AUTONOMOUS NAVIGATION IN COMPLEX INDOOR AND OUTDOOR ENVIRONMENTS WITH MICRO AERIAL VEHICLES

Shaojie Shen

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

2014

_____
Vijay Kumar, Supervisor of Dissertation
UPS Foundation Professor of Mechanical Engineering and Applied Mechanics

_____
Nathan Michael, Co-Supervisor of Dissertation
Assistant Research Professor of Robotics

_____
Saswati Sarkar, Graduate Group Chairperson
Professor of Electrical and Systems Engineering

Dissertation Committee:

Daniel D. Lee, Professor of Electrical and Systems Engineering
Vijay Kumar, Professor of Mechanical Engineering and Applied Mechanics
Nathan Michael, Assistant Research Professor of Robotics
Davide Scaramuzza, Assistant Professor of Robotics

Acknowledgments

This thesis will not be possible without the support of many people. I would first like to thank my advisors Vijay Kumar and Nathan Michael, for all their support and guidance throughout my graduate studies at Penn. I would also like to thank my thesis committee members Daniel Lee and Davide Scaramuzza, for taking the time to serve on my committee. Additionally, I would like to thank fellow and past members of the Multi-Robot Systems Lab (MRSL), as well as others in the GRASP lab, for many great technical discussions and fun time. Specially, I would like to thank Kartik Mohta and Yash Mulgaonkar for their help in making and debugging many robotics systems.

I am indebted to my parents for their love and support over years across continents and oceans. Last but not least, I thank my wife, Shuangyu, who has been with me every moment through all ups and downs.

ABSTRACT

AUTONOMOUS NAVIGATION IN COMPLEX INDOOR AND OUTDOOR ENVIRONMENTS WITH MICRO AERIAL VEHICLES

Shaojie Shen

Vijay Kumar

Nathan Michael

Micro aerial vehicles (MAVs) are ideal platforms for surveillance and search and rescue in confined indoor and outdoor environments due to their small size, superior mobility, and hover capability. In such missions, it is essential that the MAV is capable of autonomous flight to minimize operator workload. Despite recent successes in commercialization of GPS-based autonomous MAVs, autonomous navigation in complex and possibly GPS-denied environments gives rise to challenging engineering problems that require an integrated approach to perception, estimation, planning, control, and high level situational awareness. Among these, state estimation is the first and most critical component for autonomous flight, especially because of the inherently fast dynamics of MAVs and the possibly unknown environmental conditions. In this thesis, we present methodologies and system designs, with a focus on state estimation, that enable a light-weight off-the-shelf quadrotor MAV to autonomously navigate complex unknown indoor and outdoor environments using only onboard sensing and computation. We start by developing laser and vision-based state estimation methodologies for indoor autonomous flight. We then investigate fusion from heterogeneous sensors to improve robustness and enable operations in complex indoor and outdoor environments. We further propose estimation algorithms for on-the-fly initialization and online failure recovery. Finally, we present planning, control, and environment coverage strategies for integrated high-level autonomy behaviors. Extensive online experimental results are presented throughout the thesis. We conclude by proposing future research opportunities.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Micro aerial vehicles (MAVs) are ideal platforms for a wide range of applications in indoor and outdoor environments due to their small size, superior mobility, and hover capability. In such missions, it is essential that the MAV is capable of autonomous flight to minimize operator workload. Thanks to the Global Positioning System (GPS) technology, we have seen successful commercialization of autonomous MAVs in outdoor applications such as aerial photography [2], transportation [1], and intelligent farming [3]. However, we see great potential of using MAVs in surveillance and search and rescue missions in confined and complex indoor and outdoor environments that are inaccessible or dangerous for human and ground vehicles. Key challenges while operating such environments are the lack of GPS signal, limited or no external infrastructure, existence of unknown obstacles in close proximity, as well as the necessity of full autonomy with minimum human interaction. It is also desirable to have small and fast moving MAVs in such scenarios due to the time critical nature of search and rescue missions. Up to date, building a MAV that satisfies all the size, weight, and power (SWaP) constraints and is capable of fast autonomous navigation in such environments still gives rise to challenging engineering problems that requires knowledge and integration of perception, estimation, planning, control, and high level situational awareness. Among these, state estimation is the foremost critical component for autonomous flight especially because of the inher-

ently fast dynamics of MAVs and the possibly unknown environmental conditions.

In this thesis, we present methodologies and system designs, with a focus on state estimation, that enable a light-weight off-the-shelf quadrotor MAV to autonomously navigate complex unknown indoor and outdoor environments using only onboard sensing and computation. We look into different sensing options and propose algorithms to achieve the desired estimation capabilities. We propose laser- and vision-based state estimation methodologies for indoor autonomous flight. We then investigate into fusing information from heterogeneous sensors to increase capable operational environments. We further propose algorithms to enable on-the-fly initialization and online failure recovery for systems with minimum sensing capability. Finally, we present planning, trajectory generation, and environment coverage strategies for integrated high-level autonomy behaviors. Extensive online experimental results are presented throughout the thesis.

1.1 Research Problems

We begin by reviewing a number of research problems that arise during the development of autonomous MAVs. We will review literatures on each problem, as well as its relevant fields, in the next chapter (Ch. 2). The topic of autonomous aerial navigation has been studied extensively over the past few years. Due to cost and payload constraints, most MAVs are equipped with low cost proprioceptive sensors such as micro-electro-mechanical (MEMS) inertial measurement units (IMUs) that are incapable for long term state estimation. As such, exteroceptive sensors, such as GPS, laser scanners, and cameras are usually fused with proprioceptive sensors to improve estimation accuracy. While GPS-based navigation is well-developed by the aerospace community [38, 70], autonomous navigation in GPS-denied environments has only gained popularity recently.

1.1.1 Autonomous Flight in GPS-denied Environments

In general, challenges in autonomous flight in GPS-denied environments using onboard sensors include state estimation with limited sensing and computation capabilities, as well as handling the inherently fast dynamics of MAVs. Two most popular types of onboard exteroceptive sensors are laser scanners and cameras. Laser-based autonomous flight approaches are popular in earlier stages of research due to its relatively low processing demand [5, 20, 30]. Vision-based approaches enable simultaneous six degree-of-freedom (6-DOF) state estimation and 3D reconstruction. Popular approaches involve the use of monocular camera [114, 115], stereo cameras [27, 89], or RGB-D sensors [32]. Cameras offer much better information to weight ratio comparing to laser scanners, but at the expense of high computation demands for processing image data. However, recent advancements in embedded computing, especially in multi-core and GPU-equipped mobile processors, already push the computation power to a level that real-time onbaord vision-based state estimation is feasible.

1.1.2 Multi-Sensor Fusion for Autonomous Flight

However, all approaches that rely on a single exteroceptive sensing modality are only functional in certain environments. For example, laser-based approaches require structured environments, vision-based approaches demand sufficient lighting and features, and GPS only works outdoors. This makes them prone to failure in large-scale missions that involve indoor-outdoor transitions, during which the environment may change significantly. In some environments, information from multiple heterogeneous sensors may be available, and fusing all available measurements may yield improved estimator accuracy and robustness. However, this extra information is often either ignored or used as switching between sensor suites [110].

1.1.3 Estimator Initialization and Failure Recovery

In practice, even the most reliable estimator is prone to fail due to the lack of information in the environment. It may also be desirable to launch the MAV from another moving object with unknown velocity and attitude (handheld, truck, ship, or another aircraft). This motivates us to develop initialization and failure recovery mechanisms that (re)initialize the estimator from an unknown state with significant velocity and acceleration. This (re)initialization should be done as soon as enough information has been gathered by onboard sensors. This is a challenging problem since quadrotor MAVs are highly nonlinear systems that operate in full three dimensional environments, and almost all state-of-the-art nonlinear state estimation algorithms are inoperable without a good initialization point. This problem is largely overlooked by the community, however, it is a crucial step towards fail-safe navigation systems. The problem is even more bothersome for the extreme case of monocular visual-inertial systems (VINS), which consists of only a camera and an IMU, due to the lack of direct scale measurements.

1.1.4 Planning and Control

Given onboard state estimates, we are able to use a variety of existing control strategies to stabilize the MAV. For MAVs operating at low speeds with small changes in acceleration, a linear controller that assumes near hover state can be used [71]. For more aggressive motions that involve large attitude changes, a controller with nonlinear error metric is a more appropriate choice [51]. Autonomous navigation requires collision-free paths through the incrementally built environment model, this can be done by employing state-of-the-art motion planning algorithms [40]. Bridging the gap between position tracking control and high level planning is trajectory generation, which aims to provide feasible time-parameterized trajectories for the MAV [69]. All these are necessary components to enable autonomous navigation and each contains a broad collection of literatures. However, in this thesis, we focus on state estimation and only adapt relevant approaches in

planning and control for system integration.

1.1.5 Autonomous Environment Coverage

Surveillance and search and rescue missions are important applications of MAVs in confined spaces. This kind of missions demand the MAV to return a complete map of the environment of interest with minimal or hopefully no human interaction. This problem is call exploration and it is a well-known problem in the field of mobile robotics. In this thesis, we review challenges when applying existing exploration approaches on MAV platforms and propose solutions to enable efficient exploration in unknown and unstructured 3D environments.

1.2 Thesis Overview

Based on previous discussions, we require a state estimator that is suitable for MAV applications to exhibit following attributes:

- *Power-on-and-go*: Initialize from an arbitrary unknown state (Ch. 6);
- *Autonomy*: Estimate its own state in a wide range of environments (Ch. 3, 4, 5);
- *Fault-tolerant*: Deal with failure of one or more of its onboard sensors (Ch. 5);
- *Fail-safe*: Recover from total failure of all sensors (Ch. 6);

We focus on developing an estimator that meets all these requirements, and thus providing reliable state feedback for autonomous flight. Each of these requirements related to different chapters in this thesis.

Building on top this estimator, we further consider autonomous navigation and environment coverage as the the integrated functionality of a MAV to perform the following iterative process:

- Estimate its state and map the environment (Ch. 3, 4, 5, 6);

- Generate waypoints that improves environment coverage (Ch. 8);
- Generate collision-free paths to these waypoints (Ch. 7);
- Generate time parameterized trajectories from this path (Ch. 7);
- Autonomously fly through this trajectory to desired waypoints (Ch. 7);

There are a large number of building blocks in a fully functional autonomous navigation system. Although we have developed the majority of these building blocks over the past few years, not all of them involves novel scientific contributions. However, the integration of these building blocks requires careful selection and evaluation of algorithms, as well as rigorous testing. This does represent significant technical contributions.

In particular, based on state estimation approach that we developed, we adapt approaches in planning, trajectory generation, and control for our applications. This forms a system that is capable of guiding the MAV to fly to user-specified waypoints in complex indoor and outdoor environments. Built on top of such system, we propose an algorithm that eliminates the need of human operator and enables fully autonomous environment coverage. The flow of topics of this thesis, which is also the typical flow of a navigation system, is shown in Fig. 1.1.

Chapter 3 [72, 73, 93, 96] considers the problem of autonomous navigation with a MAV in confined indoor environments with multiple floors using a laser scanner as the main sensor. To ensure that the MAV is fully autonomous, we require all computation to occur onboard the MAV without need for external infrastructure, communication, or human interaction beyond high-level commands. We also highlight field experiments in multiple environments.

Chapter 4 [97, 98, 99] addresses the development of a light-weight autonomous quadrotor that uses two cameras and an inexpensive IMU as its only sensors and onboard pro-

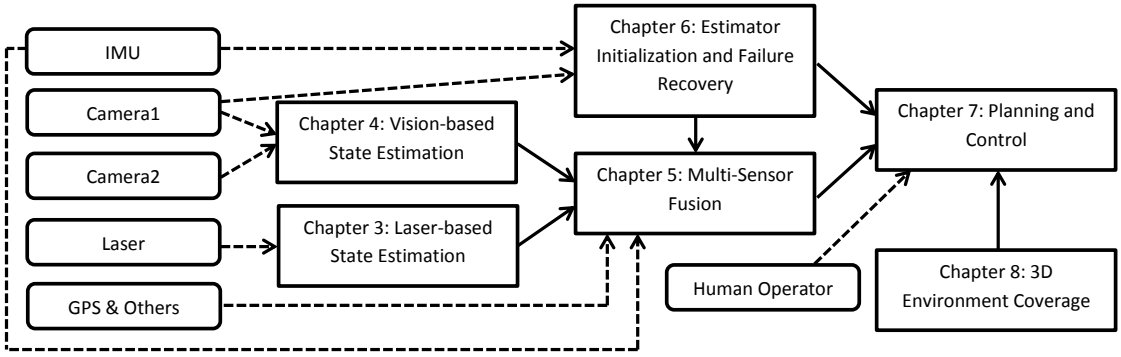


Figure 1.1: A graphical illustration of the flow of topics and connections between different chapters. It also represents the information flow of our autonomous navigation system. External information (sensor or human) is shown as dashed lines. Internal information flow is shown in solid lines. Boxes with curved edges represent sensors. Rectangular boxes represent technical chapters.

cessors for estimation and control. We describe a fully-functional, integrated system with a focus on fast visual-inertial state estimation, and demonstrate the quadrotor's ability to autonomously travel in indoor environments at high speed.

Chapter 5 [92, 100] merges results from the previous two chapters by presenting a modular and extensible approach to integrate noisy measurements from multiple heterogeneous sensors that yield either absolute or relative observations at different and varying time intervals, and to provide smooth and globally consistent pose estimates in real time for autonomous flight. We illustrate the robustness of our framework in large-scale, indoor-outdoor autonomous aerial navigation experiments.

Chapter 6 [101] studies on-the-fly initialization and failure recovery for MAV onboard estimators, with focus on a monocular VINS setting. We present an a linear formulation that is capable of estimating all critical states without any initial knowledge of the system. Based on this linear initialization, we further propose a nonlinear optimization approach that leads to highly accurate state estimates. We also present a special marginalization scheme that preserve scale observability even during degenerate motions. We demon-

strate that our approach forms a reliable VINS system that is capable of non-stationary launching and is suitable for high speed autonomous flight.

Chapter 7 presents common building blocks related to planning and control that relies on our estimation methodologies and are used in almost all online experiments. These building blocks are adaptations from existing results to match the unique need of autonomous aerial navigation.

Chapter 8 [94, 95] considers the problem of automatically generating waypoints in the incrementally built 3D environment, such that, if these waypoints are visited by the MAV, full coverage of the environment can be achieved. This is built on top of all previous chapters to enable operations without any human interaction. This problem is called exploration in robotics. We study the failure modes of directly applying existing 2D exploration strategies in a 3D setting. We then propose a stochastic algorithm that utilize sparse free space representation of the 3D environment to efficiently generate desirable waypoints.

Note that except Ch. 7 which rely on results from previous chapters, all other chapters are self-contained and may be read independently. Each chapter presents both explanations of advancements in a specific area related to autonomous flight and corresponding experimental results.

1.3 Overview of Experimental Platforms

A crucial part of our research is experimental validation of of all proposed algorithms and systems. To this end, we develop different generations of MAV platforms that are suitable for experimental purposes. All our MAV platforms (Fig. 1.2) are built using off-the-shelf components. The base platforms are either the Hummingbird (Fig. 1.2(b)) or the

Pelican (Figs. 1.2(a), 1.2(c), 1.2(d)) quadrotors purchased from Ascending Technologies, GmbH[1]. All base platforms are natively equipped with an AutoPilot board consisting of an IMU, a magnetometer, a barometer, and a user-programmable ARM7 microcontroller. We use this microcontroller for low level attitude stabilization of the quadrotor. As shown in Table. 1.1, we outfit base platforms with a variety of sensing and computation units, resulting in fully equipped MAVs that weights from 0.74 kg to 1.9 kg.

Specifically, the sensors we used (Table. 1.1) include u-blox LEA-6T GPS modules[2], Hokuyo UTM-30LX laser range finders[3] that run at 30 Hz, and mvBlueFOX-MLC200w grayscale HDR cameras[4] that capture VGA resolution images at 25 Hz. We fine tune the camera auto exposure controller to enable fast adaption during rapid light condition changes. Hardware triggering is used for synchronization between sensors.

Our earlier platforms (Figs. 1.2(a), 1.2(b)) are equipped with an embedded computer with an Intel Atom 1.6 GHz processor and with 1 GB RAM and 8 GB SD card. Although it is fastest available processor at that time, the Atom have very limited computation power and thus motivates and forces us to develop efficient estimation methodologies for autonomous navigation. In later platforms (Figs. 1.2(c), 1.2(d)), a powerful Intel NUC computer with Intel Core i3 1.8 GHz processors, up to 16 GB RAM and 120 GB SSD enables the development of more sophisticated algorithms. All methodologies discussed in this thesis are implemented in C++ using the Robot Operating System (ROS)[5] as the interfacing robotics middleware.

