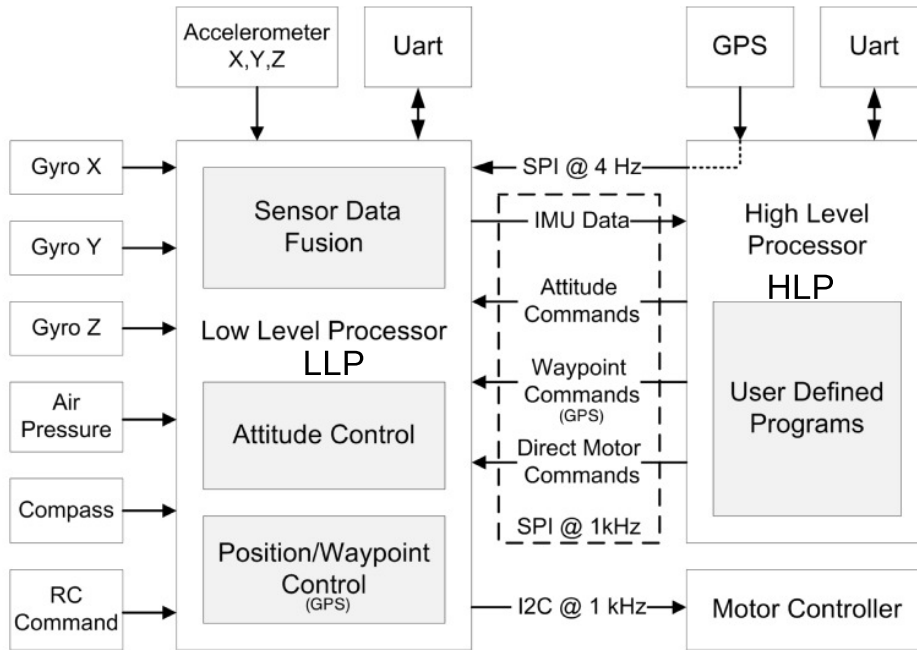


Figure 6.1.3: Overview of the data flow, interfaces and electronic architecture of the firefly, provided by Asctec, [7]. All sensors, except the GPS, are connected to the LLP, which communicates via the serial peripheral interface (SPI) with the HLP and with a I2C to the motor controllers. SPI has the advantage of very low delay, so HLP can access rotor speeds, angular rates, linear acceleration and estimated attitude from the LLP very fast. The LLP position and way-point control module, is implemented with only GPS based control, and will therefore not be used in this work, since we work in GPS-denied environments.

As neither the ROS nor standard Ubuntu operating system are particularly well deployed for real time applications, we chose to distribute the UAV processing tasks according to their computational complexity and real-time requirements, already mentioned this in figures 3.2.1 and 3.3.1 in section 3.2, among the on-board computer, and the HLP. This solution is also proposed and implemented in [58]. In figure 6.1.4, the full system overview of the different processing tasks and how these are distributed and interact with each other, applied in this thesis is presented.

The position controllers shown in section 3.2, as well the prediction part (IMU data) from the state estimation 3.3 are time-critical and require direct and fast access to data from the LLP. These processes are less complex and can therefore be executed on the HLP at a rate of 1 kHz, since no operating system employed on the HLP, and execution is triggered every 1 ms. More complex modules like, the image processing, RT-PTAM visual pose framework and the remaining state estimation, EKF propagation and measurements updates are less time-critical and can therefore be executed on the on board computer without having real-time issues. With this UAV setup, the on-board hardware is in charge of time-critical tasks within the position control loop. The only exception to this is the HL derivation, which runs off-board in the ground station computer. We select this to reduce the computational burden on the on-board computer, as the state estimation, the dual RT-PTAM and the rest processes take most of the computational resources and do not allow for extra higher-level tasks. Additionally, a heavily processing utilized on-board computer can reduce the UAV flight timing and introduce heating and power problems.

Furthermore, solely high-level tasks, such as sending UAV controlling way-points commands or trajectories, HL and RT-PTAM visualizations run on a ground station computer. These tasks are often computationally demanding, but less time critical such that a WiFi link can be employed to communicate with the ground station. In the following sections, we present results on simulation and real data. Simulations were performed for the full 3 D case with 6 DoF for each vehicle, while real experiments were carried out using the firefly UAV.

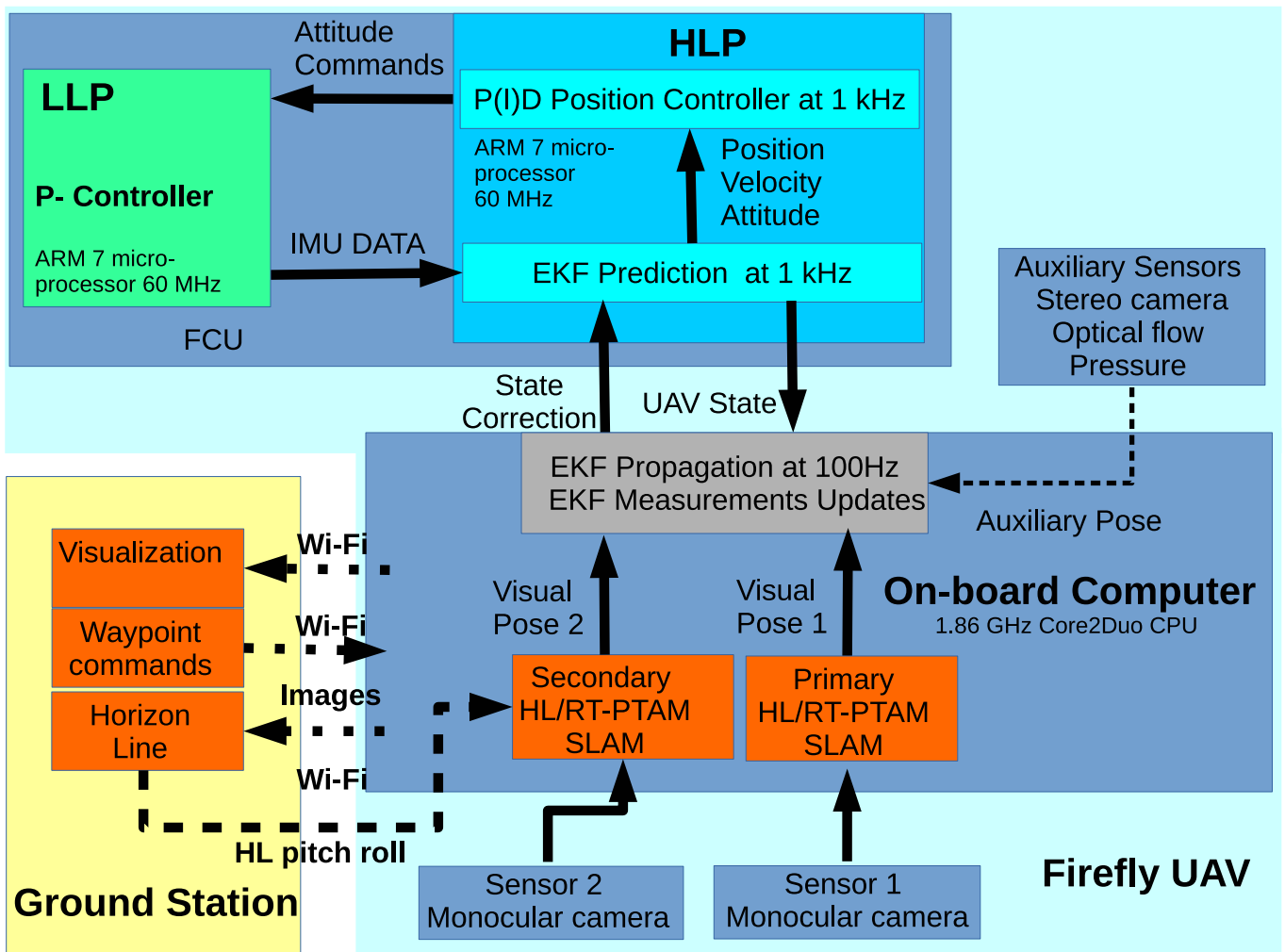


Figure 6.1.4: Thesis system overview: the computational demanding tasks are executed on the on-board computer, while the less intensive and time critical parts are executed on the high-level micro processor at a rate of 1 kHz. Way-point commands are sent from the ground station to the on-board computer, which forwards them to the position controller. The ground station and on-board computer communicate over a Wi-Fi connection. Note that all critical parts necessary to keep the UAV airborne run entirely on board and do not rely on the Wi-Fi link.

6.2 State Estimation Results

In this section, we present results on real datasets. Simulations were performed for the full 3D case with 6 DoF with dual visual sensors. While the improvement of the estimated state is one benefit of including additional sensors, the capability of seamless switching between the elements of the sensor suite online provides additional fail-safety and versatility. One such case is the change from single vision-based estimation to dual-based global position; relevant results are presented here.

Two types of datasets were employed in our experiments. The first batch of datasets, namely EuRoC micro aerial vehicle dataset, [116], was recorded in a large machine hall shown in Figure 6.2.1 and aims at testing visual-inertial motion estimation algorithms or SLAM frameworks. For ground truth, a 3D position was provided by a laser tracker and a motion capture system. A second batch of dataset was captured from indoors vicon room [117] and contains 6D pose ground truth from a Vicon motion capture system and an accurate 3D point cloud.

Figures 6.2.2 to 6.2.3 illustrate the position and attitude results from the datasets. Furthermore pose estimation is



Figure 6.2.1: Representative image of the machine hall, where the EuRoC micro aerial vehicle dataset was collected.

evaluated, using dual single camera set-up. This sensor configuration was further in depth tested with a variation of : (a) single noise sensor (NSS) (b) single sensor (TS) with optimized EKF noise parameters, and (c) a dual sensor set up (NDS), which combines the noisy sensor NSS and normal, not optimized TS one. The latter highlights the proposed dual visual sensor fusion framework in this work. Therefore a statistic evaluation is performed on the previous mentioned datasets and the results are shown in 6.2.4.

However, despite careful design and execution of the experiments, we are aware of different issues which pose additional challenges for processing and limit achievable accuracy when comparing to ground-truth. The visual-inertial sensor employs an automatic exposure control that is independent for both cameras. This resulted in different shutter times and in turn in different image brightness's, rendering pose matching and feature tracking more challenging. Some of the EuRoC datasets exhibit very dynamic motions, which are known to deteriorate the measurement accuracy of the laser tracking device. The numbers reported by the manufacturer may be overly optimistic for these events, which complicates the interpretation of ground-truth comparisons for highly accurate visual odometry approaches. The effect on sections with less dynamic motion, particularly the start and end of each dataset, is assumed to be negligible.

In addition The accuracy of the synchronization between sensor data and motion ground truth is limited by the fact that both sources were recorded on different systems and that device timestamps were unavailable for the Vicon system [116]. However, as the results illustrated the dual EKF framework achieved pose, which is close to ground truth measurements, in contrast, the case where a sensor is noisy the UAV state is totally diverge is shown in figure 6.2.6. Therefore, this issue is mitigated by additionally estimating the UAV trajectory, particularly with a second visual sensor. The effect of noisy sensor is assumed to be negligible. The proposed EKF is shown to be able to follow close by the ground truth results.

In figures 6.2.4 and 6.2.5, we summon the relative error statistics from the previous datasets. The mean error is computed with respect to the ground truth values. In addition the root mean error square are reported, in particularly in table 6.1. The yaw error values, presented in figures 6.2.4 and 6.2.5, and the yaw root mean error square are diverged the most from the ground truth case. This is due to more aggressive and dynamic motions exhibit in EuRoC datasets. Furthermore in EuRoC dataset, a forward stereo and slightly down looking sensor system was applied, and as we previously mentioned monocular single type of visual sensors are unable to compute and provide the EKF with absolute height and scene scale.

	NSS	TS	NDS
Position X	1.83	0.04	0.06
Position Y	1.68	0.03	0.04
Position Z	4.47	0.04	0.0901
Roll	2.61	1.97	2.115
Pitch	31.15	1.79	1.82
Yaw	109.05	6.84	6.89

Table 6.1: Root mean square error for single noisy sensor (NSS), optimized single sensor (TS) and dual sensor set up.