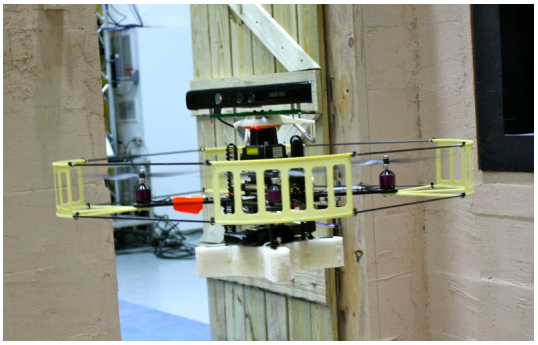
[1] http://www.asctec.de

[2] http://www.u-blox.com

[3] http://www.hokuyo-aut.jp

[4] http://www.matrix-vision.com

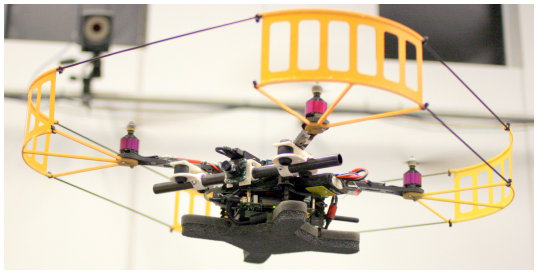
[5] http://www.ros.org



(a) Chapter 3

(b) Chapter 4

(c) Chapter 5



(d) Chapter 6

Figure 1.2: List of platforms with different size, weight, and different sensing and computation capabilities. Each platform is developed specifically for the desired estimator performance described in its corresponding chapter.

Robot	**Chapter**	**Sensing**	**Computation**	**Mass**
Fig. 1.2(a)	2	IMU, Laser, RGB-D	Intel Atom 1.6 GHz	1.9 kg
Fig. 1.2(b)	3	IMU, Cameras	Intel Atom 1.6 GHz	0.74 kg
Fig. 1.2(c)	4	IMU, Cameras, GPS Magnetometer, etc	Intel Core i3 1.8 GHz	1.9 kg
Fig. 1.2(d)	5	IMU, Camera	Intel Core i3 1.8 GHz	1.3 kg

Table 1.1: Comparison of different experimental platforms.

1.4 Thesis Contributions

While there are continuous advancements in all areas related to MAVs, we put our focus on perception and state estimation for autonomous flight, which is the foremost crucial component for any autonomous MAV. Building on top of this, we adapt existing planning and control approaches and propose environment coverage strategies to form fully autonomous systems.

Ch. 3 and 4 focus on the development of real-time navigation systems using onboard sensing and very limited onboard processing. While there are much less limitations regarding onboard computation power nowadays, methodologies and results presented in these chapters represent significant advancements of system capabilities given the available onboard computation at the time of development. They were commonly considered a baseline for MAV navigation research.

Ch. 5 considers fusing heterogeneous sensors in a globally consistent manner. A core feature of our approach is a modular derivative-free design that allows easy addition/removal of sensors with minimum calculation and coding. We give special focus on the fusion of global pose measurements to ensure smoothness in the state estimates. This is novel to existing approaches [62] in terms of both system capability and readiness for practical applications.

Ch. 6 proposes a novel linear formulation that properly take the noise characteristic of onboard sensors to enable estimation of significant but unknown initial values using a sequence of sensor readings. This reduces the sensitivity to sensor noise as in previous approaches [17, 45, 56, 66, 67] and enable real-world deployment with very noisy sensors. Based on the linear initialization, we further propose a nonlinear optimization to improve estimation accuracy. The two subsystems works interactively to enable fast maneuvers and failure recovery. We present high speed autonomous flight experiments to demonstrate system capability.

Ch. 7 adapts existing planning and control approaches to meet the need of autonomous navigation with onboard sensing. Specifically, we propose control in local frame to en-

sure smooth flight and minimum jerk trajectory generation to minimize motion blur.

Ch 8 presents a novel algorithm for generating 3D waypoints for achieving full environment coverage. This work aims to resolve difficulties when applying classic strategies originally designed for 2D environments. It outperforms other 3D exploration algorithms [18, 28, 90, 91, 111] in terms of speed. It is the first algorithm that achieves real-time operation in 3D environments using only embedded processors.

Chapter 2

Scientific Background and Literature Review

2.1 Autonomous Flight in GPS-denied Environments

The literature on autonomous flight in GPS-denied environments is recent but extensive. A survey in [42] provides a thorough review of ongoing research in the domain of autonomous navigation approaches for rotorcraft platforms.

Earlier work on laser-based autonomous flight MAVs frequently require a partially-structured environment to enable incremental motion calculations. Particularly relevant to our approach is the work in [5], and [30], where laser scanners serve as the primary source of sensory information. Similarly [20] presents an open-source implementation of a laser-based localization system for a MAV. Mechanized panning laser-scanners that add considerable payload mass [49] may also be used to improve the 3D sensing capability. With a known map, it is also possible to fly in general 3D environments by combining measurements from laser and IMU [8]. However, the main issue of laser-based approaches is the weight of the sensor, which significantly constrains the agility of the platform in confined environments.

Cameras are able to generate a large amount of data with very small footprint, and

therefore are very attractive for MAVs with limited payloads. Approsches are proposed to utilize a optical flow-based velocity estimator [114], or a monocular SLAM framework [87, 113, 115], as the main vision processing pipeline. In conjunction with a loosely coupled filtering framework, these approaches successfully enable autonomous quadrotor flight via a downward-facing camera. However, due to the unavaibility of direct distance measurement of monocular vision system, these approaches relies on the assumption of slowly-varying or good initialization of the visual scale. This can be difficult to enforce during fast motions at low-altitudes in unknown environments with potentially rapid changes in the observed features. On the other hand, stereo vision-based state estimation approaches for autonomous MAVs such as those proposed in [27, 89] have the advantage of direct scale observation. RGB-D sensors [32] have even higher depth measurement accuracy than stereo-based systems, but RGB-D sensors are inoperable in outdoor environment with strong sunlight. On the other end of the spectrum, Bills et al. [7] propose a map-less navigation algorithm for use with a MAV to circumvent this challenge, but at the cost of not building a global representation of the environment.

In the next few sections, we expand our discussion to technical fields that are most relevant to state estimation and autonomous navigation of MAVs.

2.2 Incremental Motion Estimation

Incremental motion estimation with laser scanners or cameras, or commonly known as laser/visual odometry, is a key component almost all autonomous MAV approaches. Pairwise sensor measurements, such as consecutive laser scans or images, are usually used to estimate the differential motion of the platform.

For the laser-based setting, the iterative closest point (ICP) algorithm and its variations [85] are popular methods for incremental motion estimation. However, ICP is an iterative optimization approach without explicit data association and it has poor convergence property when there exist large differences in sensor data. Multi-resolution

correlative scan matching [80, 81] reports surprising robust results even with large scan displacement. Corner or line features may also be extracted for data association between scans [5, 109]. With such data association, scans may be directly matched to obtain the transformation between them.

Vision-based approaches usually involve detection and matching/tracking of salient features in the image, such as corners [102], or blobs [6], and estimate the motion given 2D-2D [78] or 2D-3D [23] feature correspondences. Feature-based visual odometry approaches require that the environment is feature-rich. Recently, approaches that are based on dense image alignment using intensity values [43, 77] demonstrate promising results in terms of robustness in featureless environments and against motion blur. However, such approaches requires tremendous amount of computation that is currently not available onboard MAV platforms. A semi-dense approach [24] that combine the advantages of direct image alignment and feature-based approaches demonstrates promising results in terms of both speed and robustness.

Incremental motion estimates are usually fast to compute, which is beneficial for MAVs equipped with limited onboard computation, but it has the disadvantage of drifting over time. Usually, additional mapping and correction steps are required to make the system globally or at least locally drift-free.

2.3 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) is a problem formulation that requires the robot to construct a map of the observed environment, while at the same time localize within this map. In SLAM, the environment is often assumed to be previously unknown but contain sufficient salient features. The central idea of SLAM is to obtain a maximum likelihood estimate of both robot states and environment features given observations from a variety of sensors such as laser scanner [29], monocular camera [44], or stereo cameras [54].

Pioneering work in SLAM adapts the extended Kalman filter (EKF) framework by including the current robot pose and the locations of all features in a large state vector [53]. EKF SLAM needs to maintain and inverse a dense covariance matrix that has quadratic size with respect to the size of the state vector, thus EKF-based approaches become very slow as the observed environment grows.

Particle filter-based SLAM approaches are also popular due to their linear complexity with respect to the number of particles [29]. Methods have been proposed to reduce the required number of particles to accurately represent the joint probabilistic distribution of poses and features [29]. However, even for the simplest 2D problem and with very accurate sensor measurements, particle filter-based approaches cannot guarantee completeness with a limited number of particles.

Recently, graph-based optimization techniques, which also called bundle adjustment in the computer vision community, have gained popularity [16, 44, 48]. In such setting, both robot states and environment features are considered as vertices in the graph, where observations between vertices are considered as factor node in a factor graph formulation [16], or as edges in a Markov random field formulation [48]. These are generic formulations that allow modeling of observations from multiple sources, such as wheel/laser/visual odometry, feature observations, loop closures, and GPS. Despite their large size, the graphs resulting from all observations are typically sparse, allowing fast solution with sparse matrix solvers.

A simplify version of the full SLAM problem is pose-only SLAM, This formulation considers only robot states in the graph. Global consistency is achieved by detecting loop closures between current and past poses [21]. Pose-only SLAM is considerably faster in terms of processing speed, but at the expense of suboptimal results due to the ignorance of feature observation constraints across multiple poses.

However, batch SLAM can still become unacceptably slow as the size of the environment grows. Incremental update methods [39] have been proposed to speed up the computation. The sliding window formulation that only keep a limited amount of robot

states and features is also a popular way to bound the computation complexity. Conditioning [44] and marginalization [103] are two methods to propagate information from removed states into the current estimate. A comparison of incremental motion estimation, SLAM, and sliding window approaches are shown in Fig. 2.1.

As MAVs typically come with tight constraints on onboard computation, it is therefore critical to choose appropriate SLAM formulations and carefully implement the SLAM system in a way that possible latency in SLAM does not breakdown the whole system.

2.4 Multi-Sensor Fusion

When information from multiple sensors is available, it is often desirable to fuse all of is to improve the system robustness, since all sensors are subject to fail in certain environments (Sect. 1.1.2). It is straightforward perform multi-sensor fusion in a graph-based SLAM framework as each measurement simply adds another edge or factor node to the graph [11, 34, 88]. Graph-based approaches have the advantage of being able to handle delayed and out of sequence measurements as past states are stored. However, in autonomous MAVs, state estimates from sensor fusion are often directly used for stabilizing the vehicle, thus requiring high rate and low latency state estimates. In such case, computationally expensive graph-based approaches may not be appropriate.

On the other hand, Kalman filtering-based approaches are able to fuse multiple sensors simply by using multiple measurement models, resulting in significant speed up in computation. However, naive Kalman filters implementations only keep the most recent robot state. As such, while it is straightforward to fuse multiple absolute measurements such as GPS, pressure/laser altimeter in a recursive filtering formulation, the fusion of multiple relative measurements obtained from laser or visual odometry is more involved. It is common to accumulate the relative measurements with the previous state estimates fuse them as pseudo-absolute measurements [114]. However, such fusion is sub-optimal since the resulting global position and yaw covariance is inconsistently small compared to

the actual estimation error. This violates the observability properties [46], which suggests that such global quantities should be unobservable. To resolve this issue, state augmentation techniques [84], which include reference states of all relative measurements in the state vector, properly account for the state uncertainty when applying multiple relative measurements from multiple sensors.

Considering MAV applications, [62] recently proposed a open source EKF-based multi-sensor fusion framework. However, this framework does not support fusion of multiple heterogeneous relative measurements in a modular and extensible fashion.

2.5 Monocular Visual-Inertial State Estimation

Despite the fact that fusing multiple sensors lead to improved robustness, we also have cases that a platform with the smallest size is desirable. In such case, monocular visual-inertial systems (VINS) that consists of a camera and a low cost IMU are very attractive to MAVs with limited payload budget due to their small footprint, low cost, and low maintenance.

It is obvious that monocular VINS is unsolvable using only pairwise images due to the lack of direct scale measurements. Using measurements from multiple time instants, solutions to monocular VINS has been proposed in a filtering setting [37, 41, 46, 47, 55, 76, 114, 115] and in a graph-based optimization/bundle adjustment setting [35, 54, 103]. Filtering-based approaches have the advantage of relatively fast processing due to its continuous marginalization of past states, but their performance can be sub-optimal due to early fix of linearization points. Graph-based approaches benefit from iterative re-linearization of states but it requires more computation power. With proper marginalization, a constant complexity sliding window graph-based framework can be obtained [103]. Conditioning is also a popular method among the computation vision community to achieve constant computation complexity [44]. A comparison between filtering and graph-based approaches is presented in [50]. The authors reported nearly

identical results of two types of approaches. However, the platform for verification is only equipped with an optical flow sensor that is unable to perform long term feature tracking. This limits the power of graph-based approach, as a sufficiently connected graph is never constructed.

We can also categorize VINS solutions as loosely coupled [114] or tightly coupled [35, 37, 41, 46, 47, 54, 55, 76]. Loosely coupled approaches usually utilize an independent vision processing module such as PTAM [44] for up-to-scale pose estimation, and integrating the vision pose with IMU using a filter for scale estimation. Tightly coupled approaches usually lead to better estimation results (up to linearization error) because they integrate camera measurement and noise models in a systematic manner.

Another challenge with monocular VINS is the scale ambiguity due to degenerate motion. It is well known that in order to render the scale observable, accelerations in at least two axes are required [37, 41, 66]. However, for a MAV, degenerate motions such as hovering and constant velocity motions are unavoidable. The hover case can be addressed via conditioning in a keyframe-based loosely coupled approach [44], or with a last-in-first-out (LIFO) scheme in a tightly-coupled sliding window approach [47].

In fact, VINS is very closely related to SLAM in the sense that most VINS approaches operates in unknown environments which requires simultaneous estimation of both the trajectory of the sensor suite as well as environments. However, VINS estimators estimators usually use a sliding window and consider a limited number of vehicle states and environment features in order to bound the computation cost. They often do not consider loop closures or global mapping.

2.6 Estimator Initialization and Failure Recovery

All state-of-the-art nonlinear solvers requires good initial values that are close to the global optimal in order to converge properly. In practice, initialization is usually handled in the following ways. First, there are cases that full state observation is available. The

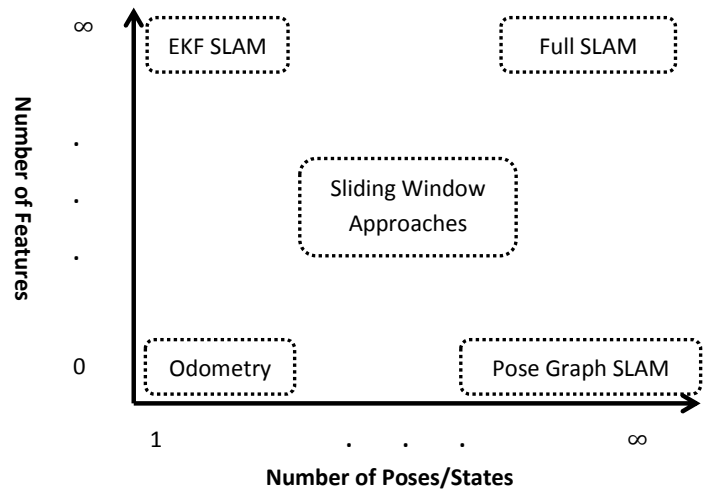


Figure 2.1: Comparision between different state estimation approaches, characterized by the number of robot poses/states and/or environment features kept in the system.

most typical example in this case is GPS-based navigation, where the first GPS position and velocity reading can be used to initialize the estimator [38, 70]. Second, if all unobservable states are known to be stationary at the time that the estimator starts, they can be initialized as zero. In SLAM with ground robots, the initial pose of the robot can be set in this way. Finally, other sources may be helpful to obtain a good approximation of the initialization value. This approach is commonly used in MAV applications, especially for the monocular VINS case. For example, [114] uses the average scene depth to initialize the scale filter, [115] uses readings from the pressure altimeter for scale initialization. If direct scale observation is available as in the stereo setup, initial velocity of the platform can be approximated by numerically differentiate first two poses. The initial attitude of the platform can be obtained from the readings from the accelerometer by assuming the linear acceleration of the platform is small.

However, none of these approaches have in principle solved the initialization problem. Consider the case that a MAV equipped with a monocular VINS sensor suite be launch from a moving object (human or another moving ground or aerial vehicles) with unknown velocity and significant acceleration, since the visual scale is not directly ob-

servable, and the platform is not launched from a stationary condition, all previously introduced initialization assumptions are invalid. Even worse, if the estimator fails during operation due to the lack of features in the environments, existing nonlinear solvers will fail completely since no controlled initialization procedure can be performed. However, it is desirable to have the estimator recover from failure if later sufficient information is gathered. In fact, initialization and failure recovery are equivalent since both of them require that the estimator to (re)initialize from an unknown state.

Pioneering work on state estimation without initialization is proposed in [60, 61], where the authors proposed to perform the estimation in the body frame of the first pose in the sliding window. An IMU pre-integration technique is proposed to handle multi-rate sensor measurements. The authors show that the nonlinearity of the system mainly arises only from rotation drift. Recent results suggests that by assuming the orientation is known, VINS may be solved in a linear closed-form [17, 45, 56, 66, 67]. It has been shown that both the initial gravity vector and the body frame velocity can be estimated linearly. These results have significant implications that a good initialization of the VINS problem may be actually not necessary. In particular, [66, 67] analytically show conditions from which initial values are solvable. However, [45] is limited to use a fixed small number of IMU measurements, which makes it very sensitive to IMU noise. Approaches that utilize multiple IMU measurements in a sliding window [17, 56, 66, 67] do not scale well to a large number of IMU measurements since they relies on double integration of accelerometer output over an extended period of time. Moreover, these closed-form approaches do not take the noise characteristic of the system into account, which lead to sub-optimal results.

2.7 Autonomous Environment Coverage

Exploration is a well-known problem in the field of mobile robotics. A traditional exploration approaches is frontier-based exploration that define locations in the map that,

if visited, reduce environment uncertainty and lead to full environment coverage. Traditional 2D approaches rely upon a dense occupancy grid representation of the world and knowledge of both the known (occupied and unoccupied) and unknown cells in the map.

Frontier-based exploration approaches generally compute exploration frontiers as the discrete boundary between the unoccupied and unknown regions of the current environment estimate. Since the introduction of frontier-based exploration [116], several algorithmic variations have been proposed to improve the exploration performance [9, 26, 112] with an evaluation of several different methods in [31]. These variations focus on improving the expected observation area as compared to the path execution cost. Extensions enabling cooperative multi-robot frontier-based exploration extend this notion by optimizing this cost across multiple vehicles [64, 105]. Variations on this theme improve performance by considering energy efficiency [68], localization accuracy [63], frontier clustering [74], and map segmentation [13].

Extensions of frontier-based exploration methods to 3D are proposed in the literature through the integration of elevation or octree map structures [25, 36] and simplified polygonal approximations [79]. In [22], 3D volumetric data is projected to a 2D information grid to generate candidate points for exploration. While these methods consider 3D environments, the approaches are focused on exploration for ground vehicles and ultimately determine candidate frontier locations assuming a planar mobility model.

The direct 3D extension of frontier-based exploration given a dense voxel grid representation of the environment and the integration of frontier voids are proposed in [28] and [18], respectively. In [91], 3D frontiers are integrated with a vector field-based approach to obtain shorter exploration trajectories. The 3D exploration problem is also considered by the vision community in the context of 3D surface exploration. Exploration goals are identified by detecting range discontinuities or openings in the observed surface which guide the exploration process [90, 111]. However, these methods require considerable computational and memory resources as evidenced by the analysis in each of these works and are unable to operate in real-time on a computationally-constrained

MAV.

Autonomous indoor exploration with MAVs using 2D frontier-based exploration approaches is presented in [4, 82]. In these works, the MAV operates in 3D but navigates with respect to a 2D occupancy grid map. The naive extension of such methods to three dimensions introduces several challenges when considering systems with payload constraints. The primary challenge is the inability of onboard sensors to faithfully provide information about known and unknown environment regions in three dimensions. Due to limited available power, our MAV is only able to carry a restricted set of sensors such as lasers and cameras (mono, stereo, or RGB-D) that fit within the payload capacity of the vehicle. However, scanning lasers provide only partial information about the three-dimensional structure of the surrounding environment. The fact that any three-dimensional information is available from a laser rigidly mounted to a MAV is a consequence of the motion of the MAV during flight. Cameras suffer from similar limitations because of limited field of view. Another challenge resulting from payload constraints is limited onboard processing. Even though processors and memory are getting faster and less expensive, we are still limited by the lack of low-power, low-mass onboard computation for the processing of sensory information.

Chapter 3

Laser-Based Autonomous Indoor Navigation

This chapter presents our first successful attempt in developing a self-contained autonomous indoor MAV using a laser scanner as the main sensor. We are interested in real-time autonomous navigation in multi-floor indoor environments using an aerial vehicle with pragmatic constraints on available computational resources (speed and memory). Therefore, we address the problems of mapping, localization, planning, and control given these system requirements.

We note that this topic is addressed by others in the community with some similarities in approach and methodology with results toward online autonomous navigation and exploration with an aerial vehicle [5, 20, 30]. The major point of differentiation between our work and existing results is that our system is capable of autonomous navigation without need of external infrastructure. We ensure that the system is able to robustly operate in challenging multi-floor environments where reliance on external processing or sustained communication with the vehicle is infeasible. We present results from field experiments in multiple environments and a live demonstration that required that the system operate without failure over an extended period of time.

In this chapter, we focus on the discuss on perception, state estimation, and mapping

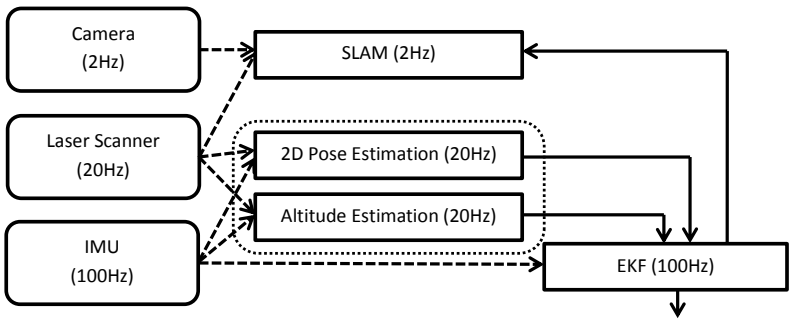


Figure 3.1: Diagram of laser-based state estimation. Dotted box shows the estimator module that provides relative measurement in Ch. 5.

methodologies (Fig. 3.1) used in the system. We present autonomous navigation results from systems powered by these approaches. We note that autonomous navigation involves careful integration of other important components such as planning, control, and user interface. However, we would like to defer the discuss of these components and the integration to Ch. 7 as most of them are adaptations of existing results in their specific areas. These components are also used as common building blocks to support and demonstrate methodologies to be presented in the next few chapters (Ch. 4, 5, 6).

3.1 Pose Estimation

Central to our work is a reliable estimation module that outputs 6-DOF pose using a 2D laser scanner and an low cost IMU and without any prior knowledge of the environment. We note that a 2D laser scanner is insufficient to estimate all parameters in 6-DOF, as such, we propose a decoupled estimator design by utilizing certain assumptions of the environment.

We define vectors in the world and body frames as $(\cdot)^w$ and $(\cdot)^b$ respectively. We are interested in the 6-DOF pose of the MAV in the world frame, which is defined as: $[x^w,\, y^w,\, z^w,\, \psi^w,\, \theta^w,\, \phi^w]$. where ψ^w, θ^w, and ϕ^w are yaw, pitch, and roll Euler angles representing the rotation from the body frame to the world frame following a ZYX Euler

angle conversion. Accordingly, we have the rotation matrix representation:

$$\mathbf{R}_b^w = \mathbf{R}(\psi^w) \cdot \mathbf{R}(\theta^w) \cdot \mathbf{R}(\phi^w)$$

We use a 2D laser-based pose estimation approach to estimate the 2D position and yaw (ψ^w). A Kalman filter (KF) with state augmentation fuses IMU data with redirected laser scans to provide altitude estimates (z^w). The remaining state variables, ϕ^w and θ^w, are output directly from the onboard IMU.

3.1.1 Assumption

As the domain of interest is indoor environments and periphery, we assume 2.5D environment models formed by collections of vertical walls and horizontal ground planes, all assumed to be piecewise constant. Let $\left[{}^s x^b,\ {}^s y^b,\ 0\right]^{\mathrm{T}}$ be the laser scan endpoints in the body frame. We can project the laser scans to a horizontal plane that is perpendicular to the gravity vector ($\mathbf{g}^w = [0,\ 0,\ g]^{\mathrm{T}}$) by:

$$\begin{bmatrix} {}^s x^g \\ {}^s y^g \\ {}^s z^g \end{bmatrix} = \mathbf{R}(\theta^w) \cdot \mathbf{R}(\phi^w) \cdot \begin{bmatrix} {}^s x^b \\ {}^s y^b \\ 0 \end{bmatrix}$$

We eliminate the scans that hit the floor or ceiling, which are generally not useful for scan matching, by comparing ${}^s z^g$ with the redirected laser scans pointing upward and downward. Although this approach largely simplifies the challenges of full 3D scan matching using only 2D laser range finders, the 2.5D environment assumption is violated in cluttered and outdoor environments with natural structure. In our experiments we see that partial structure in the environment satisfies this assumption and the overall performance is acceptable.

3.1.2 2D Pose Estimation

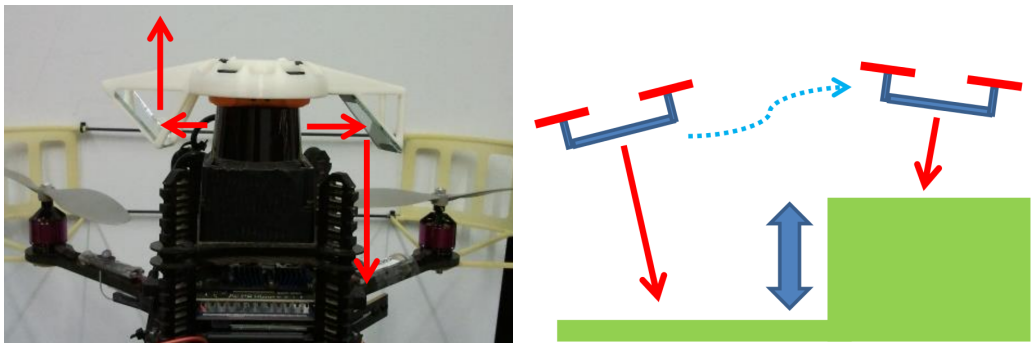
A horizontally mounted scanning serves as a primary source of information for 2D position and yaw estimation. We evaluated several laser-based methods for pose estimation

such as exhaustive search [80] and feature-based approaches [109]. However, as the vehicle dynamics require pose estimates with update rates of 20 Hz and our limited onboard computational resources, we chose the Iterative Closest Point (ICP) algorithm, which yields a robust and inexpensive continuous pose estimate. We make use of a grid-based search [12] to speed up the computationally expensive closest point search in ICP.

3.1.3 Altitude Estimation

We retrofit the laser with mirrors for beam redirection to the floor and ceiling for altitude measurement (Fig. 3.2(a)). A KF with altitude and vertical velocity as state is used for fusing IMU laser readings and detect surface discontinuities (Fig. 3.2(b)). We measure the variation in altitude with scans pointing to the floor and use this value to approximate the variance used in the measurement update of the KF. If the variation is too high, generally due to uneven surfaces or during a floor level change, we discard the current laser measurement and defer the measurement update until a stable measurement is obtained. An additional mirror deflects scans vertically upward toward the ceiling. These measurements provide additional robustness should there be no stable downward measurements available. When no upward or downward laser measurements are available, we use an onboard pressure sensor for short term measurement updates and force the MAV to lower its height.

A more principled way of fusing laser and pressure measurement together with other nonlinear measurements that also provides height estimates (e.g. visual odometry) will be presented in Ch. 5. Corrections of accumulated errors caused by frequent floor transitions are resolved through loop closure.



(a) (b)

Figure 3.2: Laser mount with mirrors for beam redirection to the floor and ceiling for laser-based altitude estimation with a KF (Fig. 3.2(a)). Surface discontinuity is detected by the differences between process (dotted) and measurement (solid) updates (Fig. 3.2(b)).

3.2 EKF-based Sensor Fusion for Control

To ensure high-rate, accurate, and drift-free pose estimates for feedback control, we use an Extended Kalman Filters (EKF) to fuse and boost the pose estimate to $100\,\text{Hz}$. The EKF combines the $20\,\text{Hz}$ pose estimate from scan matching and the altitude estimator and the $100\,\text{Hz}$ IMU data to provide $100\,\text{Hz}$ pose and linear velocity estimates in the world frame. The final estimation output has an average delay of $0.01\,\text{s}$ and feeds directly into the feedback control loop of the robot for position and velocity control.

The state vector of this EKF is defined as:

$$\mathbf{x}_t = \left[\mathbf{p}_t^w,\ \mathbf{\Phi}_t^w,\ \dot{\mathbf{p}}_t^b,\ {}^a\mathbf{b}_t^b,\ {}^\omega\mathbf{b}_t^b\right]^{\mathrm{T}}$$

where $\mathbf{p}_t^w = [x_t^w,\ y_t^w,\ z_t^w]^{\mathrm{T}}$ and $\mathbf{\Phi}_t^w = [\psi_t^w,\ \theta_t^w,\ \phi_t^w]^{\mathrm{T}}$ are position and orientation in the world frame at time t. Note that $\mathbf{p}_t^w$ can also be interpreted as the position of the body frame at time t with respect to the world frame $(\cdot)^w$, other parameters also follow similar interpretation. ${}^a\mathbf{b}_t^b$ and ${}^\omega\mathbf{b}_t^b$ are the bias of the accelerometer and gyroscope, both

expressed in the body frame. We consider an IMU-based state propagation model:

$$\mathbf{u}_t = \left[\mathbf{a}_t^b,\, \omega_t^b\right]^{\mathrm{T}}$$
$$\mathbf{v}_t = \left[{}^{a}\mathbf{v}_t,\, {}^{\omega}\mathbf{v}_t,\, {}^{\mathbf{b}}\mathbf{v}_t\right]^{\mathrm{T}}$$
$$\mathbf{x}_{t+1} = f(\mathbf{x}_t,\, \mathbf{u}_t,\, \mathbf{v}_t)$$

where $\mathbf{u}_t$ is the measurement of linear accelerations and angular velocities from the IMU in the body frame. $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{D}_t) \in \mathbb{R}^{12}$ is the process noise. ${}^{a}\mathbf{v}_t$ and ${}^{\omega}\mathbf{v}_t$ represent additive noise associated with the gyroscope and the accelerometer. ${}^{\mathbf{b}}\mathbf{v}_t$ models the Gaussian random walk of the gyroscope, accelerometer bias. The function $f(\cdot)$ is a discretized version of the continuous time dynamical equation [46].

The pose estimate from the laser-based pose estimator is first transformed to the IMU frame before being used for the measurement update. The measurement model is linear and can be written as:

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{n}_t$$

where $\mathbf{H}$ extracts the 6-DOF pose in the state and $\mathbf{n}_t$ is additive Gaussian noise. Since the measurement model is linear and the measurement update can be performed via a KF update step.

Note that this EKF is later developed into a UKF-based multi-sensor fusion framework to combine information from not only laser, but also multiple heterogeneous sensors, in a consistent way (Ch. 5).

3.3 Simultaneous Localization and Mapping

We address the problems of mapping and drift compensation via an incremental simultaneous localization and mapping (SLAM) algorithm. Given the incremental motion of the platform provided by ICP-based scan matching and the IMU, we correct the error in $\{x,\, y,\, \psi\}$ by aligning incoming laser scans against the existing map using a windowed exhaustive grid search. The cost-map generated from the obstacles in the existing map

are approximated by an image distance transform. If a stable floor transition is detected by the pose estimator, we create a new layer in a multi-layered occupancy grid. The incremental SLAM algorithm runs at $10\,\text{Hz}$ and consumes less than 30% of the total CPU time.

To correct the global inconsistency caused by incremental SLAM, we employ vision-based techniques to enable robust loop closure detection that does not depend on the actual pose estimation error [14]. A fixed-size visual vocabulary is constructed offline by clustering a large number of SURF features collected from different environments [6]. The detected SURF features of each incoming image are converted into the vocabulary format and matched against previous images via histogram voting. Any matched loop closure candidates are further verified by scan matching. Loop closures add constraints to a pose graph, where each node in the graph is a sparse sample of the vehicle poses and their associated sensor data. We apply batch graph optimization [16] to create a globally consistent multi-floor map and pose graph. The optimization occurs in the full 6-DOF pose space with the assumption that closure only happens at the same floor level.

Note that as shown in Fig. 3.1, the SLAM module runs in a separate process at a much lower frequency. This way, any latency in the SLAM module will not break the overall feedback control loop.