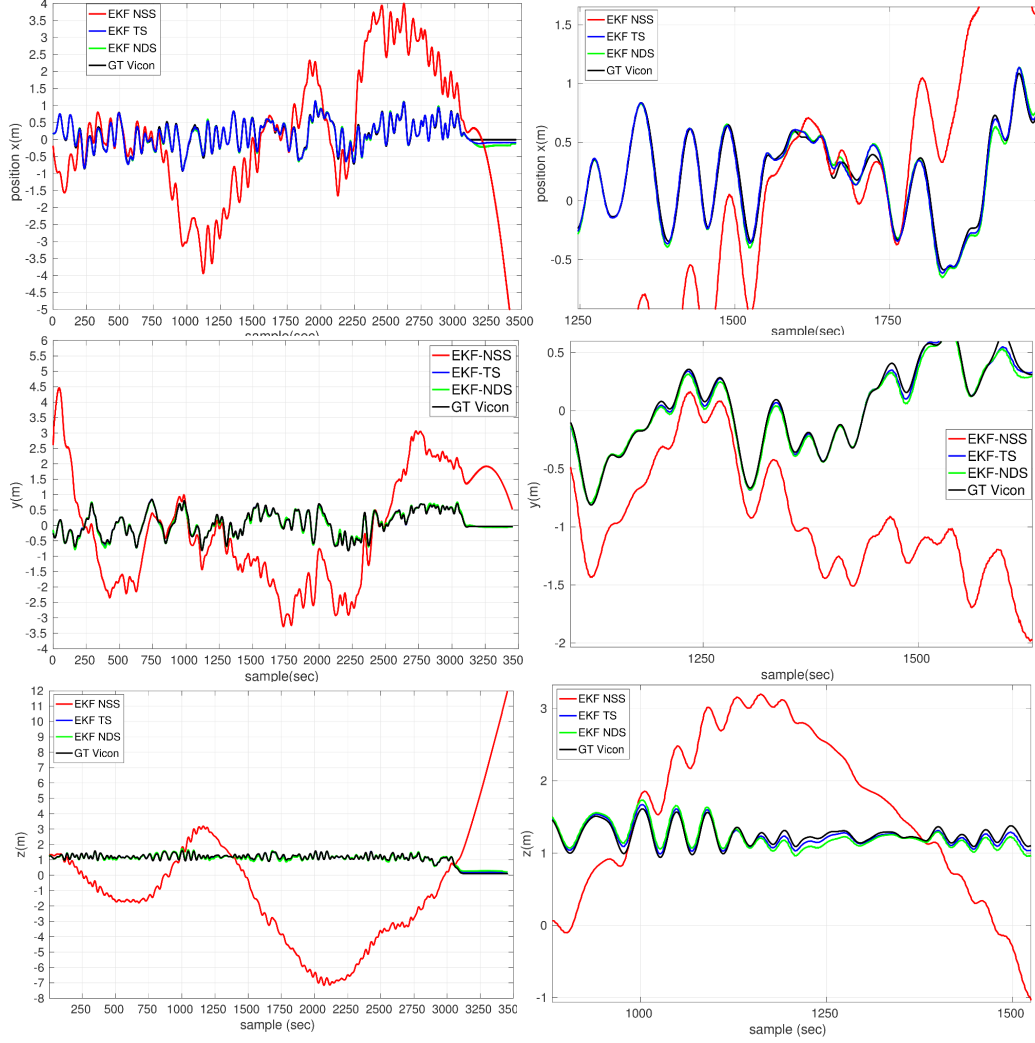


Figure 6.2.2: Position estimation comparison of NSS (red line), TS (blue line) and NDS (green line) sensors configurations and datasets position ground truth (black line); Left column: Full frame rate deployment; Right column: Specific frames comparison.

Therefore, we use an empiric scale to initialize the EKF and include scale in its state estimation, where $H_{RT-PTAM}$ is the height measure based on RT-PTAM SLAM and H_{metric} is a test metric height at which we are holding the UAV in hovering mode. However, the scale metric was not included in EuRoC dataset, and to best of our knowledge no dataset reported it. The test metric height was arbitrary chosen from running the dataset and compute the UAV position, in comparison with the surrounding environment.

$$\lambda_{init} = H_{RT-PTAM}/H_{metric} \quad (6.2.1)$$

Finally, in figure 6.2.6 a 3D trajectory navigation is shown, which is based on the above mentioned datasets. The presented trajectory depict navigation in dynamic and robust scenes. The proposed in this thesis framework succeeded in accurately estimated the 3D UAV position and considerably close to ground truth data. The Horizon Line results are summarized in the following section.

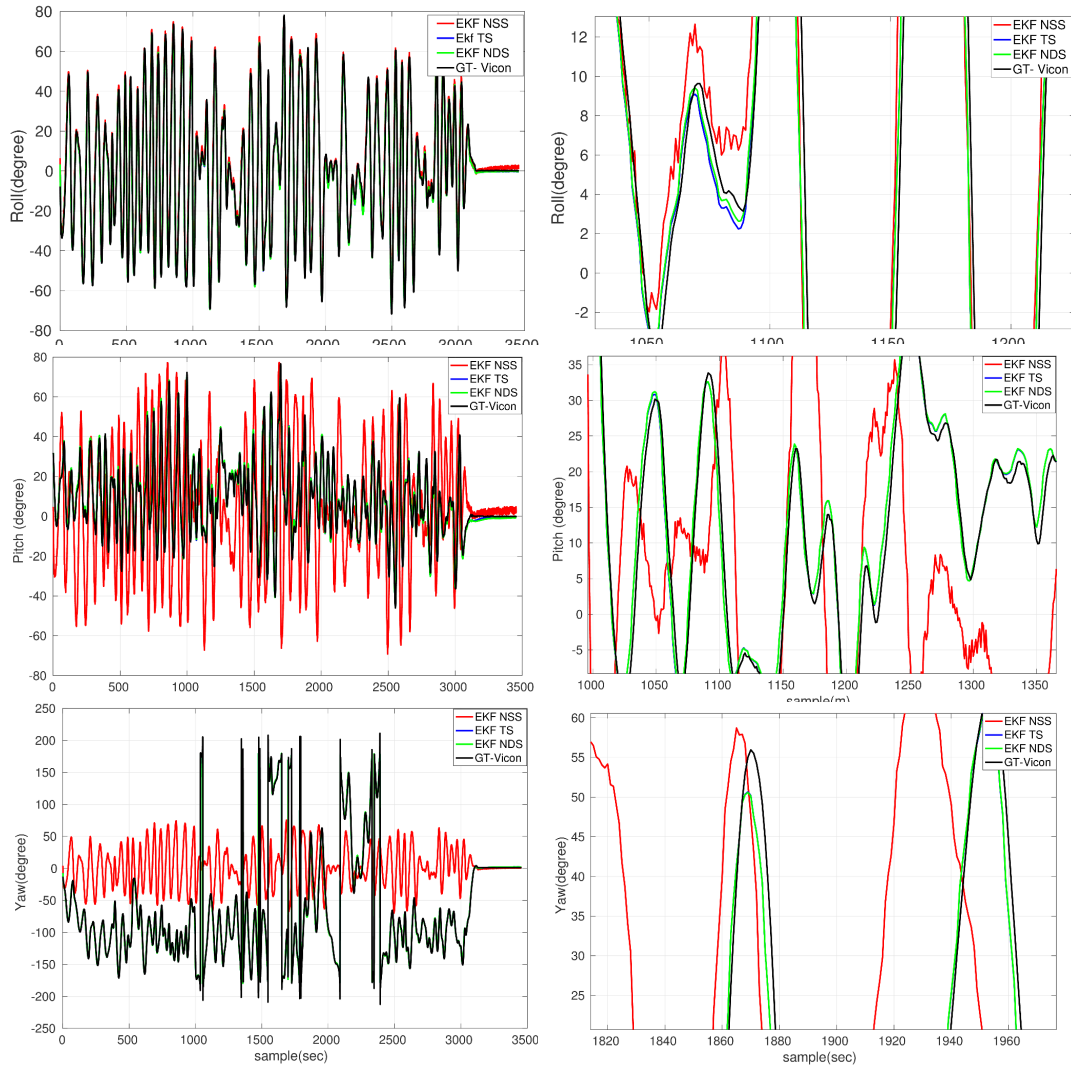


Figure 6.2.3: Attitude estimation comparison of NSS (red line), TS (blue line) and NDS (green line) sensors configurations and datasets position ground truth (black line); Left column: Full frame rate deployment; Right column: Specific frames comparison.

6.3 Horizon Line Results

In this section both qualitative and quantitative experimental results are presented in order to demonstrate the effectiveness of the proposed HL detection framework. A dataset of 300 RGB images has been employed for conducting experimentation with the proposed framework. Moreover, an implementation on an actual UAV facilitated further evaluation under real world conditions. All the aforementioned real-time experiments were performed at the facilities of the Computer Vision and Robotics lab at FORTH both outdoors and indoors in a flying space room. In the rest of this section, the obtained results are reported and contrasted to competitive ones.

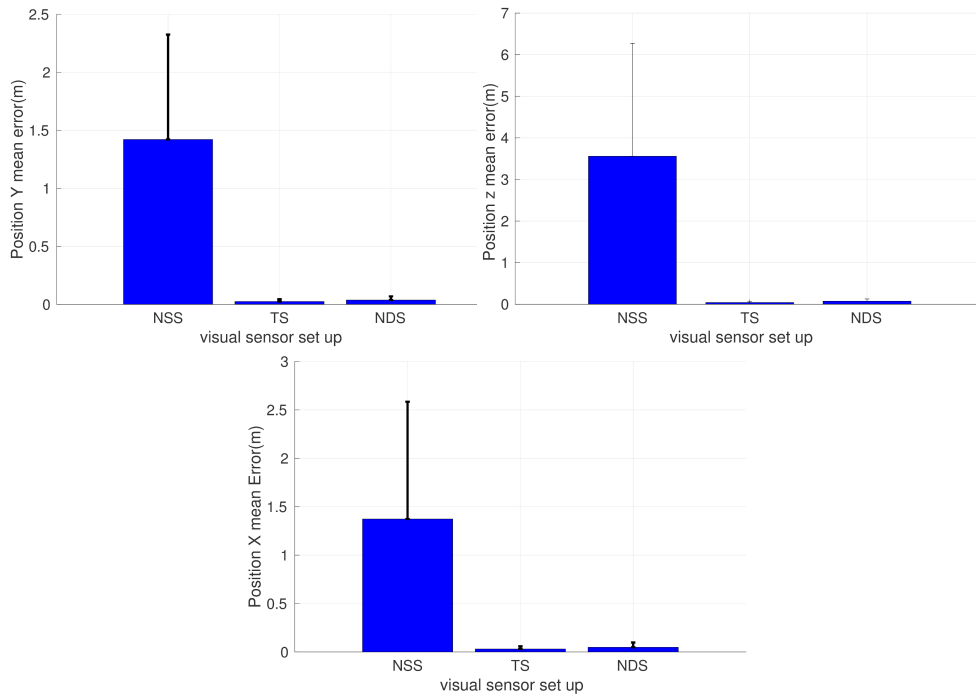


Figure 6.2.4: Position estimation mean error of NSS, TS, NDS sensors configurations with respect to datasets ground truth.

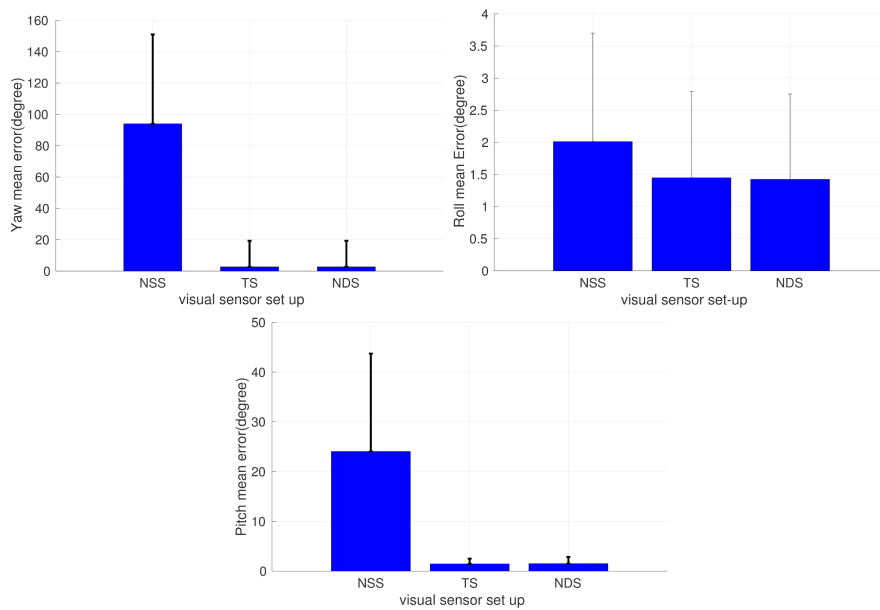


Figure 6.2.5: Attitude estimation mean error of NSS, TS, NDS sensors configurations with respect to datasets ground truth.