3.3.1 Environment Representation

Given the estimated global pose of the MAV, we can transform laser scans accordingly and produce a dense 3D environment representation for planning and obstacle avoidance. However, the amount of onboard memory is limited in mobile processors like ours. As we are primarily interested in indoor environments, the majority of obstacles take the form of vertical walls. Therefore, we use a modified multi-volume occupancy grid map, similar to [19], to create a compact occupied space representation by merging contiguous occupied cells into common vertical regions. For general indoor environments, the resulting map typically has a memory cost on the same order as a 2D occupancy grid

map.

3.4 Experimental Results

Three experiments are presented: (1) a study of the estimator performance compared to ground truth data; (2) navigation in confined multi-floor indoor environments; (3) large scale mapping across multiple floors; The motivation for each study is stated in the respective discussions. The configuration of the experimental platform is discussed in Sect. 1.3 and shown in Fig. 1.2(a). Planning, trajectory generation and control methodologies are presented in Ch. 7.

3.4.1 Evaluating Estimator Performance

We wish to study the performance of the onboard estimator as compared to ground truth, where ground truth is defined by a sub-millimeter accurate Vicon motion tracking system[1]. Two studies are presented. The first study considers the accuracy of the onboard estimate compared to the Vicon estimate while the MAV flies along a specified trajectory (Fig. 3.3). In this case, the Vicon and onboard estimate compare well with a standard deviation of $\{\sigma_x,\, \sigma_y,\, \sigma_z\} = \{2.47,\, 3.23,\, 0.70\}$ and $\sigma_\psi = 0.55$ (units in cm and deg, respectively). The second study compares feedback control to maintain a single position using the pose and velocity feedback from the onboard estimator (Fig. 3.4). The standard deviations of the hover performance are $\{\sigma_x,\, \sigma_y,\, \sigma_z\} = \{4.42,\, 2.79,\, 1.76\}$ (units in cm).

3.4.2 Navigation in Confined Multi-Floor Indoor Environments

We now consider autonomous navigation in confined indoor environments with single or multiple floors. To verify map consistency, environments with small loops are consid-

[1] http://www.vicon.com

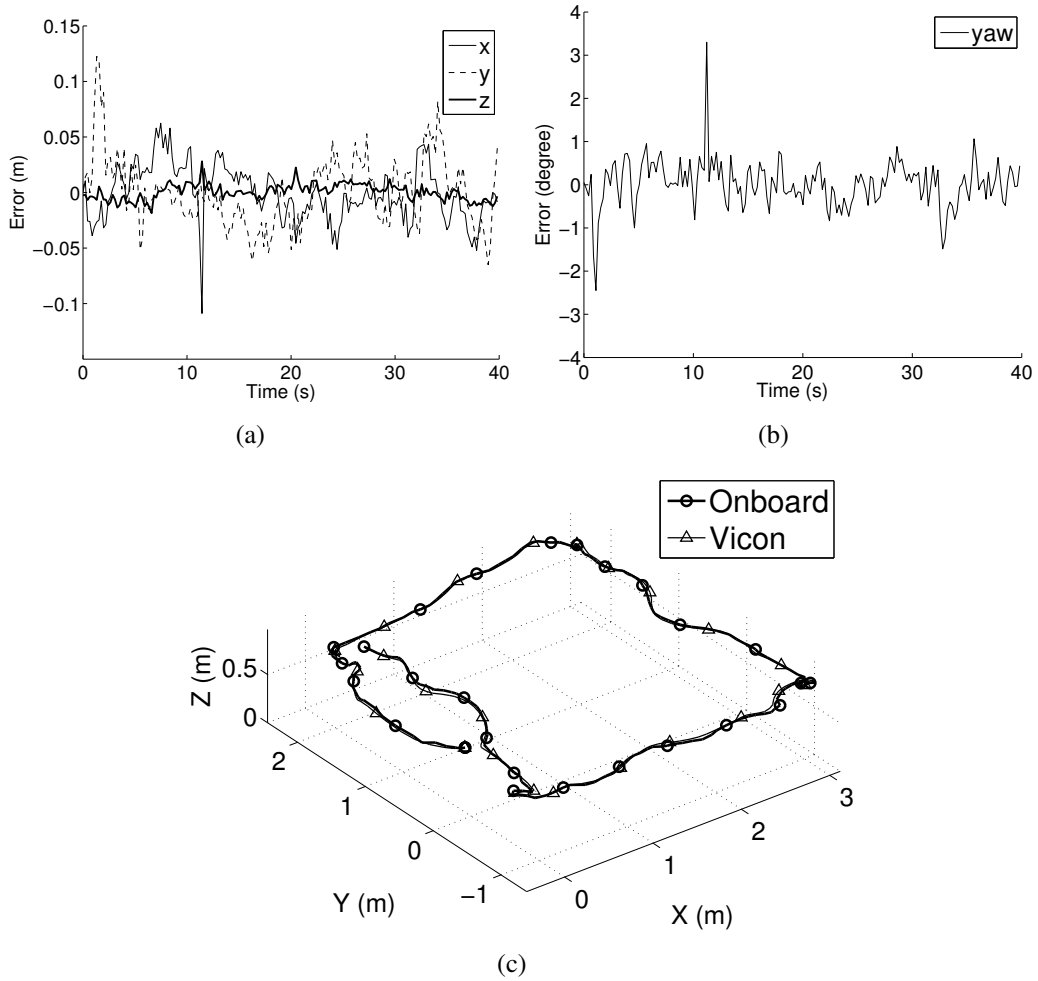
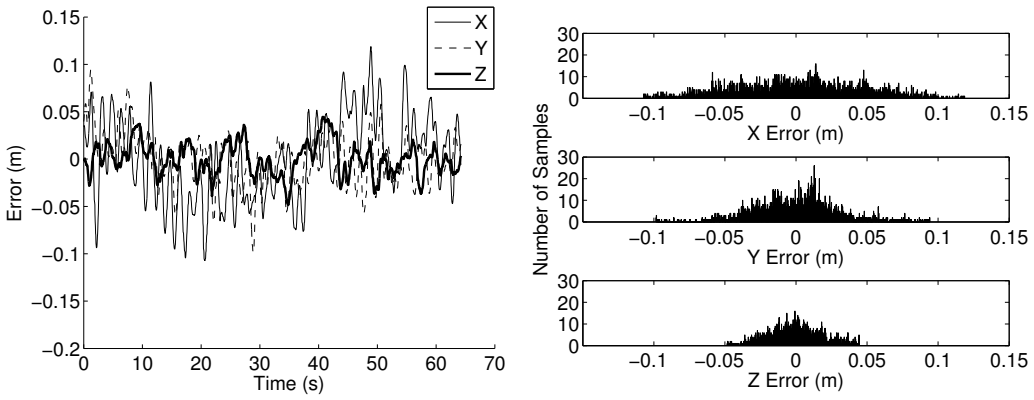


Figure 3.3: Error between estimates from Vicon and our laser-based onboard estimator (Figs. 3.3(a)–3.3(b)) while the MAV autonomously tracks a specified trajectory (Fig. 3.3(c)).



(a) Error over time

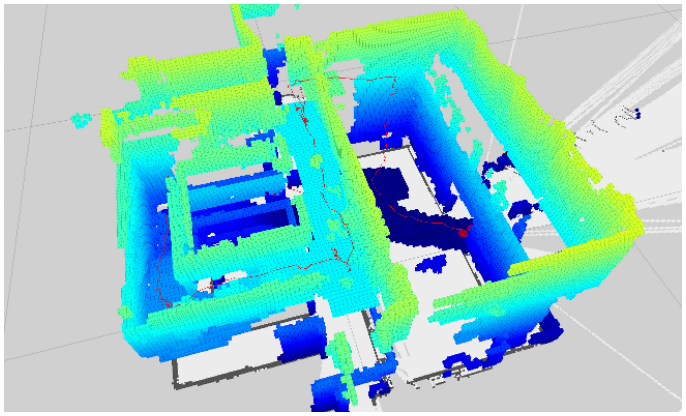
(b) Error distribution

Figure 3.4: The MAV is commanded to hover based on feedback from the onboard estimator. The resulting regulation error from the closed-loop system and its distribution are shown.

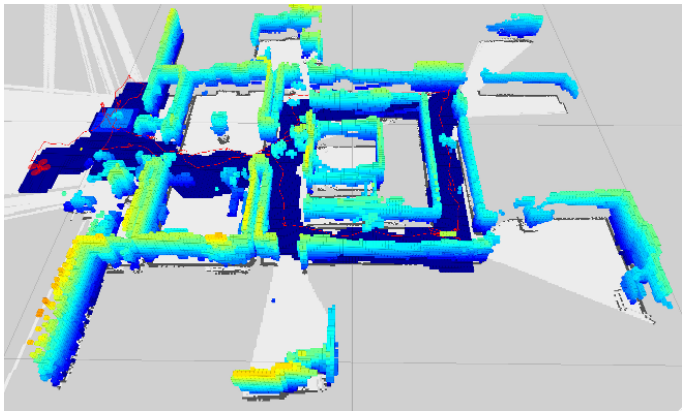
ered. While the quadrotor is autonomous, the planned trajectory is determined by a user specifying goal locations with respect to the map.

In Fig. 3.5(a), we see the map generated by the MAV navigating through an indoor two-floor environment. The vehicle starts on the first floor and transitions to the second floor via a stairwell (Fig. 3.6(a)). The vehicle returns to the starting location on the first floor by flying through a window into an open lobby (Fig. 3.6(b)). The vehicle side clearance when passing through the window is approximately 5 cm. While we do not have ground truth for this environment, we observe that there is minimal error or drift in the map. The MAV is commanded to return and hover at its starting location. The error was observed to be on the order of a couple of centimeters.

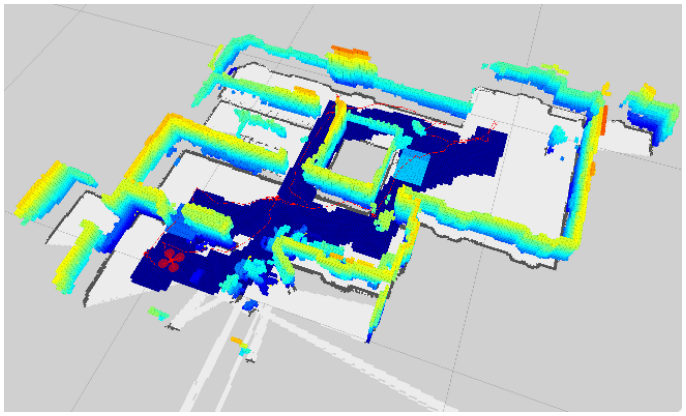
To evaluate loop closure, we consider the two buildings in Figs. 3.5(b)-3.5(c). In both environments, the vehicle starts outside and navigates through doorways, hallways, and windows, to generate 3D maps. We see in Figs. 3.6(e)-3.6(f)) that the vehicle successfully navigates through multiple doorways and around furniture and other environmental clutter. In both cases, loops in the final maps are successfully recovered and empirically appear to be consistent. The trajectory length and duration for each experiment is noted



(a) Trajectory length: 65.9 m, duration: 273 s



(b) Trajectory length: 99.7 m, duration: 340 s



(c) Trajectory length: 52.1 m, duration: 200 s

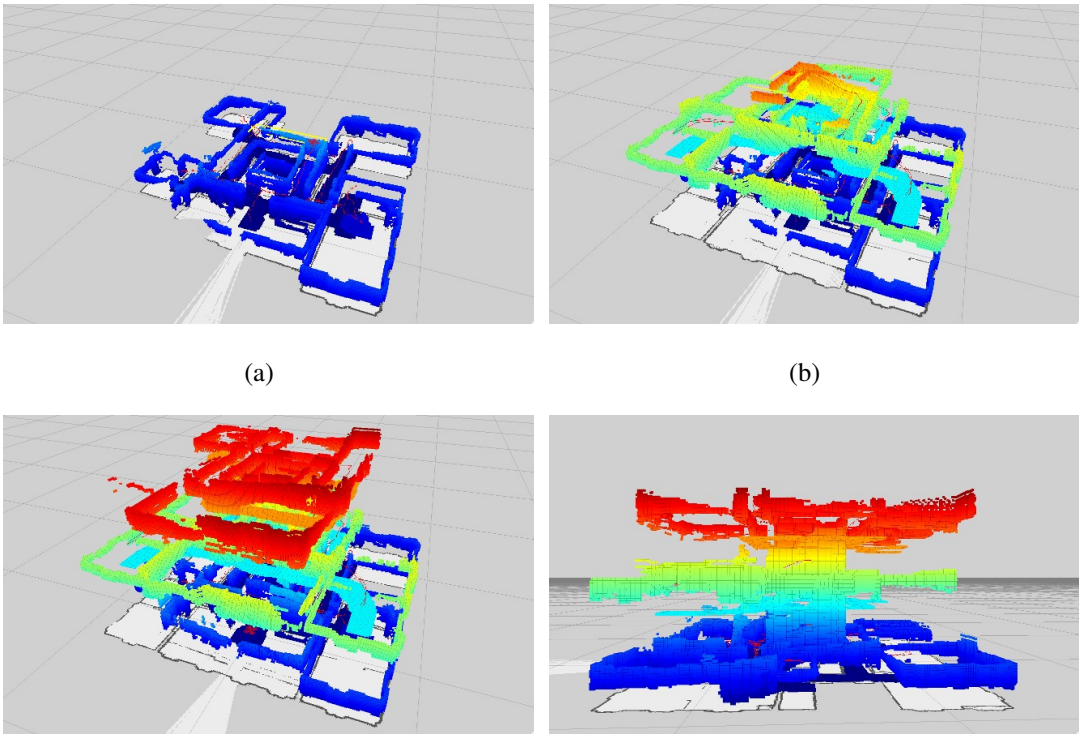
Figure 3.5: Maps generated during autonomous navigation across multiple floors (Fig. 3.5(a)) and through confined environments with small loops (Fig. 3.5(b)-3.5(c)). The vehicle and its trajectory are depicted as a red mesh and line, respectively.

(a) (b)

(c) (d)

(e) (f)

Figure 3.6: Images of the vehicle autonomously navigating between multiple floors (Figs. 3.6(a)-3.6(b)) and in confined indoor environments (Figs. 3.6(c)-3.6(f)).



(a) (b)

(c) (d)

Figure 3.7: Map generation across three floors of an indoor environment.

in Fig. 3.5.

3.4.3 Large Scale Mapping Across Multiple Floors

We now consider the performance of the onboard SLAM and loop-closure algorithms when mapping across several floors. To pursue a large scale experiment with a length that exceeds feasible flight time, we carry the vehicle such that it emulates flight. Note that the map is consistent throughout the experiment and the robot is able to close the loop when it returns to its starting location after mapping a three-story building with appearance the same as Figs. 3.6(c)- 3.6(d). Maps at multiple stages of the experiment are shown in Fig. 3.7.

3.4.4 Public Demonstration

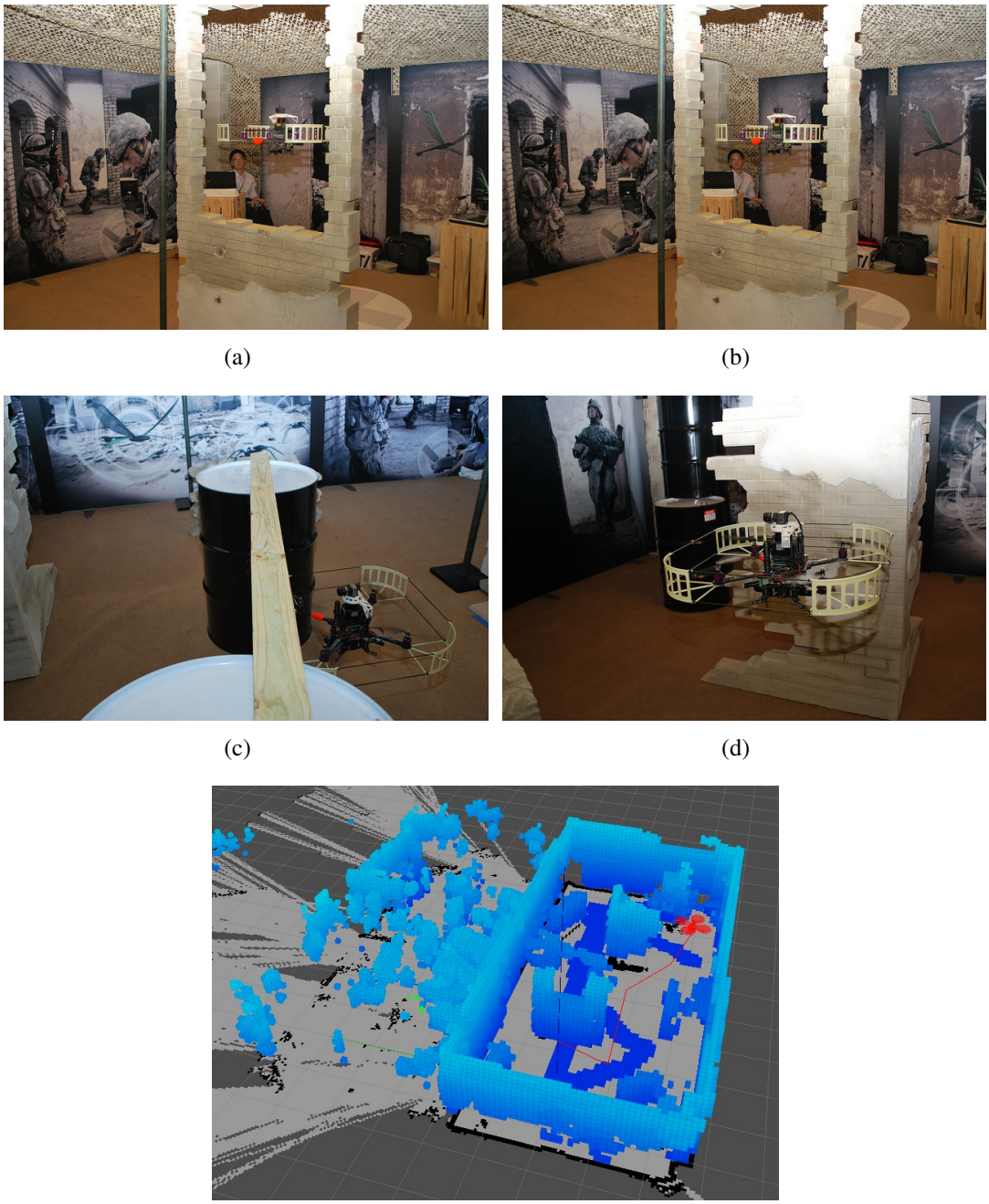
As a final discussion on the performance of our laser-based autonomous navigation system, we briefly report results from a live demonstration held at the 27^{th} Army Science Conference in Orlando, FL, USA, Nov. 29 - Dec. 2, 2010. We report these results to highlight the repeatable performance of the system.

The demonstration considered autonomous navigation through a maze-like indoor environment following the size and layout shown in Fig. 3.8(e). During each demonstration run, the vehicle began with no prior information and concurrently built a 3D map while navigating through the maze-like environment. Dynamic obstacles (people) were introduced into the environment with the intention of disrupting the planned trajectories of the vehicle. The goal of the demonstration was to show that the autonomous system was able to effectively navigate the environment while avoiding collisions with static and dynamic obstacles (Fig. 3.8). The trajectory of the vehicle was defined by waypoints provided by an external operator.

Over fifty trials occurred during the live demonstration without any system failures. The vehicle successfully navigated the environment without issue and avoided all obstacles (static and dynamic).

3.5 Discussion

In this chapter, we show that it is possible to have a computationally-constrained MAV to fly indoor fully autonomously using a laser range finder as the main sensor. The key idea behind this is to assume that the environment is 2.5D and project all laser scans into a common ground pland for scan matching. We also carefully decouple system components that require low latency performance and components that aim to provide global consistency but with possibly large delays. This way, the fast dynamics of the MAV can always be stabilized with fast but less accurate state feedback.



(a) (b)

(c) (d)

(e)

Figure 3.8: The demonstration environment with windows, walls, and various types of clutter. A 3D map of the demonstration site generated during an autonomous flight is shown in Fig. 3.8(e).

We note that all experimental results are obtained in environments that in general satisfy the 2.5D assumption (Sect. 3.1.1). When the environment is complex, we see frequent failure of the laser-based approach (Sect. 5.5.2), which motivates and justifies the development of vision-based approaches (Ch. 4), and multi-sensor fusion methodologies (Ch. 5).

Chapter 4

Vision-Based State Estimation and Autonomous Flight

In this chapter, we study how can we relax the 2.5D assumption (Sect. 3.1.1) in laser-based approaches by using cameras as the main sensor for state estimation. MAVs can, in principle, navigate quickly through 3D unstructured environments, enter and exit buildings through windows, and fly through collapsed buildings. However, it has proved to be challenging to develop small (less than 1 meter characteristic length, less than 1 kg mass) aerial vehicles that can navigate autonomously without GPS. In this work, we take a significant step in this direction by developing a quadrotor that uses a pair of cameras and an IMU for sensing and a netbook class processor for state estimation and control. The MAV weights only 740 grams and is able to reach speeds of over 10 body lengths/second. In this chapter, we detail our vision-based state estimation approach and combine it with trajectory generation and control methodologies (Sect. 7) to conduct experiments to verify the performance of the system.

The key component technology in our system is a visual-inertial state estimator that accurately tracks the pose and velocity of the quadrotor in 3D environments. We equip our quadrotor platform with two forward-facing cameras (Fig. 1.2(b) and Table. 1.1) and develop a loosely-coupled, combined monocular-stereo approach. A primary forward

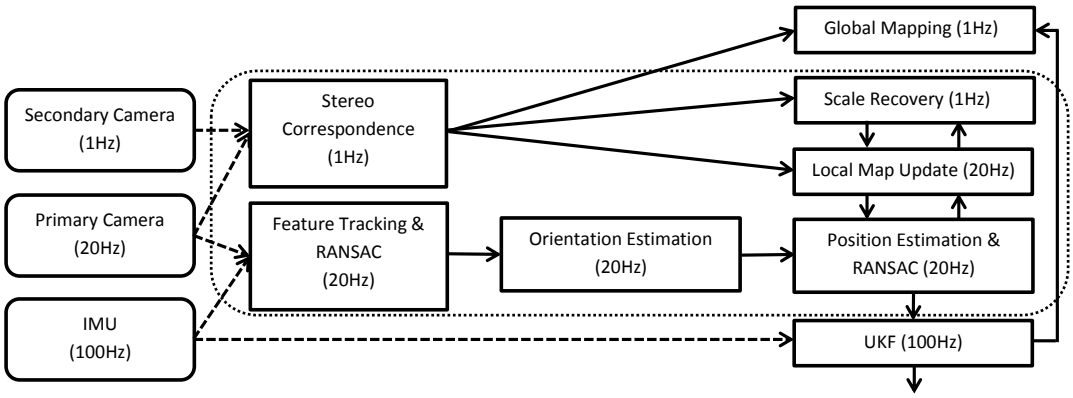


Figure 4.1: Diagram of the proposed vision-based state estimator. Note how the decoupled and asynchronous formulation saves computation power. Dotted box shows the estimator module that provides relative measurement in Ch. 5.

facing wide-angle camera operates at a high rate and supports pose estimation and local mapping, while a secondary camera operates at a low-rate and compensates for the limitations of monocular vision-based approaches. A key idea of our approach is the decoupling of different components in a visual-based estimator to achieve fast processing, as shown in Fig. 4.1. We require that visual pose estimation and map update be done at frame rate ($20\,\text{Hz}$) in order to maximize robustness to rapid changes in observable features during fast maneuvers.

The pose estimate derived from the vision system is fused with IMU information to enable feedback control. This fusion is done with a unscented Kalman filter (UKF), which is a preliminary version of the one to be presented in Ch. 5. The vision-based estimator alone does not enforce global consistency as in visual SLAM [15, 106] due to computational constraints. However, we implement a offboard global SLAM module to generate maps from low frequency stereo data and for planning and obstacle avoidance purposes.

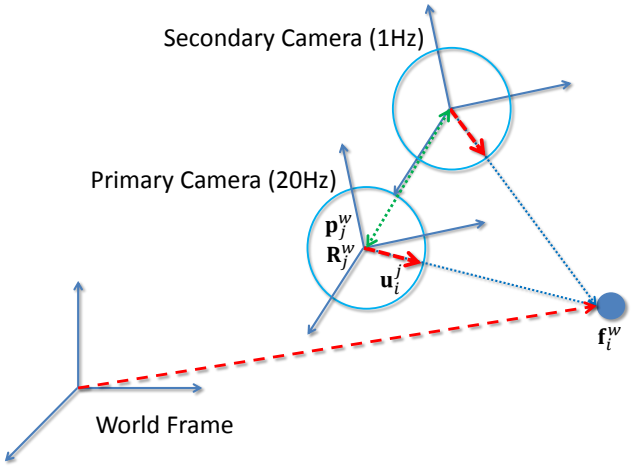


Figure 4.2: Camera geometry notation. $\mathbf{p}_j^w$ and $\mathbf{R}_j^w$ represent the j^{th} primary camera pose in the world frame and $\mathbf{f}_i^w$ is the position vector of the i^{th} feature in the world frame. $\mathbf{u}_i^j$ is unit length feature vectors in the body frame of the primary camera.

4.1 Feature Detection and Tracking

Both cameras in the system are modeled as spherical cameras and calibrated using the Omnidirection Calibration Toolbox [86]. For the primary camera that runs at 20 Hz, we detect Shi-Tomasi corners [102] and track them using the KLT tracker [59]. Due to the limited motion between image frames, We are able to perform the feature detection and tracking calculations on the distorted wide-angle image, reducing the overall computational burden. All features are transformed into unit length feature observation vectors $\mathbf{u}_i^j$ using calibration parameters. Here we denote $\mathbf{u}_i^j$ as an observation of the i^{th} feature in the j^{th} image in the camera body frame.

Following the method in [114], we remove tracking outliers by using the the estimated rotation (from short term integration of gyroscopic measurements) between two consecutive frames and unrotate the feature observation prior to applying the epipolar constraint:

$$(\mathbf{u}_i^{j-1} \times \mathbf{R}_j^{j-1}\mathbf{u}_i^j)\mathbf{t} = 0$$

where $\mathbf{R}_j^{j-1}$ is the rotation between two consecutive images estimated by integrating

gyroscope measurements, and $\mathbf{t}$ is the translation vector with unknown scale. Only two correspondences are required to solve an arbitrary scaled $\mathbf{t}$, thus a 2-point RANSAC can be used to reject outliers. This approach reaches a consensus with much fewer hypotheses compared to the traditional 5-point algorithm [78].

For the low rate stereo subsystem, candidate correspondences can be found using the KLT tracker. With calibrated stereo cameras, outlier rejection of these candidates is possible via applying epipolar geometry constraint.

4.2 Pose Estimation

In general, and especially for a monocular system, the number of features with good 3D position estimates is much smaller than the number of tracked features. Even in a stereo setting, a large number of features cannot be triangulated due to scene ambiguity. Although these "low quality" features cannot be used for position estimation, they do carry information about the orientation. Therefore, similar to [10], we decouple the orientation and position estimation subproblems.

4.2.1 Orientation Estimation

Orientation estimation is traditionally computed via the essential matrix between two consecutive images and compounding incremental rotation [108]. However, we wish to minimize rotation drift, especially for the case of hovering when the same set of features can be observed over an extended period of time. We store the index of each frame k in which feature i is observed in the set $\mathcal{J}_i \subset \mathbb{Z}_{\geq 0}$ and record its observation, $\mathbf{u}_i^k$, and the corresponding camera orientation $\mathbf{R}_k^w$. M_i denotes the frame of the index of the first observation of i and j is the current frame index. Note that M_i may be different for each feature. We maintain all features in an ascending order according to M_i (Fig. 4.3).

We pick all features that have at least T_j observations for orientation estimation. T_j

is determined by:

$$T_j = T_{j-1} + 1 - D_j$$

where the integer $D_j \in [0, T_{j-1}]$ is the minimum number of observation reduction that makes the estimated essential matrix well-posed. In other words, we require the singular values of the essential matrix to be close to $[\sqrt{2}, \sqrt{2}, 0]$. D_j is found in a brute force manner. However, if the MAV is hovering, D_j is likely to be zero as there are no large changes in the distribution of feature observations. On the other hand, fast motions can result in $D_j = T_{j-1}$ and only consecutive frames can be used for orientation estimation due to rapid changes in the feature distribution. As T_{j-1} is likely to be one in this case, the computation overhead of this brute force search is limited.

We denote the index of the last feature that has at least T_j observations as n. The image index M_n and its corresponding camera orientation $\mathbf{R}^w_{M_n}$ are used as a reference, via the 5-point algorithm [78], to estimate the essential matrix $E_{M_n,j}$ and then the rotation $\mathbf{R}^{M_n}_j$ between the M^{th}_n image and the current image j. Therefore the current orientation can be written as:

$$\mathbf{R}^w_j = \mathbf{R}^w_{M_n} \mathbf{R}^{M_n}_j.$$

We require that the onboard attitude estimate be aligned with the inertial frame and therefore employ a common IMU design strategy where drift in the vision-based attitude estimate (roll and pitch) is eliminated via fusion with accelerometer measurements. This approach assumes that the vehicle state is near hover or at a constant velocity. However, fast vehicle motions can invalidate these assumptions. In this work, we find that applying small weights to accelerometer measurements yields a reasonable estimate (Fig. 4.6) given small drift in the vision-based attitude estimate.

4.2.2 Position Estimation

We begin by assuming a known 3D local feature map and describe the maintenance of this map in the next subsection. Given observations of a local map consisting of known

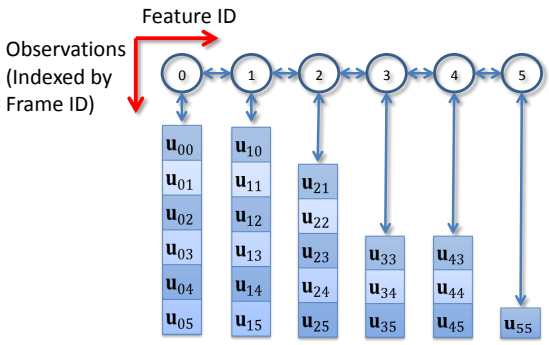


Figure 4.3: Data structure for feature storage. Features are managed in a linked list and newly added features are added to the end of the list. For every feature, all observations are recorded in pre-allocated memory. Feature deletion, addition, as well as the lookup of observations of a given feature can be performed in constant time.

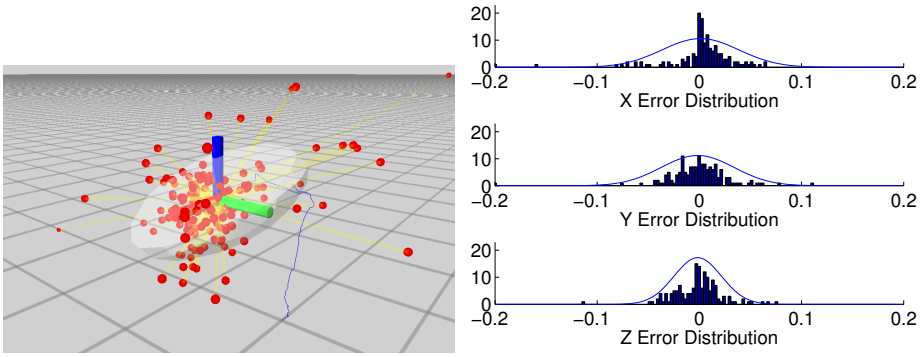
3D features, and assuming that this local map is noiseless, the 3D position of the camera can be estimated by minimizing the sum-of-square sine of angle error of the observed features:

$$\mathbf{p}_j^w = \underset{\mathbf{p}_j^w}{\operatorname{argmin}} \sum_{i \in \mathcal{I}} \left\| \frac{\mathbf{p}_j^w - \mathbf{f}_i^w}{\|\mathbf{p}_j^w - \mathbf{f}_i^w\|} \times \mathbf{R}_j^w \mathbf{u}_i^j \right\|^2 \tag{4.1}$$

where, as shown in Fig. 4.2, $\mathbf{p}_j^w$ is the 3D position of primary camera in the world frame when the j^{th} image is captured, $\mathcal{I}$ represents the set of features observed in the j^{th} image, and $\mathbf{f}_i^w$ is the 3D position of the i^{th} feature in the world frame. $\mathbf{R}_j^w$, which is the estimated rotation of the primary camera, is treated as a known quantity while doing position estimation. Note that (4.1) is nonlinear. However, since the change of feature distance between two consecutive images is small, we can approximate (4.1) and solve the camera position $\mathbf{p}_j^w$ via the following linear system:

$$\left(\sum_{i \in \mathcal{I}} \frac{\mathbb{I}_3 - \mathbf{u}_i^w \mathbf{u}_i^{w\mathrm{T}}}{\|\mathbf{p}_{j-1}^w - \mathbf{f}_i^w\|^2} \right) \mathbf{p}_j^w = \sum_{i \in \mathcal{I}} \frac{\mathbb{I}_3 - \mathbf{u}_i^w \mathbf{u}_i^{w\mathrm{T}}}{\|\mathbf{p}_{j-1}^w - \mathbf{f}_i^w\|^2} \mathbf{f}_i^w \tag{4.2}$$

where $\mathbf{u}_i^w \triangleq \mathbf{R}_j^w \mathbf{u}_i^j$. Equation (4.2) always represents three equations in three unknowns, regardless of the number of observed features. Therefore, the position estimation problem can be solved efficiently in constant time.