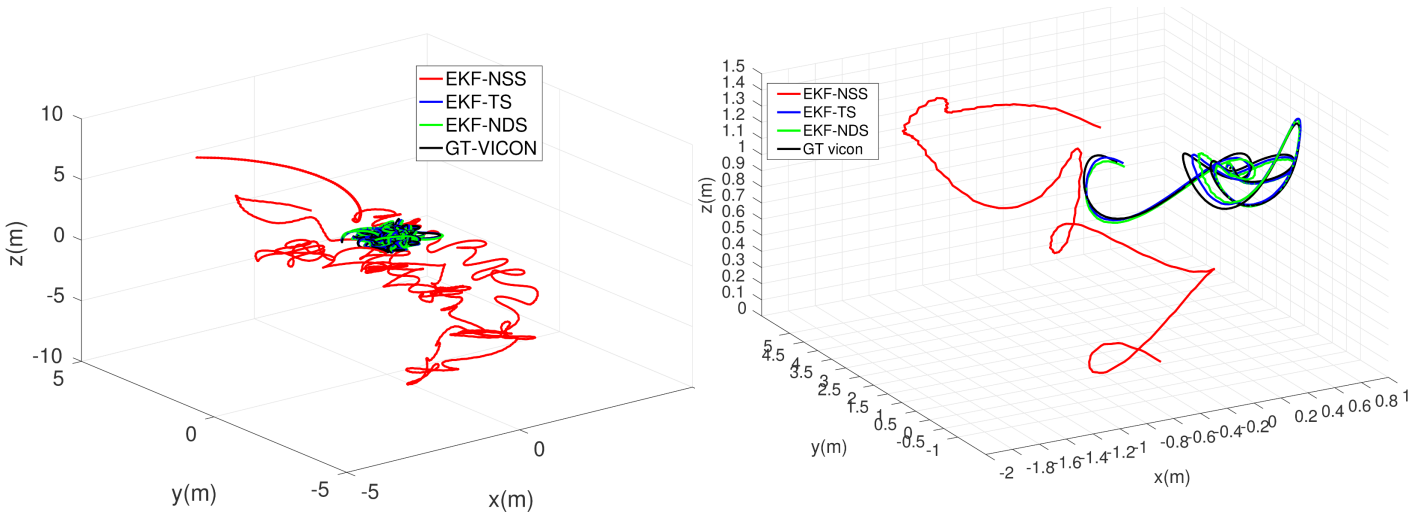


Figure 6.2.6: A 3D navigation trajectory comparison of NSS (red line), TS (blue line) and NDS (green line) sensors configurations and datasets ground truth (black line) trajectory.

6.3.1 Qualitative Results

A dataset consisting of 300 RGB images has been compiled to facilitate in-depth experimentation. It contains images from three different sources: (a) a set of images from the recently published Horizon Lines in the Wild, HLW, dataset [118], (b) Images collected from the web, termed as WHS, (c) a set of images captured with a mobile camera mounted on a moving vehicle, termed as VHS and (d) images captured with a mobile sensor mounted on a moving UAV, termed as UHS. The content and quality of the images in the dataset are analogous to images commonly acquired by UAV-cameras during flight. Accordingly, brightness, blur, contrast, resolution and horizon orientation vary considerably across the images as it is the usual case in UAV captured data.

In the following, we present HL detection results on sample images from the above dataset. For comparison purposes the results obtained via the application of the proposed framework, termed PSO-HL, are contrasted to the ones obtained with the Hough-HL [75]. Figures 6.3.1, 6.3.2 and 6.3.3 present indicative results for various environment scenes. In all cases, we used a value $\sigma = 0.1$ for the Gaussian filter in the Canny operator, a step size of $\rho = 0.1$ and $\theta = 0.1$ for the Hough transform parameters, and the initial PSO particles and number of evolutions were both set to 50. In addition, the dataset images were sub-sampled to 40×40 pixels, which reduce the processing time and enable real time acquisition in 10 Hz.

Figure 6.3.1 illustrates HL detection in cases of rather standard and simplified images. The image at the top row is a typical sky-sea image, whereas the image at the bottom is a sky-ground one. In the former case, color variations, i.e. sky-sea color differences, play an important role in HL detection, while in the latter case, texture variations are more informative for HL detection. In both cases, the two tested detectors accurately estimated the correct HL, as verified by a human-operator. Case presented in Figure 6.3.1 and refer to image from set WHS.

The following Figures 6.3.2 and 6.3.3 present results on images acquired from the mobile vehicle-camera (image set VHS). The presented images depict HL detection in difficult landscape scenes. Mountains, sky, sea, and trees along with blurring effects, low brightness, and different orientations formulate challenging images for HL detectors. Still, in all cases the proposed PSO-HL succeeded in accurately detecting HL and considerably outperformed Hough-HL.

Before we proceed with the quantitative results, we present in figure 6.3.4, two specific images obtained on-board the UAV. The first issue is based on the use of Logitech c920 web camera, which have rolling shutter, and distort images, figure 6.3.4 left, particularly when having large motion and flying low. Hence, it is rather difficult to be useful for any UAV pose estimation. In addition, to the best of our knowledge, web cameras send compressed video stream over the USB bus. That is another quality loss and explains the high cpu usage, e.g. for de-compression. To overcome these drawbacks the fastest camera shutter speeds are applied, along with high illumination experimental conditions. Furthermore, the use of high contrast and low-frequent features is succeed to mitigate both the issue with rolling shutter and quality loss due to

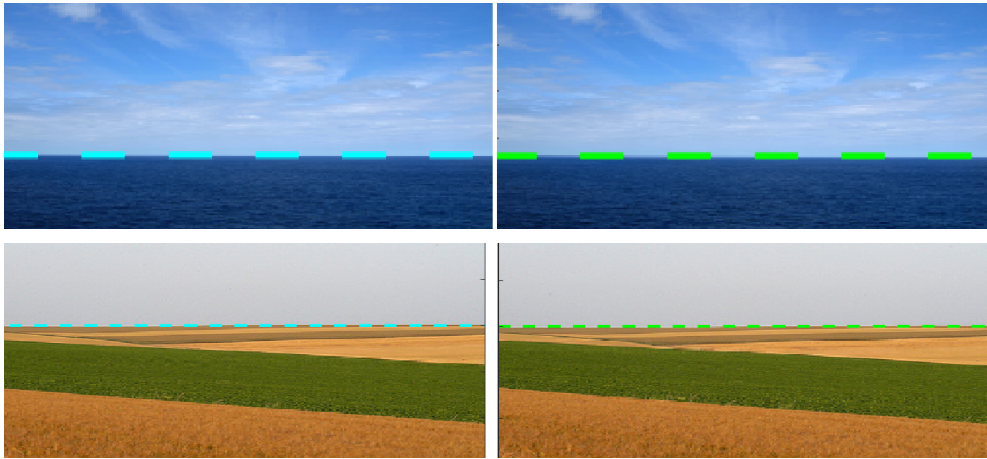


Figure 6.3.1: HL detection. Top row: sky-sea image; bottom row: sky-ground image. Left column: Hough-HL; right column: PSO-HL.

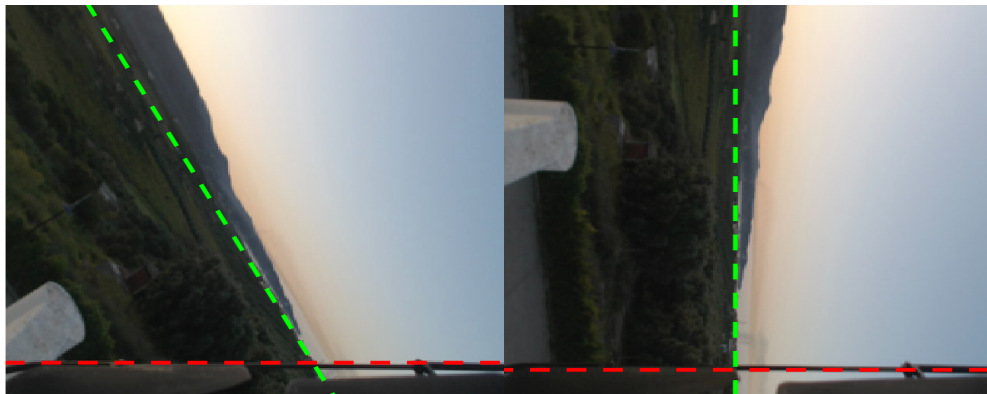


Figure 6.3.2: HL detection in a challenging image. Left column: Wide angle orientation; right column: 90 deg orientation, Hough-HL (red) and PSO-HL (green).

compression. Reducing the image to the lowest level of resolution is a good idea in this camera case. The second issue, appealing in figure 6.3.4, is based on acquiring IMU and camera pitch and roll measurements in different coordinate frames, which is discussed in sensor fusion section 3.3. A transformation is employed to resolve this and enable HL framework and IMU comparison, which is shown below. The transformation between the two images is described by a homography transformation matrix, as presented in map based localization section 5.2.1. Furthermore, the HL was correctly derived by HL-PSO in both images, in contrast with the Hough-HL.

6.3.2 Quantitative Results

Two sets of experiments were conducted to quantitatively assess the proposed framework. In the first case, the above mentioned dataset of 300 RGB images has been utilized. The detection results of PSO-HL and Hough-HL were acquired and visually verified by a trained human-operator. The latter Hough-HL method is based on HL estimation, where the strongest peak or longest Hough line is selected as the candidate HL [75], [71], [119], [120], [121], [68]. Furthermore, we extend the evaluation of HL detection framework, with comparing the proposed method with a HL acquisition using segmentation of camera images into sky and non-sky regions based on machine learning decision trees [73]. Machine learning is employed to learn the appearance of the classes of sky and non-sky in images. This method, namely HL-Tree, includes a variety of features, i.e. mean value, standard deviation, smoothness, moments, uniformity and texture.

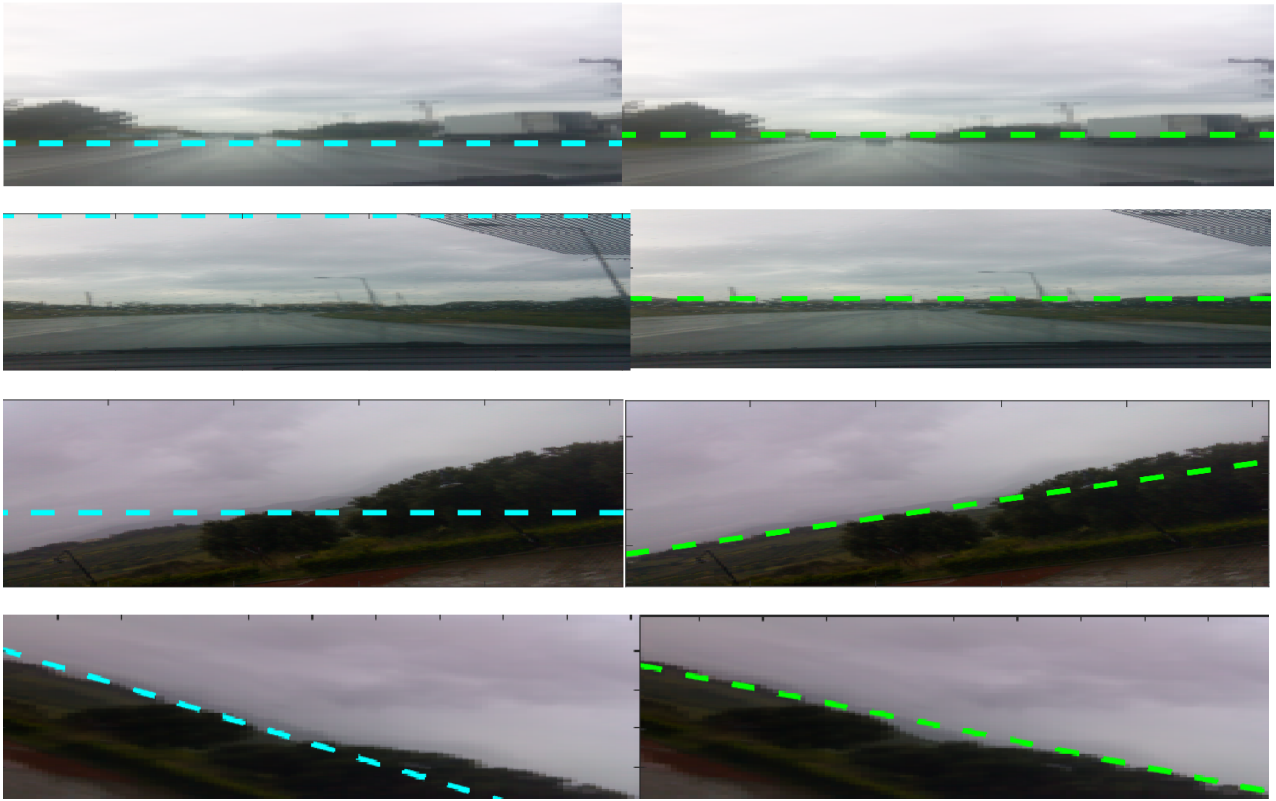


Figure 6.3.3: HL detection in the case of a moving camera, Left column: Hough-HL; right column: PSO-HL.