(a) (b)

Figure 4.4: The 3D distribution of $\mathbf{e}_{ij}$ and the ellipsoid of the best fit Gaussian distribution (Fig. 4.4(a)). The error distribution histogram for each axis is shown in Fig. 4.4(b)).

In our formulation, position estimation is essentially an intersection of multiple rays. We can therefore represent the localization error via the statistical distribution of the ray-to-robot distance. We experimentally verify that this distribution can be be approximate by a 3D Gaussian distribution (Fig. 4.4). The covariance for position estimation at the j^{th} frame is obtained as:

$$\Sigma_{\mathbf{p}_j^w} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mathbf{e}_{ij} {\mathbf{e}_{ij}}^{\mathrm{T}}$$

$$\mathbf{e}_{ij} = \left(\mathbf{p}_j^w - \mathbf{f}_i^w\right) \times \mathbf{u}_i^w \times \mathbf{u}_i^w.$$

We apply a second RANSAC to further remove outliers that cannot be removed from the epipolar constraint check (Sect. 4.1). A minimum of two feature correspondences are required to solve this linear system. As such, an efficient 2-point RANSAC can be applied for outlier rejection.

4.3 Mapping

As a iterative process, we update the local map and recovery its metric scale given pose estimates. To assist the navigation system to perform path planning and obstacle avoid-

ance, similar to Sect. 3.3, we also implement a SLAM module with a number of modifications to accommodate the characteristic of cameras.

4.3.1 Local Map Update

As stated in Sect. 4.2.2, a map consisting of 3D features is required to estimate the position of the camera. We approach the local mapping problem as an iterative procedure where the pose of the camera ($\mathbf{p}_j^w$ and $\mathbf{R}_j^w$) is assumed to be a noiseless quantity. We do not perform optimizations for the position of both the camera and the features at the same time (like traditional SLAM approaches) due to CPU limitations.

We define the local map as the set of currently tracked features and cull features with lost tracking. New features are introduced into the local map when the current number of tracked features falls below a pre-defined threshold. Given $\mathcal{J}_i$, the set of observations of the i^{th} feature up to the j^{th} frame, we can formulate the 3D feature location $\mathbf{f}_i^w$ via triangulation as:

$$\mathbf{f}_i^w = \underset{\mathbf{f}_i^w}{\operatorname{argmin}} \sum_{k \in \mathcal{J}_i} \left\| (\mathbf{f}_i^w - \mathbf{p}_k^w) \times \mathbf{R}_k^w \mathbf{u}_i^k \right\|^2$$

The feature position $\mathbf{f}_i^w$ up to the j^{th} frame can be solved via the following linear system:

$$\mathbf{A}_{ij} \mathbf{f}_i^w = \mathbf{b}_{ij} \tag{4.3}$$

where

$$\mathbf{A}_{ij} = \sum_{k \in \mathcal{J}_i} \left(\mathbb{I}_3 - \mathbf{u}_i^w \mathbf{u}_i^{w\mathrm{T}} \right)$$

$$\mathbf{b}_{ij} = \sum_{k \in \mathcal{J}_i} \left(\mathbb{I}_3 - \mathbf{u}_i^w \mathbf{u}_i^{w\mathrm{T}} \right) \mathbf{p}_k^w$$

Again, it can be seen that regardless of the number of observations of a specific feature, the dimensionality of (4.3) is always three. This enables multi-view triangulation with constant computation complexity. Also, this system is memoryless, meaning that for the i^{th} feature up to the j^{th} frame, only $\mathbf{A}_{ij}$ and $\mathbf{b}_{ij}$ need to be stored, removing the need of repeated summation of observations. Moreover, the condition number or the ratio

between the maximum and minimum eigenvalues of the matrix A_{ij} gives us information about the quality of the estimate of $\mathbf{f}_i^w$. We evaluate every feature based on the condition number and reject those features with high condition numbers as unsuitable for position estimation.

4.3.2 Scale Recovery

One drawback of the above pose estimation approach is the drifting of scale due to accumulated error in the monocular-based triangulation. Here we propose a methodology that makes use of the low-rate stereo measurement to compensate scale drift. Using current observations from the primary and secondary cameras only, we can perform stereo triangulation and obtain a set of 3D points $\breve{\mathbf{f}}_k^j$ in the reference frame of the primary camera, where $k \in \mathcal{K}$ is the set of features that gives valid stereo correspondences in the current image. The ratio:

$$\tilde{\gamma} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \frac{\|\mathbf{f}_k^w - \mathbf{p}_j^w\|}{\|\breve{\mathbf{f}}_k^j\|} \tag{4.4}$$

measures the drift in scale (from $\gamma = 1$). Scaling all features according to the inverse of this ratio preserves scale consistency. However, as this measurement can be noisy, we apply a complementary filter to estimate the scale drift:

$$\gamma = (1 - \alpha)\gamma + \alpha\tilde{\gamma} \tag{4.5}$$

where $0 < \alpha \ll 1$. Hence, the proposed approach assumes that the scale *drifts* slowly. This is a major differentiation between our approach and [114], which requires that the scale changes slowly. As such, our approach is able to accommodate variations in the visual scene and resulting scale changes that can arise during fast indoor flight with a forward-facing camera.

Fig. 4.5 shows changes in γ and $\tilde{\gamma}$ during the flight of a figure eight pattern (Sect. 4.5.1). The new position of the feature $\mathbf{f}_i^w$ can be updated by modifying $\mathbf{b}_{ij}$ as (4.6) and solve the linear system (4.3) again.

$$\mathbf{b}_{ij}^+ = \frac{1}{\gamma}\mathbf{b}_{ij} - \frac{1}{\gamma}\mathbf{A}_{ij}\mathbf{p}_j^w + \mathbf{A}_{ij}\mathbf{p}_j^w \tag{4.6}$$

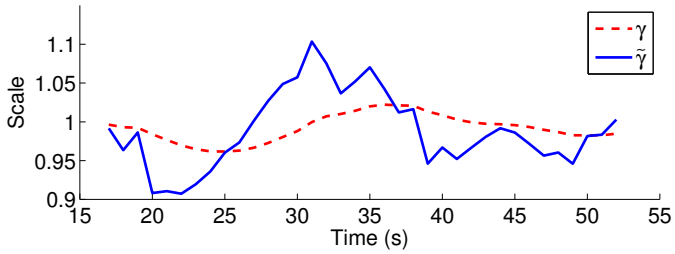


Figure 4.5: Scale changes during the flight of a trajectory in Sect. 4.5.1.

4.3.3 Global Mapping

We implement a visual SLAM module to eliminate the drift in the vision-based estimator. In our system, due to the limited onboard computation resources, limited wireless transmission bandwidth, and the accuracy of the onboard estimator, a high rate visual SLAM is both unnecessary and infeasible. Therefore, our visual SLAM module runs offboard with a maximum rate of 1 Hz. A pose graph-based SLAM back-end, together with a front-end that utilize SURF features [6] for wide baseline loop closure detection, yield robust performance at such low rates. We sparsely sample the estimated trajectory to generate nodes for the pose graph. For each node, we compute sparse 3D points by detecting and matching SURF features between the stereo images. Dense disparity images and dense point clouds are also computed.

We detect loop closures by checking nodes that fall inside the uncertainty ellipsoid of the current node. We check a constant number of nodes, starting from the earliest candidate, for possible loop closures. SURF features are used to test the similarity between two scenes. We compute the relative transform between the current node and the loop closure candidate using RANSAC PnP [23]. A rigidity test, proposed in (Sect. 3.4, [80]), is performed to verify the geometric consistency of the loop closure transform. Candidate transforms that pass the geometric verification are added to the pose graph. Graph optimization [16] is used to find the globally consistent configuration of the graph. Once an optimized pose graph is found, we can construct a 3D voxel grid map by projecting the dense point cloud to the global frame. This map is used for the high level planning

and to enable the human operator to monitor the progress of the experiment.

4.4 UKF-Based Sensor Fusion

The 20 Hz pose estimate from the vision system alone is not sufficient to control the robot. We therefore employ a UKF (Unscented Kalman filter) framework with delayed measurement compensation to estimate the pose and velocity of the robot at 100 Hz [70]. We switch from EKF as in Sect. 3.2 to UKF because of the fact that UKF gives better approximatation of the nonlinear dynamics comparing to EKF. The system state, as well as process and measurement models are identical to those in Sect. 3.2. However, for the sake of completeness of this chapter, we present the content again here:

The state vector of this UKF is defined as:

$$\mathbf{x}_t = \left[\mathbf{p}_t^w,\, \mathbf{\Phi}_t^w,\, \dot{\mathbf{p}}_t^b,\, {}^a\mathbf{b}_t^b,\, {}^\omega\mathbf{b}_t^b\right]^{\mathrm{T}}$$

where $\mathbf{p}_t^w = [x_t^w,\, y_t^w,\, z_t^w]^{\mathrm{T}}$ and $\mathbf{\Phi}_t^w = [\psi_t^w,\, \theta_t^w,\, \phi_t^w]^{\mathrm{T}}$ are position and orientation in the world frame at time t. Note that $\mathbf{p}_t^w$ can also be interpreted as the position of the body frame at time t with respect to the world frame $(\cdot)^w$, other parameters also follow similar interpretation. ${}^a\mathbf{b}_t^b$ and ${}^\omega\mathbf{b}_t^b$ are the bias of the accelerometer and gyroscope, both expressed in the body frame. We consider an IMU-based state propagation model:

$$\begin{aligned}
\mathbf{u}_t &= \left[\mathbf{a}_t^b,\, \omega_t^b\right]^{\mathrm{T}} \\
\mathbf{v}_t &= \left[{}^a\mathbf{v}_t,\, {}^\omega\mathbf{v}_t,\, {}^{\mathbf{b}}\mathbf{v}_t\right]^{\mathrm{T}} \\
\mathbf{x}_{t+1} &= f(\mathbf{x}_t,\, \mathbf{u}_t,\, \mathbf{v}_t)
\end{aligned}$$

where $\mathbf{u}_t$ is the measurement of linear accelerations and angular velocities from the IMU in the body frame. $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{D}_t) \in \mathbb{R}^{12}$ is the process noise. ${}^a\mathbf{v}_t$ and ${}^\omega\mathbf{v}_t$ represent additive noise associated with the gyroscope and the accelerometer. ${}^{\mathbf{b}}\mathbf{v}_t$ models the Gaussian random walk of the gyroscope, accelerometer bias. The function $f(\cdot)$ is a discretized version of the continuous time dynamical equation [46]. We avoid the need to estimate

the metric scale in the filter (as in [114]) through the stereo-based scale recovery noted above.

The pose estimate from the vision-based pose estimator is first transformed to the IMU frame before being used for the measurement update. The measurement model is linear and can be written as:

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{n}_t$$

where $\mathbf{H}$ extracts the 6-DOF pose in the state and $\mathbf{n}_t$ is additive Gaussian noise. Since the measurement model is linear and the measurement update can be performed via a KF update step.

This UKF is later developed into a multi-sensor fusion framework to combine information from vision, as well as other sensors, into consistent state estimates (Ch. 5).

4.5 Experimental Results

The experiment environment includes a laboratory space with ground truth motion tracking system, indoor environments with large loops, and complex but feature-rich outdoor environments. In all experiments, the MAV is autonomously controlled through the onboard generated trajectories (Ch. 7) using its onboard state estimate as feedback. The experimental platform that weights only 740 grams is discussed in Sect. 1.3 and shown in Fig. 1.2(b). The platform is based on the Hummingbird quadrotor from Ascending Technologies. This off-the-shelf platform comes with an AutoPilot board that is equipped with an IMU and an user-programmable ARM7 microcontroller. The high level computer onboard includes an Intel Atom 1.6GHz processor and 1GB RAM. The only new additions to this platform are two grayscale mvBlueFOX-MLC200w cameras with fisheye lenses. We use one camera to capture images at $20\,\text{Hz}$ as the primary camera. The secondary camera captures images at 1Hz. All camera images are at 376×240 resolution. The synchronization between cameras is ensured via hardware triggering. The total mass of the platform is 740 g. All algorithm development is in C++ using ROS as the interfacing

robotics middleware. We utilize the OpenCV library for corner extraction and tracking. The maximum number of features is set to be 300.

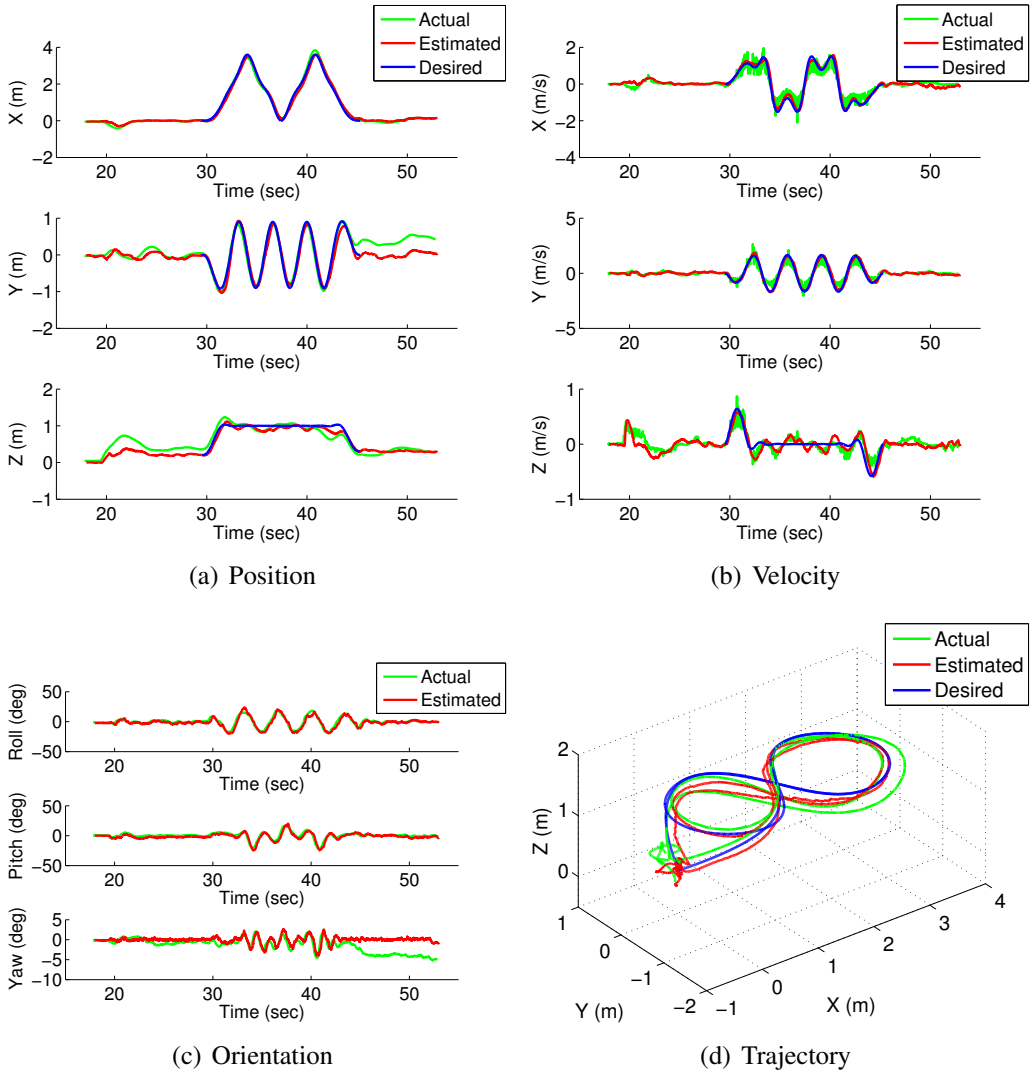
4.5.1 Autonomous Trajectory Tracking with Ground Truth

In this experiment, the MAV is programmed to fly through a figure eight pattern in which each circle in the pattern is $0.9\,\text{m}$ in radius. The maximum speed of this flight is approximately $2\,\text{m/s}$. Performance is evaluated against the ground truth from Vicon. The estimated, actual, and desired values of the trajectory, position, and velocity are shown in Fig. 4.6. Large and frequent attitude changes can be seen in the figure, as well as in the snapshots (Fig. 4.7).

Our focus is on generating state estimates that are suitable for high-speed flight, rather than generating globally consistent maps. Therefore, it makes less sense to discuss the drift in absolute position. The onboard velocity estimate, on the other hand, compares well with the Vicon estimates with standard deviation of $\{\sigma_{v_x}, \sigma_{v_y}, \sigma_{v_z}\} = \{0.1105, 0.1261, 0.0947\}$ (m/s). We can also see that the velocity profile matches well with the desired velocity. Note that the Vicon velocity estimate is obtained by a one-step numerical derivative of the position and in fact nosier than the onboard velocity estimate. It is likely that the actual velocity estimation errors are smaller than the values reported above. It should also be pointed out that the tracking error is the result of a combination of the noise of the estimator and the tracking error of the controller.

4.5.2 High Speed Straight Line Tracking

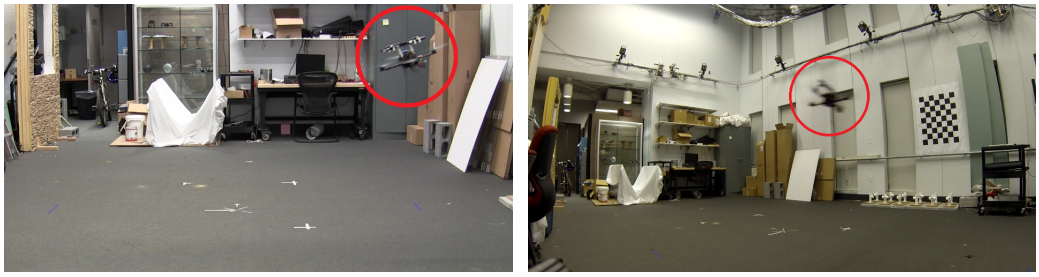
This experiment represents the highest speed that our system is able to handle. The vehicle is commanded to follow an approximately 15 m long straight line trajectory with a maximum speed of $4\,\text{m/s}$ in a environment shown in Fig 4.9(c). The estimated and desired trajectory, position, and velocity are shown in Fig. 4.8. It can be seen that the



(a) Position (b) Velocity

(c) Orientation (d) Trajectory

Figure 4.6: The MAV autonomously follow a figure eight pattern at high speed.



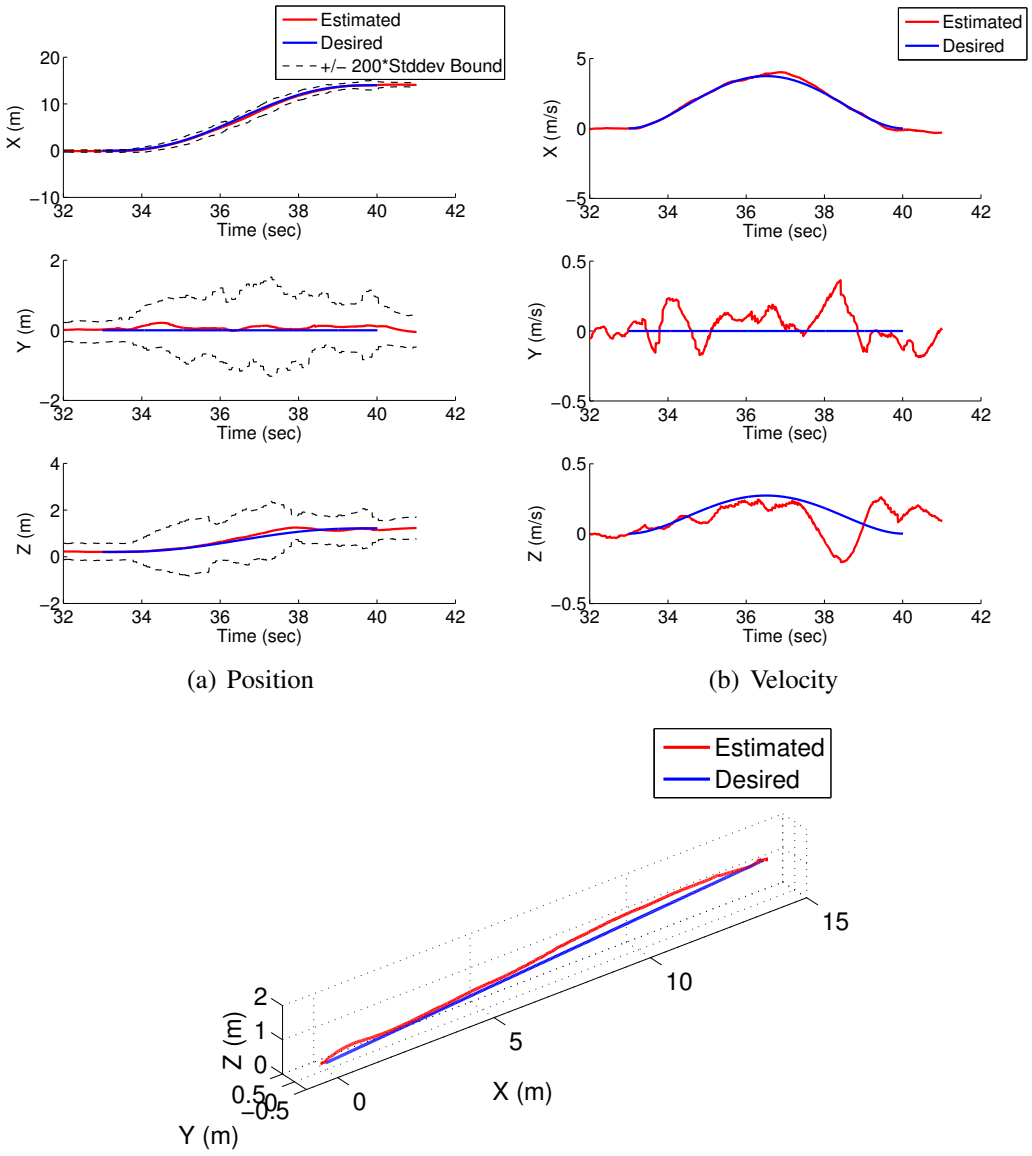
(a) (b)

Figure 4.7: Snapshots taken from different cameras of the quadrotor autonomously tracking a figure eight pattern at 2m/s . Note the large rotation of the vehicle.

estimated covariance scales with respect to the speed of the vehicle. Although we do not have ground truth for this experiment, we measure estimator performance by initially placing the vehicle in the middle of the hallway and visually verifying the drift after the trajectory is completed. The rough measurement of the drift is $\{0.5,\ 0.1,\ 0.3\}$ (m) in X, Y, and Z axes, respectively.

4.5.3 Navigation of Indoor Environments with Large Loops

We now consider a case where the robot autonomously navigates through a large-scale environment with loops. Due to the size of the loop (approximately $190\,\text{m}$), and the short battery life cycle (less than 5 min), we must achieve smooth and fast navigation in order to complete the task. The experiment is conducted with a maximum speed of over $1.5\,\text{m/s}$ and an average speed of $1\,\text{m/s}$. This is a challenging environment due to the lack of featureless (Fig. 4.10). A major loop closure is detect at $257\,\text{s}$ (Fig. 4.9(c)), during which both the SLAM pose and the 3D map change significantly (Figs. 4.9(a)-4.9(b)).

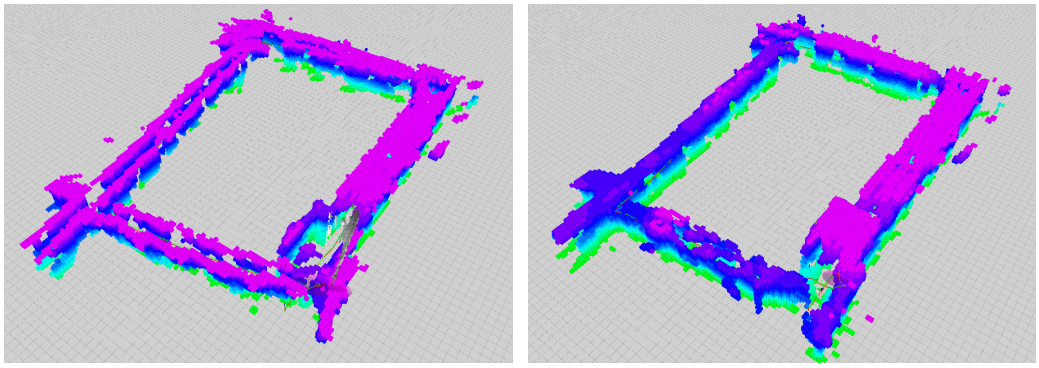


(a) Position

(b) Velocity

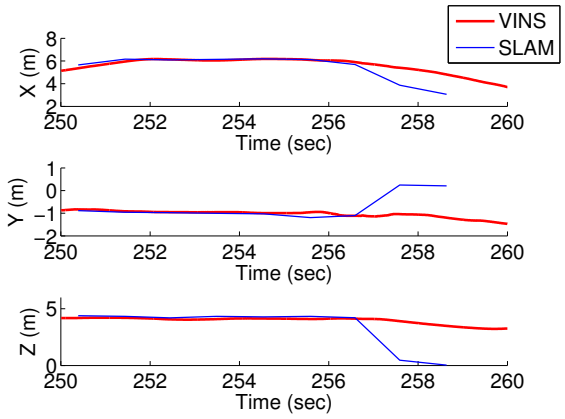
(c) Trajectory

Figure 4.8: The MAV is commanded to track a straight line at high speed. A estimated position error standard deviation is presented in Fig. 4.8(a). Note that we do not have ground truth in this figure. The plot of the estimated covariance (multiplied by 200) shows that the covariance scales with the speed of the vehicle.



(a) 3D map before loop closure

(b) 3D map after loop closure



(c) Position estimates during loop closure

Figure 4.9: Maps and estimated positions during the indoor navigation experiment. Note the siginficant corrections in the pose estimates obtained via SLAM after the loop closure (Fig. 4.9(c)).

(a) Snapshot (b) Onboard image

Figure 4.10: A snapshot of the indoor environment (Fig. 4.10(a)), together with the image captured by the onboard camera (Fig. 4.10(b)).

4.5.4 Autonomous Navigation in Complex Outdoor Environments

This experiment demonstrates the performance of the proposed system in outdoor environments. The experiment is conducted in a typical winter day at Philadelphia, PA, where the wind speed goes up to 20 km/hr. The total travel distance is approximately 170 m with a total duration of 166 s (Fig. 4.11). Representative images captured by the onboard camera are shown in Fig. 4.12. Note that the outdoor environment is largely unstructured, consisting of trees and vegetation, demonstrating the ability of the system to also operate in unstructured environments. However, we do note that this particular outdoor environment is very righ in texture, which is favorable for vision-based approaches. In practice, we may run into featureless environments that lead to failure the proposed algorithm. In addition, GPS signal with varying quality may be available in outdoor environmentswhich gives additional information for state estimation. This motivates the Ch. 5, which focuses on the development of fusing multiple heterogeneous sensors in a consistent way to improve system robustness in a wide variety of environments.

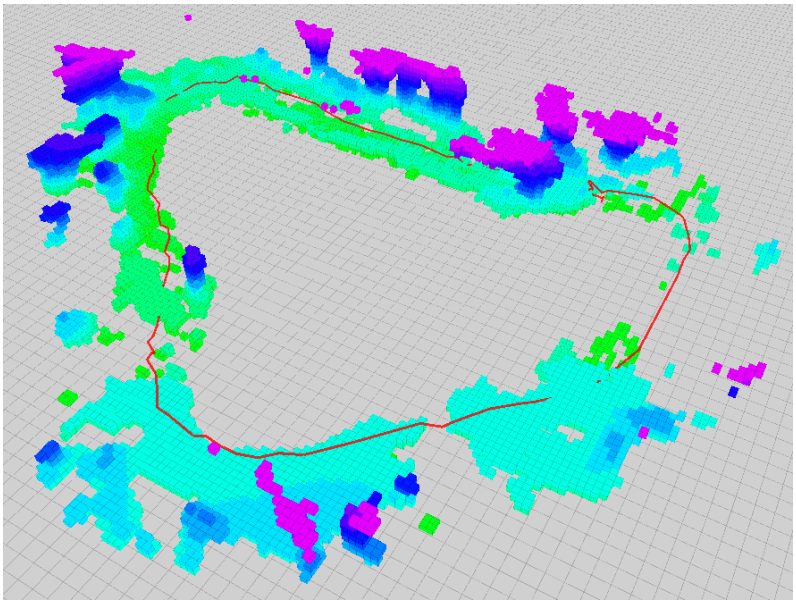


Figure 4.11: 3D map generated in outdoor experiment after loop closure

(a) (b)

Figure 4.12: Images from the onboard camera during autonomous navigation of complex outdoor environments.

4.6 Discussion

In this chapter, we propose a loosely-coupled, combined monocular-stereo approach that is able to accurately track the state of the MAV in real-time with $20\,\text{Hz}$ visual update rate, and $100\,\text{Hz}$ update rate after fusion with IMU. We decouple different components in the system to reduce computation load, and show that such decoupling leads to a fast algorithm that is able to run on mobile processors with limited computation. Our approach is sufficient for feedback control with speed up to $4\,\text{m/s}$.

However, we note that our vision-based approach fails in featureless environments such as rooms with only white walls. However, these kind of environments can be ideal for laser-based approaches as they perfectly satisfy the 2.5D assumption. This motivates and the development of multi-sensor fusion methodologies (Ch. 5) that are able to optimally fuse vision and laser, as well as other measurements, in a consistent manner.

Chapter 5

Multi-Sensor Fusion for Indoor and Outdoor Operations

This chapter describes a methodology for fusing information from multiple sensors, including those presented in previous chapters (Ch. 3 and 4) to improve system robostness in a large variety of environments. The main goal of this work is to develop a modular and extensible approach to integrate noisy measurements from multiple heterogeneous sensors that yield either absolute or relative observations at different and varying time intervals, and to provide smooth and globally consistent state estimates in real time for autonomous flight. The first key contribution, that is central to our work, is a principled approach, building on [84], to fusing relative measurements by augmenting the vehicle state with copies of previous states to create an augmented state vector for which consistent estimates are obtained and maintained using a filtering framework. A second significant contribution is our UKF formulation in which the propagation and update steps circumvent the difficulties that result from the semi-definiteness of the covariance matrix for the augmented state. Finally, we demonstrate results with our experimental platform (Sect. 1.3 and Fig. 1.2(c)) to illustrate the robustness of our framework in large-scale, indoor-outdoor autonomous aerial navigation experiments involving traversals of over 440 meters at average speeds of 1.5 m/s with winds around 10 mph while entering

and exiting two buildings.

We aim to develop a modular framework that allows easy addition and removal of sensors with minimum coding and mathematical derivation. We note that in the popular EKF-based formulation [89, 114], the computation of Jacobians can be problematic for complex systems like MAVs. As such, we employ a loosely coupled, derivative-free Unscented Kalman Filter (UKF) framework [38]. Switching from EKF to UKF poses several challenges, which will be detailed and addressed in Sect. 5.2.1.

5.1 Multi-Sensor System Model

We define vectors in the world and body frames as $(\cdot)^w$ and $(\cdot)^b$ respectively. For the sake of brevity, we assume that all onboard sensors are calibrated and are attached to the body frame. The *main* states of the MAV is defined as:

$$\mathbf{x}_t = \left[\mathbf{p}_t^w,\, \mathbf{\Phi}_t^w,\, \dot{\mathbf{p}}_t^b,\, {}^a\mathbf{b}_t^b,\, {}^\omega\mathbf{b}_t^b,\, {}^z\mathbf{b}_t^w\right]^{\mathrm{T}}$$

where $\mathbf{p}_t^w = [x_t^w,\, y_t^w,\, z_t^w]^{\mathrm{T}}$ is the 3D position in the world frame at time t. Note that $\mathbf{p}_t^w$ can also be interpreted as the position of the body frame at time t with respect to the world frame $(\cdot)^w$, other parameters also follow similar interpretation. $\mathbf{\Phi}_t^w = [\psi_t^w,\, \theta_t^w,\, \phi_t^w]^{\mathrm{T}}$ is the yaw, pitch, and roll Euler angles that represent the 3-D orientation of the body in the world frame [1], from which a matrix $\mathbf{R}_t^w$ that represent the rotation of a vector from the body frame at time t to the world frame can be obtained. $\dot{\mathbf{p}}_t^b$ is the 3D velocity in the body frame. ${}^a\mathbf{b}_t^b$ and ${}^\omega\mathbf{b}_t^b$ are the bias of the accelerometer and gyroscope, both expressed in the body frame. ${}^z\mathbf{b}_t^w$ models the bias of the pressure altimeter in the world frame.