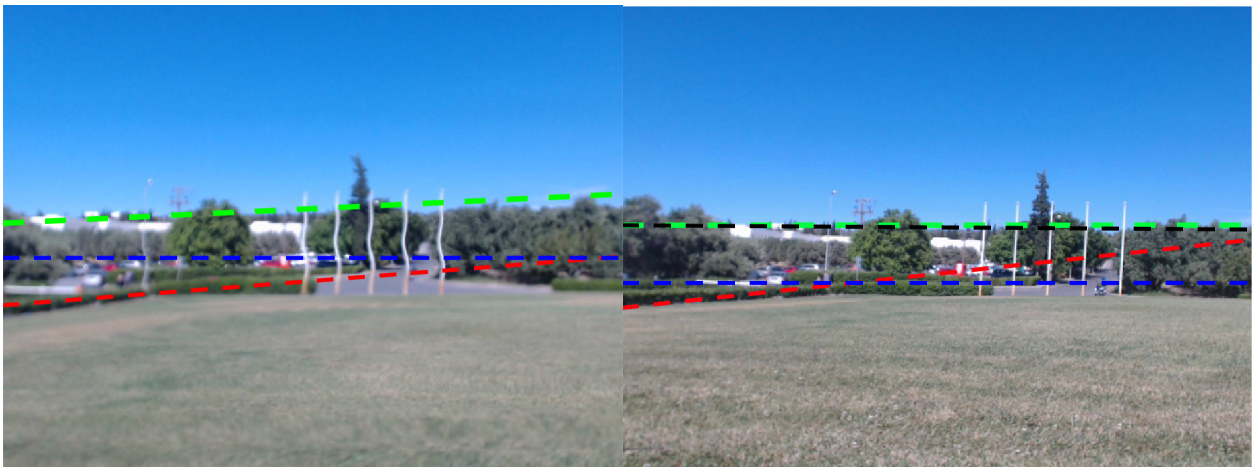


Figure 6.3.4: HL detection in the case of a UAV on-board moving camera, Left column: Camera distortion based on logitech C920 rolling shutter ; right column: IMU measurements frame are translated to HL measurements frame PSO-HL, Hough-HL (red), PSO-HL(green), IMU-untranslated(blue) and IMU-translated(black)

In addition, we compared our method against the one presented in [118] that attempt HL detection in HLW dataset. The method was initially based on detecting horizontal vanishing points and the zenith vanishing point in man-made environment in a non Manhattan-world assumption [74]. The authors proposed a set of horizon line candidates and score each based on the vanishing points it contains, where a global image context was extracted with a deep convolutional network, to constrain the set of candidate HLs under consideration. For each candidate, a vanishing points was identified

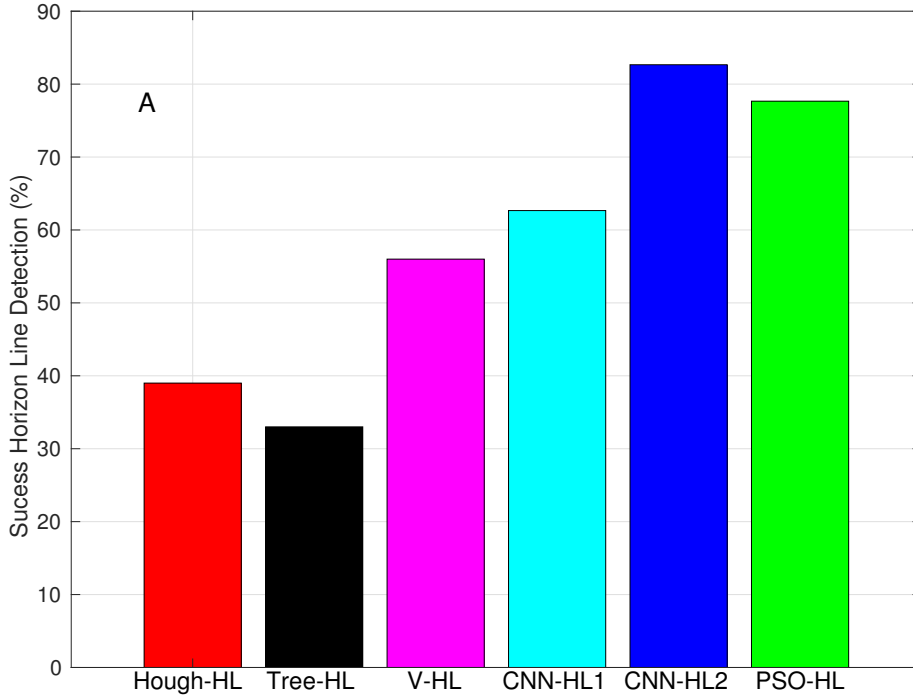


Figure 6.3.5: HL detection accuracy. *Hough – HL*: (red bar) 39%, *HL-Tree*: (black bar) 33%; *V-HL*: (magenta bar) 56%, *CNN-HL1* (single CNN):(cyan bar) 62.66%; *CNN-HL2* (multiple CNN):(blue bar) **82.66%**, and *PSO – HL*:(green bar) 77.66%.

by solving a discrete-continuous optimization problem. The final score for each candidate line is based on the consistency of the lines in the image with the selected vanishing points. The aboved method was further investigated and optimized using HLW dataset, with the application of convolutional neural networks for directly estimating the horizon line from raw pixel intensities, without requiring any explicit geometric constraints or other special cues. Hence, the CNN was directly estimated the horizon line location. We include both methods to PSO-HL comparison, e.g. the vanishing point (V-HL) and the cnn (CNN-HL).

Figure 6.3.5 presents the HL detection accuracy for the tested methods. In all cases, the results were deemed as correct or erroneous depending on whether the obtained HL visually agreed with the one suggested by the operator. As can be observed, PSO-HL significantly outperformed Hough-HL, HL-Tree and V-HL with a detection accuracy of **77.66%** contrasted to the low figure of **39%**, **33%** and the **56%** achieved by Hough-HL HL-Tree and V-HL approaches respectively. The HL-Tree and V-HL methods suffers from not be able to even process a large percentage of dataset, 16% and 12% for each one. The CNN-HL method was closely by V-HL, with a **62.66%** for a single deep Learning network and **82.66%** for multiple utilized CNN networks. However, the latter is significantly slower than the PSO-HL method, which is crucial in real time applications.

Finally, an experiment using an Asctec Firefly UAV (see Figure 6.3.6) in a realistic environment was performed to assess correct estimation of ϕ_{roll} and θ_{pitch} . Images were captured with the on-board RGB-camera at 20 FPS, 640×480 resolution, 74° field of view, and a focal length f of $3.67mm$. HL detection and computation of ϕ_{roll} and θ_{pitch} are performed on a remote PC exchanging data with the Firefly over WIFI; all implementations are based on the Robot Operating System (ROS). Given that the mentioned experiments were conducted outdoors, no ground-truth information is available. To this end, we utilized a complementary filter [1] to provide accurate ϕ_{roll} and θ_{pitch} estimates from the on-board IMU data, running on LLP micro-processor. The latter estimates, are employed as valid substitutes for the missing ground-truth data. The obtained results for a 20s outdoors flight are shown in Figures 6.3.7 and 6.3.8. Figure 6.3.7 illustrates the ϕ_{roll} estimation results, whereas Figure 6.3.8 presents the θ_{pitch} results. In both cases, angle estimation results are presented for IMU-data, PSO-HL, and Hough-HL. The ϕ_{roll} root mean square error for PSO-HL over IMU is **1.55 deg**, compared to



Figure 6.3.6: Outdoor experiments UAV with dual camera set-up; Firefly.

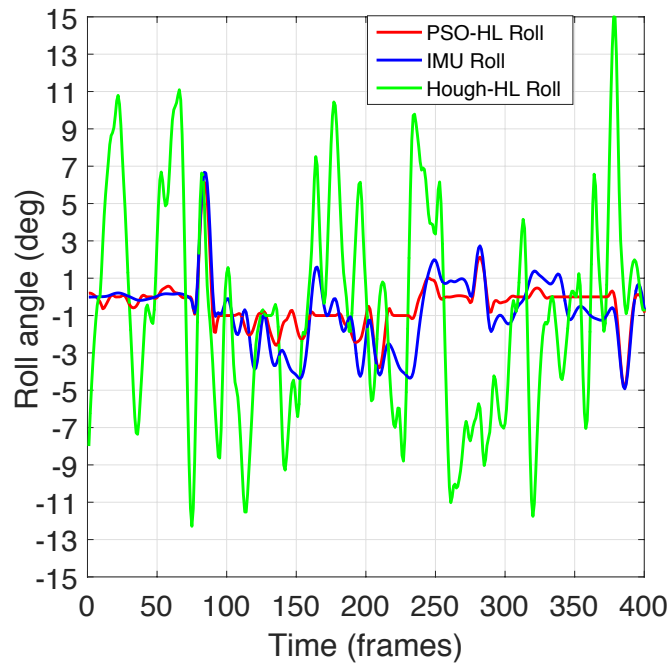


Figure 6.3.7: Roll angle estimation: IMU (blue line), PSO-HL (red line), Hough-HL (green line).

8.02 deg for Hough-HL. The θ_{pitch} root mean square error for PSO-HL over IMU is **1.04 deg**, compared to **2.90 deg** for Hough-HL. Evidently, the estimation of both angles was carried out with improved accuracy by the proposed optimization methodology.

6.4 UAV SLAM Navigation

In this section we present the results of experimental studies conducted using the navigation framework presented in figure 6.1.4, which main part are consisted by the EKF fusion framework and the RT-PTAM slam. Therefore in this section experimental results are presented in order to demonstrate the effectiveness of the proposed navigation framework.

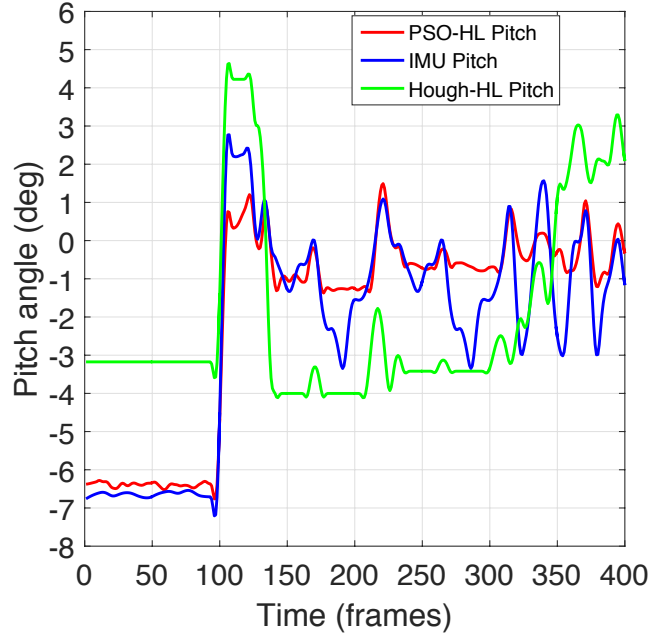


Figure 6.3.8: Pitch angle estimation: IMU (Blue line), PSO-HL (Red line), Hough-HL (Green line).

6.4.1 RT-PTAM SLAM Results

We evaluate the accuracy of our RT-SLAM in the 11 sequences of the KITTI dataset, [122]. The dataset provides synchronized video sequences with rectified high resolution images. We compare our method thoroughly with state-of-the-art stereo VO and SLAM methods, both feature-based and direct method on KITTI. The dataset is very challenging, because the low frame rate in combination with fast navigation speeds leads to large inter-frame motions up to 2.8 m per frame. This greatly limits the amount of possible feature correspondences. Moreover, dynamic motions from passing vehicles, bicycles or pedestrians, that have great impact on the performance of visual odometry systems, are included frequently. We compare the performance of our semi-direct method with four state-of-the-art methods for visual odometry and SLAM.

We compare our method to LSD-SLAM, [111], SVO and SVO2 [123] SLAM, ORB-SLAM2 [124] and PTAM/S-PTAM, [114], a stereo version of PTAM SLAM, which are currently the state-of-the-art direct and feature-based stereo VO methods respectively. We present the mean error results in figure 6.4.1.

The RT-PTAM SLAM is outperformed both SVO and LSD SLAMs. Along with PTAM, the RT-PTAM achieved the best result in position Z error, e.g. translation motion. However in position x and in 2D estimation, both the ORB-SLAM2 and SVO2 SLAM are performed better. This is mainly due to two reasons: (a) KITTI datasets are low frame-rate, hence very fast inter-frame motion, (b) strong rotations without any translation during turns, yield relatively small field of view, particularly for forward facing camera motion.

Furthermore, in figure, 6.4.2, the trajectory generated by the above mentioned dataset is shown. In addition the ground truth measurements are provided. The RT-SLAM performs relatively well in straight navigation parts, however as we previously mentioned, under rotation movement the estimation error is increased. We conclude the comparison in Table 6.2, where the above mentioned SLAM, root mean square error is reported for x, z and 2D x-z position estimation.

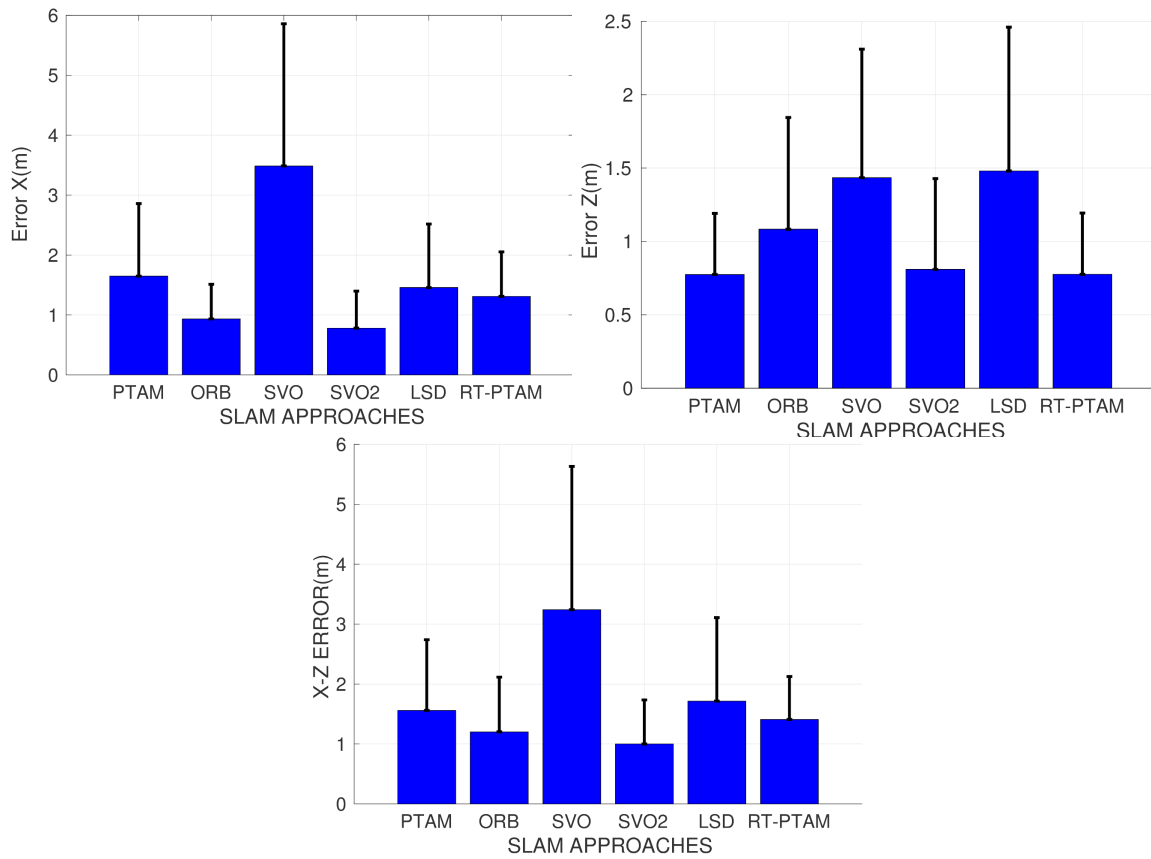


Figure 6.4.1: KITTI datasets trajectory mean error of LSD-SLAM ORB-SLAM2, SVO, SVO2, PTAM and PTAM2 with respect to datasets ground truth. Upper-Left: Position x error, Upper right: Position z error and Bottom: 2D position error.

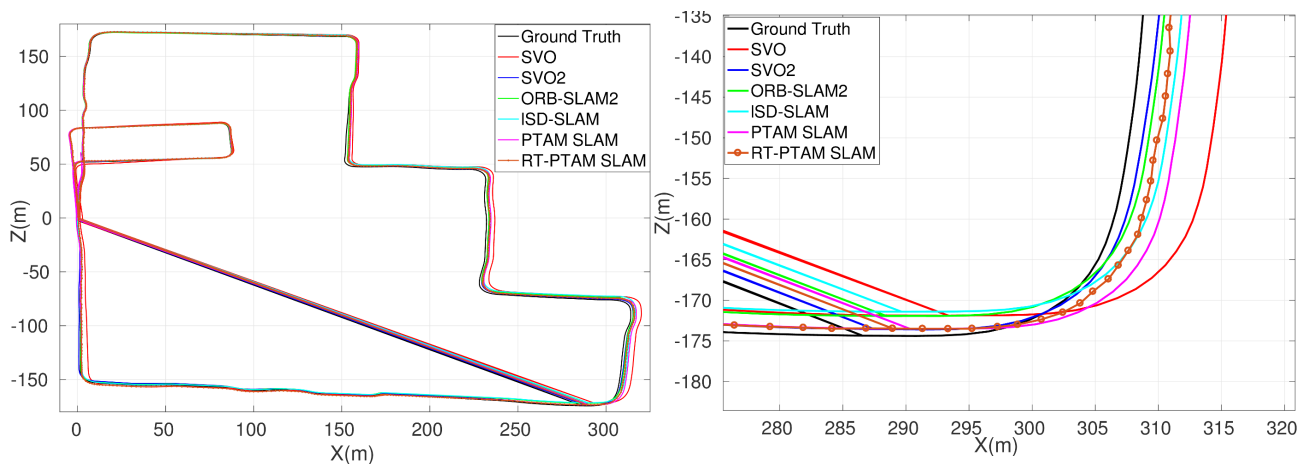


Figure 6.4.2: KITTI datasets navigation trajectory based on SLAM algorithms. LSD-SLAM (cyan line), ORB-SLAM2 (green line), SVO and SVO2 (red and blue line respectively), PTAM (magenta line) and PTAM2 (orange dashed line) and ground truth (black line). Left: Full navigation trajectory, Upper right: Specific navigation frames comparison

	XZ Position	X Position	Y Position
PTAM	1.38	0.87	2.04
SVO	2.84	1.68	4.21
SVO2	0.8778	1.0186	0.99
LSD	1.5626	1.77	1.80
ORB2	1.06	1.32	1.09
RT-PTAM	1.11	0.88	1.50

Table 6.2: KITTI datasets, root mean square error for RT-PTAM, PTAM, SVO, SVO2, LSD, ORB2 SLAM with respect to the datasets ground truth.

6.4.2 Navigation Framework Results

We conducted our field experiments at a large sized area in which a large building, several containers, smaller buildings, trees grass and asphalt and stone path, provided a realistic scenario, as shown in Figure 6.4.3.



Figure 6.4.3: Representation of outdoors navigation experiments in complex realistic field area, acquired with on-board UAV visual sensors.

In our outdoors experiments, we deployed the navigation stack, proposed in this thesis. Hence, an experiment using an Asctec Firefly UAV (see Figure 6.3.6) in a realistic environment was performed. Images were captured, with dual monocular on-board cameras: (a) A down looking camera (mvBlueFox MLC200) is a 1/3" CMOS monochrome camera capturing images with 752×480 pixels at up to 90 FPS. The 8 bit monochrome camera can distinguish between 2^8 gray levels. The camera was equipped with a global shutter type. (b) A forward looking RGB-camera(Logitech C920), was using a rolling shutter type, with resolution 640×480 , 74° field of view, running up to 30 FPS.

The high level implementation is presented in figure 6.1.4, and is based on the Robot Operating System (ROS). Given that the mentioned experiments were conducted outdoors, no ground-truth information is available. To this end, we utilized the ORB-SLAM2 to provide accurate UAV pose estimates from the on-board sensor data, running on Ground station. The latter estimates, were employed as valid substitutes for the missing ground-truth data. However, ORB-SLAM2 was not sufficient initialize, with forward looking on-board camera, in contrast the proposed RT-PTAM was properly initialized. Therefore, UAV ground-truth state was only acquired from bottom looking camera, with respect to ORB-SLAM2. Before we proceed with the experimental results, we discuss the initialization process of ours framework. The procedure is started with a initial $P_0 = [x_0, y_0, z_0, \phi_0, \theta_0, \psi_0]^T$ position. In outdoor experiments cases the UAV start position was the ground-zero position, where this position was required for horizon line detection.

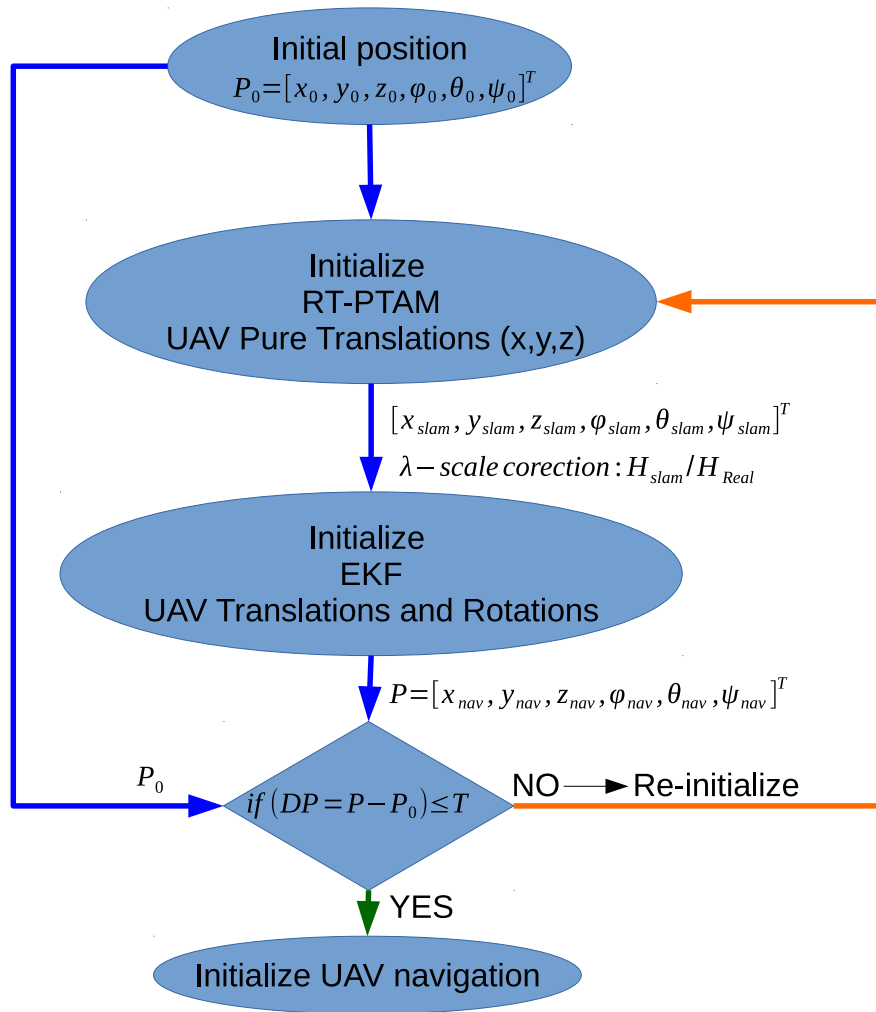


Figure 6.4.4: Overall UAV navigation initialization process.

Hence, RT-PTAM was initialized with pure translations along with horizon line and camera acquisitions processes. The RT-PTAM SLAM output is a 6 DoF position vector, which acts as EKF framework update measurement, as we previously

mentioned. Additionally, the scale correction was included, as given by equation 6.2.1. Then the EKF was initialized with both rotations and translations to excited each axis, until EKF bias were converged. Both RT-PTAM SLAM translations and EKF excitations were performed in a high feature place, which further improved initialization process. This achieved during a small UAV random walk navigation. To evaluate the framework initialization a simple criterion is proposed, where the difference between the initial UAV start position P_0 and the UAV initial position estimated by the framework $P = [x, y, z, \phi, \theta, \psi]^T$, DP is less that a threshold T , as illustrated by figure 6.4.4. The threshold T values in the experiments were 0.1m for x , y and z motion and 1 deg for $roll(\phi)$, $pitch(\theta)$ and $yaw(\psi)$ angles. It is rather obvious that, to achieved this initialization criteria comparison, the UAV after brief initialization maneuvers, was returned to the start position. If the DP was less that the T , the navigation mission was then performed.

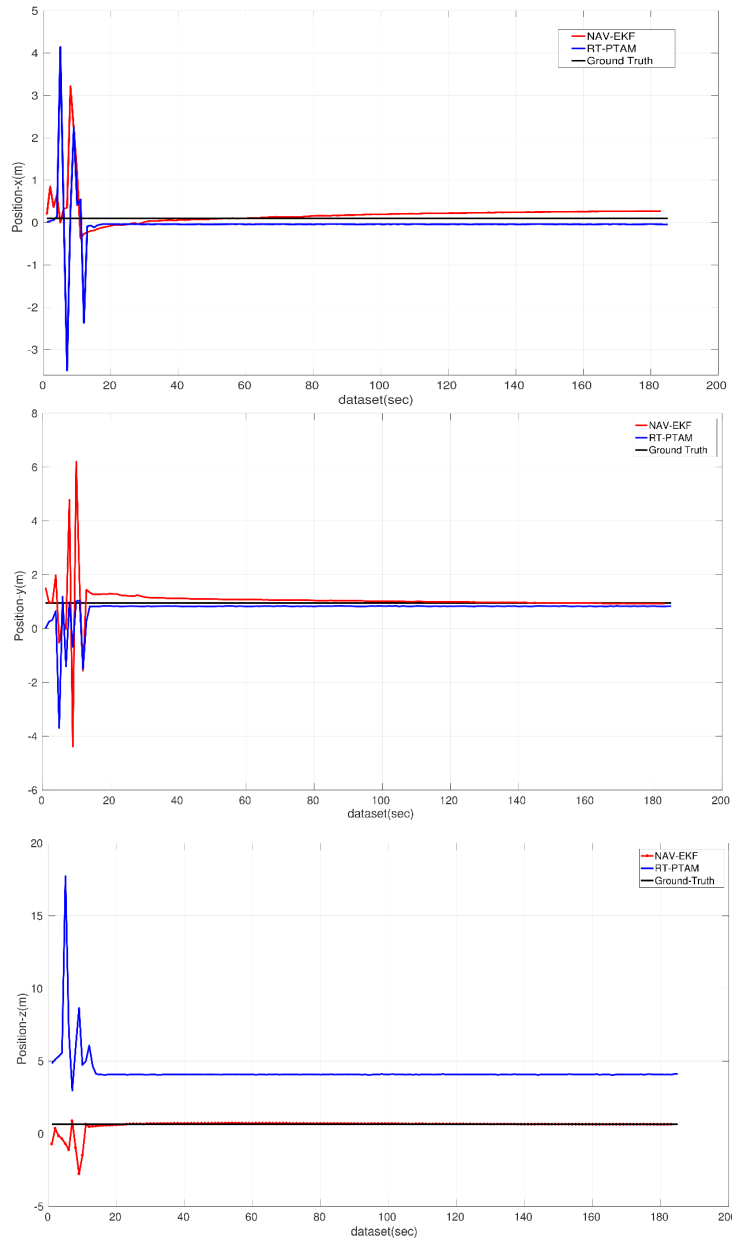


Figure 6.4.5: Initialization process of RT-PTAM(blue line), NAV-EKF(red line) and Ground Truth(black line) in x , y and z axes. A dataset of 180 sec was used. Upper: Position- x (m), Mid: Position- y (m) and Bottom: Position- z (m).

In figure 6.4.5, an initialization result of both the RT-SLAM and the overall visual framework (NAV-EKF) is presented. The UAV start position $[x_0, y_0, z_0]$ was $[0.2, 0.95, 0.66]$, which was used as the experiment ground truth. Both the RT-SLAM and the NAV-EKF framework were able to converge successfully in the start position x_0 and y_0 after the previously mentioned a initialization processable. The z_0 position was not successfully estimated by RT-SLAM alone, however the full framework was able to achieved it. This RT-SLAM false estimation was mainly based to scale and absolute height drawback of monocular SLAM approaches.

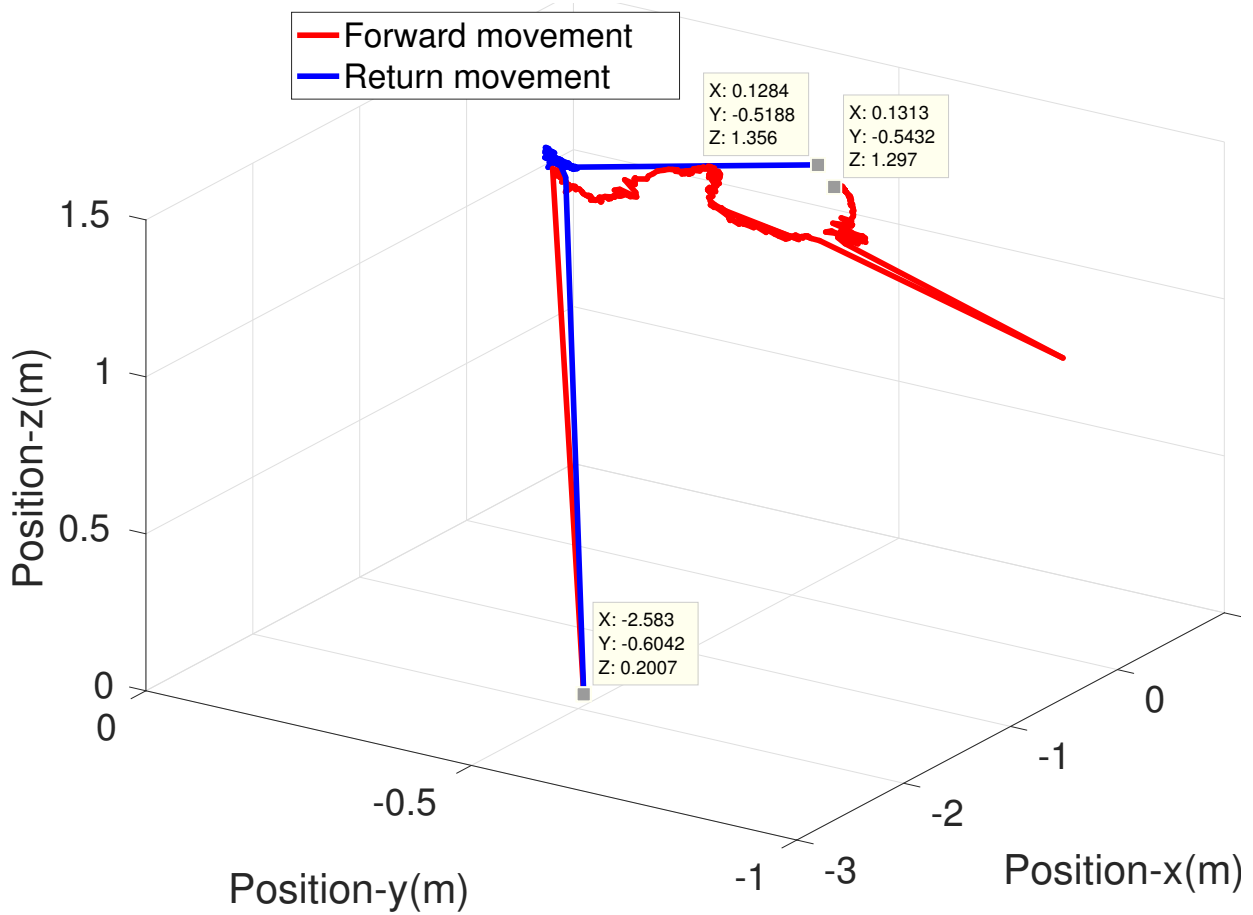


Figure 6.4.6: UAV homing experiment. Red line: Forward navigation trajectory. Blue line: Return trajectory.

When a successful initialization was established, we proceed with the navigation experiments. A UAV homing navigation experimental result is presented in figure 6.4.6. The navigation starting point was $x = 0.1313m$, $y = -0.5432m$ and $z = 1.297m$ and the UAV return-end point was $x = 0.1284m$, $y = -0.5188m$ and $z = 1.356m$. The navigation framework was able to closely estimate and return to the start position. The absolute start and end position differences was $Dx = 0.003m$, $Dy = -0.02m$ and $Dz = 0.06m$. The UAV performed trajectory was up to $3m$ in x , $1m$ in y and $1.5m$ to z axes. Random rotations were introduced during the navigation.

Figure 6.4.7 illustrates, the above presented UAV homing experiment visualization, which was implemented in the ground station. Red and yellow circles in figure 6.4.7 are represented the detect corner points from the RT-PTAM SLAM. The UAV poses (red arrows) and the produced point cloud (red-black squares) were integrated in ROS visualizer as figure 6.4.7 is shown.

Finally, quantitative experimental results from 20 real time navigation experiments are presented in table 6.3, where the RT-SLAM, PTAM and NAV full framework are compared with respect to ORB-SLAM2 acquired results, which was used as experimental ground truth. The estimates x , y , z , $roll$, $pitch$ and yaw root mean square error is reported in table 6.3 with respect to the ground truth.

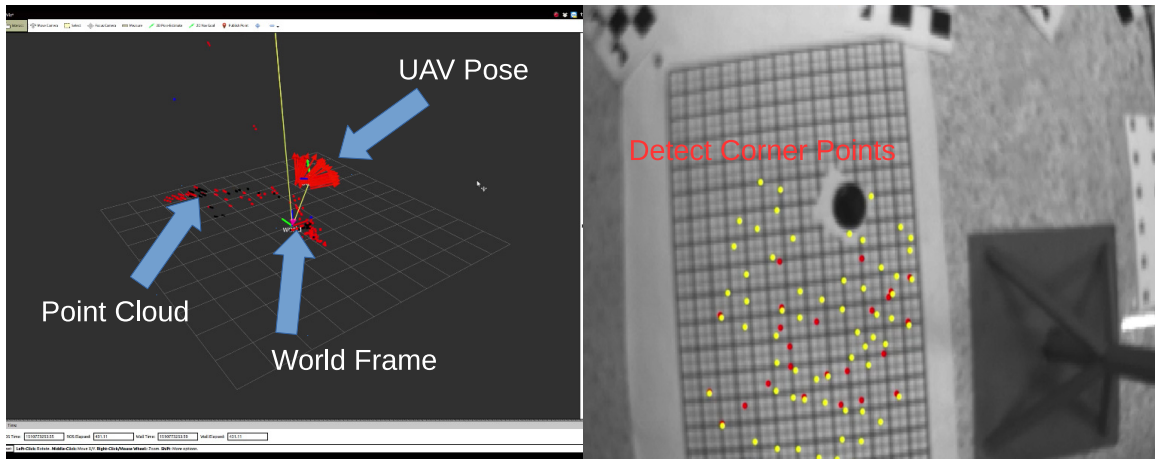


Figure 6.4.7: UAV homing experiment visualizer. Left column: ROS RVIZ ; Right column: RT-PTAM visualizer.

	PTAM	RT-PTAM	NAV Framework
Position X	0.31	0.28	0.21
Position Y	0.68	0.26	0.23
Position Z	0.58	0.23	0.15
Roll	0.54	0.17	0.11
Pitch	0.63	0.12	0.11
Yaw	0.19	0.10	0.06

Table 6.3: Visual Navigation, Root mean square error for PTAM, RT-PTAM and Navigation framework, with respect to ground truth (ORB-SLAM).

In table 6.3, PTAM SLAM was less successful to estimate the state than RT-SLAM. The NAV full framework was benefit from the EKF and dual RT-SLAM and horizon line fusion and further improved the estimation in comparison to the ground truth (Orb-SLAM2). We note at this point that the Orb-SLAM2 was not able to initialize in most of the indoors datasets and in some of the outdoors datasets, where a planar or wall initialization was requested.

Chapter 7

Discussion and Conclusions

In this thesis, we have presented a scalable framework for visual navigation of aerial vehicles in the wild. The presented algorithms facilitate robust autonomous flight in outdoor environments based on a dual monocular camera set up. We examine and propose several methods from different disciplines, namely control, perception or state estimation and visual localization, enabling a UAV to perform autonomous flights. We reduce the human operator part in the navigation loop, in providing high-level commands whereas being completely outside the time-critical low level stabilization. In the current, final thesis chapter, we summarize the results and contributions and place them into perspective of future work.

Overall, in carrying out this work we employed a multitude of techniques and methods from various scientific fields. Specific strengths and knowledge of shortcomings were utilized towards enabling autonomous flights with the employed sensory system. Precise software development was also an important part of this dissertation, giving rise to the controller, EKF fusion system, and the RT-PTAM VSLAM framework.

Our work started with the introduction of the camera sensor (cf. chapter 2) as an abstracted sensor yielding a 6DoF pose. We revisited the underlying principles of robotic vision and structure from Motion, required later, such as camera model, efficient bundle adjustment and feature matching. The latter facilitated the formulation of a map-based approach in visual navigation, described in chapter 5.

The abstraction from a bearing sensor to a 6DoF pose sensor brought different advantages and disadvantages to the overall system. While in existing approaches features are observed in contrast to a reference frame, we implement the camera to be a standalone 6DoF pose sensor, as detailed in chapter 3. Accordingly, we explicitly introduce a world reference frame in addition to the camera's own reference frame. This highlights an advantage of our approach, where the world reference-frame is aligned with gravity to ensure proper UAV navigation. Moreover, we overcome the drift states between the vision frame and the world reference frame, which are hidden in contemporary approaches that employ a camera sensor for navigation.

The separate world reference frame also allows incorporation of any additional sensors, and renders our fusion approach (chapter 3) into a versatile multi-sensor system. We explicitly consider state-drift with an extensive analysis of the behavior of our improved visual pose estimation framework (chapter 3). Furthermore, we discuss a proper choice of the process-model utilized within our indirect Extended Kalman Filter (EKF) framework, and introduce the basic principles of our EKF framework. The framework is designed to facilitate implementation of dual visual poses (cameras), along with auxiliary sensors, i.e. pressure and optical flow. We present a modular integration of this complex state estimation framework, which is entirely executed on constrained on-board hardware. Individual processing tasks are distributed according to their real-time and computational demands, yielding a distributed EKF architecture that allows UAV state prediction at cycles up to 1 kHz. Finally, the continuous online estimation of calibration and visual drift states enables long term-navigation and eliminates the need of tedious permission re-calibration procedures.

An analysis of the necessary dynamics and differential flatness of UAV (chapter 3) showed potential abstractions in order to employ position and trajectory controllers. We addressed the issue of position control of a UAV by presenting a relatively conventional, yet robust and reliable position control approach. Furthermore, we presented a control approach based on dynamic inversion, particularly a two-loop design, where we eliminated the need of the typically used attitude loop, minimizing the computational complexity and improving bandwidth. We also presented an outer control loop to directly command angular rates and thrust, by using the states given by our state fusion framework. We formulated a method

for control allocation for a hex-rotor, and we presented an approach to estimate vehicle parameters, which are otherwise tedious to determine. The controller was evaluated during navigation mission, from simple hovering to fast dynamic moving.

In chapter 4 of this thesis, a novel HL detection framework was presented and experimentally assessed. The framework is based on the Edge-Hough line detector, with a further optimization step carried out by a PSO algorithm. The latter utilizes an objective function that effectively fuses three different features, namely color, texture and SIFT. For computational efficiency BOW is employed to readily calculate the objective function for candidate HLs and provide a compact way for representing image content and fusing the comparison of different cues. In addition, the detected HL was utilized for computing two important UAV navigation quantities, namely roll and pitch angles.

Having a functional and tested state estimation sub-system, a navigation framework was introduced that consists of two main modules, the visual pose derived from RT-PTAM and horizon line and the following precise fusion in state estimator framework (chapter 5). Modifications were utilised in RT-PTAM, such that we were able to employ it not only in small workspaces, but also in challenging outdoor scenarios.

The mentioned framework, consisting of the state estimation part and RT-PTAM SLAM, has been thoroughly tested using state of the art, publicly available databases, and under a variety of challenges, exhibiting robustness in the presence of wind gusts, difficult light conditions causing saturated images, and large scale changes of outdoors flight. Successful tracking in such scenarios demonstrates the capabilities of our robust sensor fusion and estimation approach.

In addition, in horizon line detection, an extensive experimentation with images acquired from a ground camera and from a flying UAV revealed the accuracy and robustness of the proposed approach. The obtained results demonstrate correct HL detection and superior performance compared to the Hough HL detector. Moreover, the proposed scheme showed robustness to low resolution and blurred input images. Finally, the derived roll and pitch angles were contrasted to the ones provides by the UAV's onboard IMU validating our approach.

The overall results in this thesis demonstrate that vision-based UAV navigation is possible as long as we are only concerned about the local consistency of our environment, i.e. position and global yaw drift can be neglected for the task at hand. We also showed the full capabilities of our modular sensor-fusion approach by switching between single and dual visual sensor setups. Furthermore we introduced additional state noises and erroneous measurements. We tested the different configurations in order to analyze the states which are affected. The advantages of multi-fusion systems were shown, with disadvantages regarding the extra weight and computational burden for the UAV.

The computational capacity of the on-board hardware (onboard computer and HLP) restrict us to simple and efficient control approaches, especially since we aim at running the controller on the HLP. This unfortunately excludes Model Predictive Control (MPC) based approaches, which require a few milliseconds for computation even on a desktop computer. SLAM framework was further restrained, due to the use of a rolling shutter camera, introducing image distortion and feature error detection. However RT-PTAM was able to initialize and to produce results comparable to ORB-SLAM2. Even in the case of our robust tightly-coupled sensor fusion framework, errors may occur, particularity in respect to z-axis and yaw movement. This is a drawback of monocular vision, where height and metric scale is not available. However, such sensors drifts are slowly integrated in UAV state and can be totally neglected, as we showed with false pose estimation.

In this dissertation we proposed and implemented several parts towards accomplishing UAVs that perform complex missions autonomously. However, to achieve this goal, significant research and development is still needed; the promising results presented here may constitute fundamental building blocks in this endeavor.

From a broader perspective, it seems plausible in the near future to employ UAVs for real-time applications. These scenarios require robustification of the approaches employed, such that these are able to handle varying environmental conditions, as well as uncertainty and noise in the environment. These conditions are particularly an issue for estimation approaches, and probably need most attention in future works. Technological advances in other fields are required as well. One of the biggest limitations is battery life, which only allows for autonomy of 20-30 minutes in the best case.

However, to enable fully autonomous UAVs in commercial applications, robustness remains an issue of utmost importance. In particular, it will be difficult to certify robustness of vision-based algorithms. Therefore, significant engineering efforts are still required, which should become the task of the industry that is applying this technology. On the other hand, the work in this thesis can be continued along several exciting research paths.

There is a major discrepancy between available camera technology and the sensors commonly used for SLAM research. While high resolution and high frame-rate cameras are a commodity today, research on SLAM is still focused on low

resolution monochrome cameras. For instance, there is a trade-off between using direct pixel tracking in high frame-rate cameras versus computing robust descriptors for wide-baseline data association in low frame-rate cameras. Moreover, there exist a variety of completely novel cameras such as event-based neuromorphic vision sensors [125], where the output is a low-latency stream of “events” that is generated when single pixels perceive a brightness change, rather than a periodic stream of frames. It remains an open question which sensor should be selected, how they should be mounted on a robot, and which algorithms should be applied such that the robot can achieve a certain task with the required reliance and energy usage, in a given environment.

Furthermore, in future work, HL detection can be drastically improved, possibly via the inclusion of image vanishing points and continuous HL tracking. In addition, HL is readily amenable to be employed in real UAV navigation scenarios. Next to that, parallel consideration and possible redesign and implementation of RT-SLAM and HL may lead to increased HL detection frequency. Moreover, to simplify the EKF filter utilization and the quaternion conversion, a rotation matrix based approach [126] can be applied in future steps. A full real time deployment of pressure sensor integration can be a solution to scaling drawback of monocular vision in outdoor navigation.

Finally, in order to enable smooth and reliable navigation, a global 3D navigation framework with a path planing module is required [127], [128], along with the inclusion of collision avoidance. Hence, the navigation state estimation needs to be extended with the capability to build a 3D map of the environment and re-localize with respect to this map. Accordingly, future work should undoubtedly explore higher level tasks as major building blocks in novel UAV applications. Such tasks include environment recognition [129], [130], human-robot interaction [131] and relevant AI approaches with emphasis in deep learning methods and deep reinforcement learning [132]. Combining the above with lidar-based pose estimation is expected to be a unique challenge, still with far-reaching effects.

Bibliography

- [1] M. W. Achtelik, M. Achtelik, Y. Brunet, M. Chli, S. Chatzichristofis, J. D. Decotignie, K. M. Doth, F. Fraundorfer, L. Kneip, D. Gurdan, L. Heng, E. Kosmatopoulos, L. Doitsidis, G. H. Lee, S. Lynen, A. Martinelli, L. Meier, M. Pollefeys, D. Piguet, A. Renzaglia, D. Scaramuzza, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, “Sfly: Swarm of micro flying robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2649–2650.
- [2] M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, “Path planning for motion dependent state estimation on micro aerial vehicles,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3926–3932.
- [3] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, “Monocular vision for long-term micro aerial vehicle state estimation: A compendium,” *Journal of Field Robotics*, vol. 30, no. 5, p. 803–831, 2013.
- [4] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys, *Large Scale Visual Geo-Localization of Images in Mountainous Terrain*. Springer Berlin Heidelberg, 2012, pp. 517–530.
- [5] O. Oreifej, N. Lobo, and M. Shah, “Horizon constraint for unambiguous uav navigation in planar scenes,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1159–1165.
- [6] D. Nister, “An efficient solution to the five-point relative pose problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.
- [7] *Ascending Technologies*, <http://www.asctec.de>.
- [8] R. R. Murphy, S. Tadokoro, and A. Kleiner, *Disaster Robotics*, B. Siciliano and O. Khatib, Eds. Springer International Publishing, 2016.
- [9] E. Altug, J. P. Ostrowski, and R. Mahony, “Control of a quadrotor helicopter using visual feedback,” in *IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, 2002, pp. 72–77.
- [10] S. Klose, J. Wang, M. Achtelik, G. Panin, F. Holzapfel, and A. Knoll, “Markerless, vision-assisted flight control of a quadcopter,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 5712–5717.
- [11] D. Mellinger, N. Michael, and V. Kumar, “Trajectory generation and control for precise aggressive maneuvers with quadrotors,” vol. 31, no. 5, 2012, pp. 664–674.
- [12] M. Hehn and R. D’Andrea, “Quadcopter trajectory generation and control,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1485 – 1491.
- [13] R. Ritz, M. W. Müller, M. Hehn, and R. D’Andrea, “Cooperative quadcopter ball throwing and catching. ,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4972–4978.
- [14] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, “Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3056–3063.

- [15] T. Lee, M. Leoky, and N. H. McClamroch, “Geometric tracking control of a quadrotor uav on $se(3)$,” in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.
- [16] C. Richter, A. Bry, and N. Roy, *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*. Springer International Publishing, 2016.
- [17] J. Wang, T. Bierling, L. Höcht, F. Holzapfel, S. Klose, and A. Knoll, *Novel Dynamic Inversion Architecture Design for Quadcopter Control*. Springer Berlin Heidelberg, 2011.
- [18] M. W. Mueller and R. D’Andrea, “A model predictive controller for quadcopter state interception,” in *European Control Conference (ECC)*, 2013, pp. 1383–1389.
- [19] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, “Fast nonlinear model predictive control for multicopter attitude tracking on $so(3)$,” in *IEEE Conference on Control Applications (CCA)*, 2015, pp. 1160–1166.
- [20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.
- [21] S. Shen, N. Michael, and V. Kumar, “Autonomous indoor 3d exploration with a micro-aerial vehicle,” in *IEEE International Conference on Robotics and Automation*, pp. 9–15.
- [22] P. Corke, J. Lobo, and J. Dias, “An introduction to inertial and visual sensing,” *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 519–535, 2007.
- [23] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies, “Vision-aided inertial navigation for spacecraft entry, descent, and landing,” *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 264–280, 2009.
- [24] S. Weiss and R. Siegwart, “Real-time metric state estimation for modular vision-inertial systems,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4531–4537.
- [25] E. S. Jones and S. Soatto, “Visual-inertial navigation, mapping and localization: A scalable real-time causal approach,” *The International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011.
- [26] J. Kelly and G. S. Sukhatme, “Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration,” *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.
- [27] F. M. Mirzaei and S. I. Roumeliotis, “A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1143–1156, 2008.
- [28] H. H. B. V. Grabe and P. R. Giordano., “A comparison of scale estimation schemes for a quadrotor uav based on optical flow and imu measurements,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5193–5200.
- [29] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, “Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle,” *J. Field Robot.*, vol. 33, no. 4, pp. 431–450, 2016.
- [30] Google, “Google Self-Driving Car Project, Monthly Report. ” 2015.
- [31] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, “Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments,” in *IEEE International Conference on Robotics and Automation*, 2012, pp. 957–964.
- [32] I. E. Sutherland, “Three-dimensional data input by tablet,” *Proceedings of the IEEE*, vol. 62, no. 4, pp. 453–461, 1974.
- [33] Z. A. Hartley R, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

- [34] Bouguet, “Camera calibration toolbox for matlab, <http://www.vision.caltech.edu/bouguetj>,” 2010.
- [35] “Camera calibration with opencv, https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html.”
- [36] “Camera calibration with ros, http://wiki.ros.org/camera_calibration.”
- [37] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.
- [38] F. Devernay and O. Faugeras, “Straight lines have to be straight: Automatic calibration and removal of distortion from scenes of structured environments,” *Mach. Vision Appl.*, vol. 13, no. 1, pp. 14–24, 2001.
- [39] R. Tsai, “A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.
- [40] D. Scaramuzza, A. Martinelli, and R. Siegwart, “A flexible technique for accurate omnidirectional camera calibration and structure from motion,” in *Fourth IEEE International Conference on Computer Vision Systems (ICVS’06)*, 2006, pp. 45–45.
- [41] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.
- [42] H. Bay, T. Tuytelaars, and L. Van Gool, *SURF: Speeded Up Robust Features*, 2006.
- [43] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.
- [44] H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Real-time monocular slam: Why filter?” in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2657–2664.
- [45] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Bundle Adjustment — A Modern Synthesis*. Springer Berlin Heidelberg, 2000.
- [46] K. LEVENBERG, “A method for the solution of certain non-linear problems in least squares,” *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.
- [47] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.
- [48] K. Madsen, H. B. Nielsen, and O. Tingleff, *Methods for Non-Linear Least Squares Problems (2nd ed.)*. Informatics and Mathematical Modelling, Technical University of Denmark, 2004.
- [49] R. Szeliski, *Computer Vision: Algorithms and Applications*. London: Springer London, 2011.
- [50] M. I. A. Lourakis and A. A. Argyros, “Sba: A software package for generic sparse bundle adjustment,” *ACM Trans. Math. Softw.*, vol. 36, no. 1, pp. 2:1–2:30, 2009.
- [51] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking,” in *IEEE International Conference on Computer Vision*, vol. 2, 2005, p. 1508–1515.
- [52] J. E. Bresenham, “Algorithm for computer control of a digital plotter,” *IBM Systems journal*, vol. 4, no. 1, pp. 25–30, 1965.
- [53] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, *Adaptive and Generic Corner Detection Based on the Accelerated Segment Test*. Springer Berlin Heidelberg, 2010.

- [54] R. Mahony, V. Kumar, and P. Corke, “Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor,” *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.
- [55] S. Bouabdallah, *Design and control of quadrotors with application to autonomous flying*, PhD thesis. Ecole Polytechnique Federale de Lausanne (EPFL), 2007.
- [56] M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, “Inversion based direct position control and trajectory following for micro aerial vehicles,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2933–2939.
- [57] P. Martin and E. Salaün, “The true role of accelerometer feedback in quadrotor control.” in *IEEE International Conference on Robotics and Automation*. IEEE, 2010, p. 1623–1629.
- [58] M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, “Recent advances in visual-inertial mav navigation,” in *Workshop on Integration of perception with control and navigation for resource-limited, highly dynamic, autonomous systems, Robotics: Science and Systems*, vol. 4, 2013, p. 1623–1629.
- [59] L. Kneip, R. Siegwart, and M. Pollefeys, *Finding the Exact Rotation between Two Images Independently of the Translation*. Springer Berlin Heidelberg, 2012.
- [60] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, “A robust and modular multi-sensor fusion approach applied to mav navigation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3923–3929.
- [61] N. Trawny and S. I. Roumeliotis, “Indirect Kalman filter for 3D attitude estimation,” Tech. Rep. 2005-002, 2005.
- [62] J. B. Kuipers, *Quaternions and rotation sequences : a primer with applications to orbits, aerospace, and virtual reality*. Princeton, N.J. : Princeton University Press, 1999.
- [63] M. Raibert, *Quaternions proposed standard conventions*. Jet Propulsion Laboratory, 1999.
- [64] S. I. Roumeliotis and J. W. Burdick, “Stochastic cloning: a generalized framework for processing relative state measurements,” in *IEEE International Conference on Robotics and Automation*, vol. 2, 2002, p. 1788–1795.
- [65] P. S. Maybeck, *Stochastic models, estimation, and control*, ser. Mathematics in Science and Engineering, 1979.
- [66] R. Hermann and A. Krener, “Nonlinear controllability and observability,” *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, 1977.
- [67] S. Todorovic and M. C. Nechyba., “A vision system for intelligent mission profiles of micro air vehicles,” *Transactions on Vehicular Technology*, vol. 53, no. 22, pp. 1713–1725, 2004.
- [68] D. Dusha, W. W. Boles, and R. Walker, “Fixed-Wing Attitude Estimation Using Computer Vision Based Horizon Detection,” in *Australian International Aerospace Congress*, vol. 53, 2007, pp. 1–19.
- [69] R. J. D. Moore, S. Thurrowgood, D. Bland, D. Soccol, and M. V. Srinivasan, “A fast and adaptive method for estimating uav attitude from the visual horizon,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4935–4940.
- [70] W.-N. Lie, T. C. I. Lin, T.-C. Lin, and K.-S. Hung, “A Robust Dynamic Programming Algorithm to Extract Skyline in Images for Navigation,” *Pattern Recogn. Letter*, vol. 26, no. 5, pp. 221–230, 2005.
- [71] N. Ho and P. Chakravarty, “Localization on freeways using the horizon line signature,” in *International Conference on Robotics and Automation*, 2014.
- [72] J. A. Christian, “Optical Attitude Determination from Horizon Orientation Using Image Segmentation,” *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 1, pp. 113–123, 2012.

- [73] G. C. H. E. de Croon, C. D. Wagter, B. D. W. Remes, and R. Ruijsink, "Sky Segmentation Approach to obstacle avoidance," *Aerospace Conference*, pp. 1–16, 2011.
- [74] M. Zhai, S. Workman, and N. Jacobs, "Detecting vanishing points using global image context in a non-manhattan world," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [75] P. H. N. d. W. Bahman Zafarifar, Hans Weda, "Horizon detection based on sky-color and edge features," vol. 6822, 2008, pp. 6822 – 6822 – 9.
- [76] G. Evgeny, L. Tzvika, and S. Kosolapov, "Horizon Line Detection in Marine Images: Which Method to Choose?" *International Journal On Advances in Intelligent Systems*, 2013.
- [77] T. Ahmad, G. Bebis, M. Nicolescu, A. Nefian, and T. Fong, "Fusion of edge-less and edge-based approaches for horizon line detection," in *6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2015, pp. 1–6.
- [78] N. S. Boroujeni, S. A. Etemad, and A. Whitehead, "Robust horizon detection using segmentation for uav applications," in *Ninth Conference on Computer and Robot Vision*, 2012, pp. 346–352.
- [79] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, IEEE International Conference on*, vol. 4, 1995, pp. 1942–1948.
- [80] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.
- [81] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [82] Hough and V. C. Paul, "METHOD AND MEANS FOR RECOGNIZING COMPLEX PATTERNS," 1962.
- [83] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *British Machine Vision Conference (BMVC)*, vol. 1, no. 2, 2011, pp. 1–11.
- [84] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.
- [85] Y. Ke and R. Sukthankar, "Pca-sift: a more distinctive representation for local image descriptors," in *IEEE Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, 2004, pp. II–506–II–513 Vol.2.
- [86] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.
- [87] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *Int. J. Comput. Vision*, vol. 43, no. 1, pp. 29–44, 2001.
- [88] A. Klaser, M. Marszalek, and C. Schmid, "A Spatio-Temporal Descriptor Based on 3D-Gradients," in *BMVC 19th British Machine Vision Conference*, 2008, pp. 275:1–10.
- [89] J. M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast anisotropic gauss filtering," *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 938–943, 2003.
- [90] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, no. Supplement C, pp. 53 – 65, 1987.
- [91] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

- [92] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *In VISAPP International Conference on Computer Vision Theory and Applications*, 2009, pp. 331–340.
- [93] A. Miller, M. Shah, and D. Harper, “Landing a uav on a runway using image registration,” in *IEEE International Conference on Robotics and Automation*, 2008, pp. 182–187.
- [94] T. D. Cornall, G. K. Egan, and A. Price, “Aircraft attitude estimation from horizon video,” *Electronics Letters*, vol. 42, no. 13, pp. 744–745, 2006.
- [95] S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak, “Vision-guided flight stability and control for micro air vehicles,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, pp. 2134–2140 vol.3.
- [96] B. D. W. Remes, P. Esden-Tempski, F. van Tienen, E. Smeur, C. D. Wagter, and G. C. H. E. de Croon, *Lisa" S 2.8g Autopilot for GPS-based Flight of MAVs*. Delft University of Technology and Thales, 2014.
- [97] S. L. W. Haomiao Huang, Gabriel M. Hoffmann and C. J. Tom-lin, “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering,” in *IEEE International Conference on Intelligent Robots and Systems*, 2009.
- [98] S. H. Kim, C. W. Roh, S. C. Kang, and M. Y. Park, “Outdoor navigation of a mobile robot using differential gps and curb detection,” in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3414–3419.
- [99] J. Biswas and M. Veloso, “Wifi localization and navigation for autonomous indoor mobile robots,” in *IEEE International Conference on Robotics and Automation*, 2010, pp. 4379–4384.
- [100] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal, “Efficient, generalized indoor wifi graphslam,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1038–1043.
- [101] S. Timotheatos, G. Tsagakatakis, P. Tsakalides, and P. Trahanias, “Feature extraction and learning for rssi based indoor device localization,” in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2017.
- [102] R. Williams, B. Konev, and F. Coenen, “Scalable distributed collaborative tracking and mapping with micro aerial vehicles,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 3092–3097.
- [103] E. Olson, “Apriltag: A robust and flexible visual fiducial system,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407.
- [104] “Vicon, <https://www.vicon.com/>.”
- [105] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “Monoslam: Real-time single camera slam,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.
- [106] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “Robust visual inertial odometry using a direct ekf-based approach,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 298–304.
- [107] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, “An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications,” in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1736–1741.
- [108] H. C. Longuet-Higgins, “A computer algorithm for reconstructing a scene from two projections,” *Nature*, no. 5828, pp. 133–135, 1981.
- [109] F. Fraundorfer, P. Tanskanen, and M. Pollefeys, *A Minimal Case Solution to the Calibrated Relative Pose Problem for the Case of Two Known Orientation Angles*. Springer Berlin Heidelberg, 2010, pp. 269–282.

- [110] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, “Orb-slam: A versatile and accurate monocular slam system,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [111] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 834–849.
- [112] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “Dtam: Dense tracking and mapping in real-time,” in *2011 International Conference on Computer Vision*, 2011, pp. 2320–2327.
- [113] C. Forster, M. Pizzoli, and D. Scaramuzza, “Svo: Fast semi-direct monocular visual odometry,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22.
- [114] T. Pire, T. Fischer, G. Castro, P. D. Cristóforis, J. Civera, and J. J. Berlles, “S-ptam: Stereo parallel tracking and mapping,” *Robotics and Autonomous Systems*, no. Supplement C, pp. 27 – 42, 2017.
- [115] G. Sibley, C. Mei, I. Reid, and P. Newman, “Vast-scale Outdoor Navigation Using Adaptive Relative Bundle Adjustment,” vol. 29, no. 8, 2009, pp. 958–980.
- [116] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, “The euroc micro aerial vehicle datasets,” *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.
- [117] “ASL dataset, http://projects.asl.ethz.ch/datasets/doku.php?id=bibtex:euroc_datasets.”
- [118] S. Workman, M. Zhai, and N. Jacobs, “Horizon Lines in the Wild,” in *British Machine Vision Conference*, 2016.
- [119] T. Praczyk, “A quick algorithm for horizon line detection in marine images,” *Journal of Marine Science and Technology*, 2017.
- [120] M. H. Tehrani, M. A. Garratt, and S. G. Anavatti, “Low-altitude horizon-based aircraft attitude estimation using uv-filtered panoramic images and optic flow,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2362–2375, 2016.
- [121] M. Schwendeman and J. Thomson, “A horizon-tracking method for shipboard video stabilization and rectification,” *Journal of Atmospheric and Oceanic Technology*, vol. 32, no. 1, pp. 164–176, 2015.
- [122] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.
- [123] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “Svo: Semidirect visual odometry for monocular and multicamera systems,” *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.
- [124] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [125] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, “The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.
- [126] S. Piperakis and P. Trahanias, “Non-linear zmp based state estimation for humanoid robot locomotion,” in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 202–209.
- [127] G. Best, J. Faigl, and R. Fitch, “Multi-robot path planning for budgeted active perception with self-organising maps,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3164–3171.
- [128] J. Yu, M. Schwager, and D. Rus, “Correlated orienteering problem and its application to persistent monitoring tasks,” *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

- [129] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss, in *IEEE International Conference on Robotics and Automation (ICRA)*.
- [130] P. M. Dames, M. Schwager, D. Rus, and V. Kumar, “Active magnetic anomaly detection using multiple micro aerial vehicles,” *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 153–160, 2016.
- [131] S. Nagavalli, M. Chandarana, K. Sycara, and M. Lewis, “Multi-Operator Gesture Control of Robotic Swarms Using Wearable Devices,” in *International Conference on Advances in Computer-Human Interactions*, 2016.
- [132] F. Sadeghi and S. Levine, “(cad)\$^2\$rl: Real single-image flight without a single real image,” *CoRR*, vol. abs/1611.04201, 2016.