We consider an IMU-based state propagation model:

$$\begin{aligned} \mathbf{u}_t &= \left[\mathbf{a}_t^b,\, \omega_t^b\right]^{\mathrm{T}} \\ \mathbf{v}_t &= \left[{}^a\mathbf{v}_t,\, {}^\omega\mathbf{v}_t,\, {}^{\mathbf{b}}\mathbf{v}_t\right]^{\mathrm{T}} \\ \mathbf{x}_{t+1} &= f(\mathbf{x}_t,\, \mathbf{u}_t,\, \mathbf{v}_t) \end{aligned} \tag{5.1}$$

[1] It is straightforward to formulate the filter with quaternion-based rotation representation [62, 89], We present the direct formulation for the brevity of presentation in Sect. 5.2.

where $\mathbf{u}_t$ is the measurement of linear accelerations and angular velocities from the IMU in the body frame. $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{D}_t) \in \mathbb{R}^{13}$ is the process noise. ${}^{a}\mathbf{v}_t$ and ${}^{\omega}\mathbf{v}_t$ represent additive noise associated with the gyroscope and the accelerometer. ${}^{\mathbf{b}}\mathbf{v}_t$ model the Gaussian random walk of the gyroscope, accelerometer and altimeter bias. The function $f(\cdot)$ is a discretized version of the continuous time dynamical equation [46].

Exteroceptive sensors are usually used to correct the errors in the state propagation. Following [84], we consider measurements as either being *absolute* or *relative*, depending on the nature of the underlying sensor. We allow an arbitrary number of either absolute or relative measurement models.

5.1.1 Absolute Measurements

All absolute measurements can be modeled in the form:

$$\mathbf{z}_{t+m} = h_a(\mathbf{x}_{t+m}, \mathbf{n}_{t+m}) \tag{5.2}$$

where $\mathbf{n}_{t+m} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t) \in \mathbb{R}^p$ is the measurement noise that can be either additive or not. $h_a(\cdot)$ is in general a nonlinear function. An absolute measurement connects the *current* state with the sensor output. Examples are shown in in Sect. 5.4.1.

5.1.2 Relative Measurements

A relative measurement connects the *current* and the *past* states with the sensor output, which can be written as:

$$\mathbf{z}_{t+m} = h_r(\mathbf{x}_{t+m}, \mathbf{x}_t, \mathbf{n}_{t+m}) \tag{5.3}$$

The formulation accurately models the nature of odometry-like algorithms (Sect. 5.4.2 and Sect. 5.4.3) as odometry measures the incremental changes between two time instants of the state. We also note that, in order to avoid temporal drifting, most state-of-the-art laser/visual odometry algorithms are *keyframe* based. As such, we allow multiple future measurement ($m \in \mathcal{M}$, $|\mathcal{M}| > 1$) that corresponds to the same past state $\mathbf{x}_t$.

5.2 UKF-based Multi-Sensor Fusion

We wish to design a modular sensor-fusion filter that is easily extensible even for inexperienced users. This means that amount of coding and mathematical deviation for the addition/removal of sensors should be minimal. One disadvantage of the popular EKF-based filtering framework is the requirement of computing the Jacobian matrices, which is proven to be difficult and time consuming for a complex MAV system. As such, we employ the derivative-free UKF-based approach [38]. The key of UKF is the approximation of the propagation of Gaussian random vectors through nonlinear functions via the propagation of sigma points. Let $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \mathbf{P^{xx}}) \in \mathbb{R}^n$ and consider the nonlinear function:

$$\mathbf{y} = g(\mathbf{x}), \tag{5.4}$$

and let:

$$\begin{aligned} \mathcal{X} &= \left[\hat{\mathbf{x}},\ \hat{\mathbf{x}} \pm \left(\sqrt{(n+\lambda)\mathbf{P^{xx}}}\right)_i\right] \quad \text{for } i = 1, ..., n \\ \mathcal{Y}_i &= g(\mathcal{X}_i), \end{aligned} \tag{5.5}$$

where $g(\cdot)$ is a nonlinear function, λ is a UKF parameter. $\left(\sqrt{(n+\lambda)\mathbf{P^{xx}}}\right)_i$ is the i^{th} column of the square root covariance matrix, which is usually computed via Cholesky decomposition. And $\mathcal{X}$ are called the sigma points. The mean, covariance of the random vector $\mathbf{y}$, and the cross-covariance between $\mathbf{x}$ and $\mathbf{y}$, can be approximated as:

$$\begin{aligned} \hat{\mathbf{y}} &= \sum_{i=0}^{2n} w_i^m \mathcal{Y}_i \\ \mathbf{P^{yy}} &= \sum_{i=0}^{2n} w_i^c (\mathcal{Y}_i - \hat{\mathbf{y}})(\mathcal{Y}_i - \hat{\mathbf{y}})^{\mathrm{T}} \\ \mathbf{P^{yx}} &= \sum_{i=0}^{2n} w_i^c (\mathcal{Y}_i - \hat{\mathbf{y}})(\mathcal{X}_i - \hat{\mathbf{x}})^{\mathrm{T}} \end{aligned} \tag{5.6}$$

where w_i^m and w_i^c are weights for the sigma points. This unscented transform can be used to keep track of the covariance in both the state propagation and measurement update, thus avoiding the need of a Jacobian-based covariance approximation.

5.2.1 State Augmentation for Multiple Relative Measurements

Since a relative measurement depends both the *current* and *past* states, it is a violation of the fundamental assumption in the Kalman filter that the measurement should *only* depend on the *current* state. One way to deal with this is through state augmentation [84], where a copy of the past state is maintained in the filter. Here we present an extension of [84] to handle arbitrary number of relative measurement models with the possiblity that multiple measurements correspond to the same augmented state. Our generic filtering framework allows convenient setup, and facilitates addition and removal of absolute and relative measurement models.

Note that a measurement may not affect all components in the state $\mathbf{x}$. For example, a visual odometry only affects the 6-DOF pose, not the velocity or the bias terms. We define the i^{th} augmented state as $\mathbf{x}_i \in \mathbb{R}^{n_i}$, $n_i \leq n$. $\mathbf{x}_i$ is an arbitrary subset of $\mathbf{x}$. We define a binary selection matrix $\mathbf{B}_i$ of size $n_i \times n$, such that $\mathbf{x}_i = \mathbf{B}_i\mathbf{x}$. Consider a time instant, there are I augmented states in the filter, along with the covariance:

$$\begin{aligned} \check{\mathbf{x}} &= \left[\hat{\mathbf{x}},\, \hat{\mathbf{x}}_1,\, ...,\, \hat{\mathbf{x}}_I\right]^{\mathrm{T}} \\ \check{\mathbf{P}} &= \begin{bmatrix} \mathbf{P}^{\mathbf{x}\mathbf{x}} & \mathbf{P}^{\mathbf{x}\mathbf{x}_1} & \dots & \mathbf{P}^{\mathbf{x}\mathbf{x}_I} \\ \mathbf{P}^{\mathbf{x}_1\mathbf{x}} & \mathbf{P}^{\mathbf{x}_1\mathbf{x}_1} & \dots & \mathbf{P}^{\mathbf{x}_1\mathbf{x}_I} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^{\mathbf{x}_I\mathbf{x}} & \mathbf{P}^{\mathbf{x}_I\mathbf{x}_1} & \dots & \mathbf{P}^{\mathbf{x}_I\mathbf{x}_I} \end{bmatrix}. \end{aligned} \tag{5.7}$$

The addition of a new augmented state $\mathbf{x}_{I+1}$ can be done by:

$$\check{\mathbf{x}}^{+} = \mathbf{M}^{+}\check{\mathbf{x}}, \quad \mathbf{M}^{+} = \begin{bmatrix} \mathbf{I}_{n+\sum_I n_i} \\ \mathbf{B}_{I+1} \end{bmatrix} \tag{5.8}$$

Similarly, the removal of an augmented state $\mathbf{x}_j$ is given as:

$$\check{\mathbf{x}}^{-} = \mathbf{M}^{-}\check{\mathbf{x}}, \quad \mathbf{M}^{-} = \begin{bmatrix} \mathbf{I}_a & \mathbf{0}_{a\times n_j} & \mathbf{0}_{a\times b} \\ \mathbf{0}_{b\times n} & \mathbf{0}_{b\times n_j} & \mathbf{I}_b \end{bmatrix},$$

where $a = n + \sum_{i=1}^{j-1} n_i$ and $b = \sum_{i=j+1}^{I} n_i$. The updated augmented state covariance is

given as:

$$\check{\mathbf{P}}^{\pm} = \mathbf{M}^{\pm}\check{\mathbf{P}}\mathbf{M}^{\pm\mathrm{T}}.$$

The change of keyframes in an odometry-like measurement model is simply the removal of an augmented state $\mathbf{x}_i$ followed by the addition of another augmented state with the same $\mathbf{B}_i$. Since we allow multiple relative measurements that correspond to the same augmented state, contrast to [84], augmented states are *not* deleted after measurement updates (Sect. 5.2.4).

This state augmentation formulation works well in an EKF setting, however, it poses issues when we try to apply it to the UKF. Since the addition of a new augmented state (5.8) is essentially a copy of the *main* state. The resulting covariance matrix $\check{\mathbf{P}}^{+}$ will not be positive definite, and the Cholesky decomposition (5.5) for state propagation will fail (non-unique). We now wish to have something that is similar to the Jacobian matrices for EKF, but *without* explicitly computing the Jacobians.

5.2.2 Statistical Linearization for UKF

In [52], the authors present a new interpretation of the UKF as a Linear Regression Kalman Filter (LRKF). In LRKF, we seek to find the optimal linear approximation $\mathbf{y} = \mathbf{A}\mathbf{x}+\mathbf{b}+\mathbf{e}$ of the nonlinear function (5.4) given a weighted discrete (or sigma points (5.6)) representation of the distribution $\mathcal{N}(\hat{\mathbf{x}}, \mathbf{P}^{\mathbf{xx}})$. The objective is to find the regression matrix $\mathbf{A}$ and vector $\mathbf{b}$ that minimize the linearizion error $\mathbf{e}$:

$$\min_{\mathbf{A},\mathbf{b}} \sum_{i=0}^{2n} w_i(\mathcal{Y}_i - \mathbf{A}\mathcal{X}_i - \mathbf{b})(\mathcal{Y}_i - \mathbf{A}\mathcal{X}_i - \mathbf{b})^{\mathrm{T}}.$$

As shown in [52], the optimal linear regression is given by:

$$\mathbf{A} = \mathbf{P}^{\mathbf{yx}}\mathbf{P}^{\mathbf{xx}^{-1}}, \quad \mathbf{b} = \hat{\mathbf{y}} - \mathbf{A}\hat{\mathbf{x}} \tag{5.9}$$

The linear regression matrix $\mathbf{A}$ in (5.9) serves as the linear approximation of the nonlinear fuction (5.4). This statistical linearization matrix can be used similar to the Jacobian in the EKF formulation. As such, the propagation and update steps in UKF can be performed in a similar fashion as EKF.

5.2.3 State Propagation

Observing the fact that during state propagation only the *main* state changes, we start off by partitioning the augmented state and the covariance (5.7) into:

$$\check{\mathbf{x}}_{t|t} = \begin{bmatrix} \hat{\mathbf{x}}_{t|t} \\ \hat{\mathbf{x}}_{\mathcal{I}_{t|t}} \end{bmatrix}, \quad \check{\mathbf{P}}_{t|t} = \begin{bmatrix} \mathbf{P}^{\mathbf{xx}}_{t|t} & \mathbf{P}^{\mathbf{x}\mathbf{x}_\mathcal{I}}_{t|t} \\ \mathbf{P}^{\mathbf{x}_\mathcal{I}\mathbf{x}}_{t|t} & \mathbf{P}^{\mathbf{x}_\mathcal{I}\mathbf{x}_\mathcal{I}}_{t|t} \end{bmatrix}.$$

The linear approximation of the nonlinear state propagation (5.1), applied on the augmented state (5.7), is:

$$\begin{aligned} \check{\mathbf{x}}_{t+1|t} &= f(\check{\mathbf{x}}_{t|t},\, \mathbf{u}_t,\, \mathbf{v}_t) \\ &= \begin{bmatrix} \mathbf{F}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{|\mathcal{I}|} \end{bmatrix} \check{\mathbf{x}}_{t|t} + \begin{bmatrix} \mathbf{J}_t & \mathbf{G}_t \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}_t \\ \mathbf{v}_t \end{bmatrix} + \mathbf{b}_t + \mathbf{e}_t, \end{aligned} \tag{5.10}$$

from which we can see that the propagation of the full augmented state is actually unnecessary since the only nontrivial regression matrix corresponds to the main state. We can propagate only the main state $\mathbf{x}$ via sigma points generated from $\mathbf{P}^{\mathbf{xx}}_{t|t}$ and use the UKF linearization matrix $\mathbf{F}_t$ to update the cross-covariance $\mathbf{P}^{\mathbf{x}\mathbf{x}_\mathcal{I}}_{t|t}$. Since the covariance matrix of the main state $\mathbf{P}^{\mathbf{xx}}_{t|t}$ is always positive definite, we avoid the Cholesky decomposition failure problem.

Since the process noise is not additive, we augment the main state with the process noise and generate sigma points from:

$$\bar{\mathbf{x}}_{t|t} = \begin{bmatrix} \hat{\mathbf{x}}_{t|t} \\ \mathbf{0} \end{bmatrix}, \quad \bar{\mathbf{P}}_{t|t} = \begin{bmatrix} \mathbf{P}^{\mathbf{xx}}_{t|t} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_t \end{bmatrix}. \tag{5.11}$$

The state is then propagated forward by substituting (5.11) into (5.1), (5.5) and (5.6). We obtain $\hat{\mathbf{x}}_{t+1|t}$, the estimated value of $\mathbf{x}$ at time $t+1$ given the measurements up to t, as well as $\mathbf{P}^{\mathbf{xx}}_{t+1|t}$ and $\mathbf{P}^{\mathbf{x}\bar{\mathbf{x}}}_{t+1|t}$. Following (5.9), we know that:

$$\mathbf{P}^{\mathbf{x}\bar{\mathbf{x}}}_{t+1|t} \bar{\mathbf{P}}^{-1}_{t|t} = [\mathbf{F}_t,\, \mathbf{G}_t]\,.$$

The propagated augmented state and its covariance is updated according to (5.10):

$$\check{\mathbf{x}}_{t+1|t} = \begin{bmatrix} \hat{\mathbf{x}}_{t+1|t} \\ \hat{\mathbf{x}}_{\mathcal{I}_{t|t}} \end{bmatrix}, \quad \check{\mathbf{P}}_{t+1|t} = \begin{bmatrix} \mathbf{P}^{\mathbf{xx}}_{t+1|t} & \mathbf{F}_t \mathbf{P}^{\mathbf{x}\mathbf{x}_\mathcal{I}}_{t|t} \\ \mathbf{P}^{\mathbf{x}_\mathcal{I}\mathbf{x}}_{t|t} \mathbf{F}_t{}^{\mathsf{T}} & \mathbf{P}^{\mathbf{x}_\mathcal{I}\mathbf{x}_\mathcal{I}}_{t|t} \end{bmatrix}. \tag{5.12}$$

5.2.4 Measurement Update

Let there be m state propagations between two measurements, and we maintain $\check{\mathbf{x}}_{t+m|t}$ and $\check{\mathbf{P}}_{t+m|t}$ as the newest measurement arrives. Consider a relative measurement (5.3) that depends on the j^{th} augmented state, the measurement prediction and its linear regression approximation can be written as:

$$\begin{aligned}\hat{\mathbf{z}}_{t+m|t} &= h_r(\hat{\mathbf{x}}_{t+m|t},\, {\mathbf{B}_j}^{\mathrm{T}}\hat{\mathbf{x}}_{j_{t+m|t}},\, \mathbf{n}_{t+m}) \\ &= \mathbf{H}_{t+m|t}\check{\mathbf{x}}_{t+m|t} + \mathbf{L}_{t+m}\mathbf{n}_{t+m} + \mathbf{b}_{t+m} + \mathbf{e}_{t+m} \\ \mathbf{H}_{t+m|t} &= \left[\mathbf{H}^{\mathbf{x}}_{t+m|t},\, \mathbf{0},\, \mathbf{H}^{\mathbf{x}_j}_{t+m|t},\, \mathbf{0}\right].\end{aligned}$$

Again, since only the main state and one augmented state are involved in each measurement update, we can construct another augmented state together with the possibly non-additive measurement noise:

$$\grave{\mathbf{x}}_{t+m|t} = \begin{bmatrix}\hat{\mathbf{x}}_{t+m|t} \\ \hat{\mathbf{x}}_{j_{t+m|t}} \\ \mathbf{0}\end{bmatrix}, \grave{\mathbf{P}}_{t+m|t} = \begin{bmatrix}\mathbf{P}^{\mathbf{xx}}_{t+m|t} & \mathbf{P}^{\mathbf{x}\mathbf{x}_j}_{t+m|t} & \mathbf{0} \\ \mathbf{P}^{\mathbf{x}_j\mathbf{x}}_{t+m|t} & \mathbf{P}^{\mathbf{x}_j\mathbf{x}_j}_{t+m|t} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}_{t+m}\end{bmatrix}.$$

After the state propagation (5.12), $\grave{\mathbf{P}}_{t+m|t}$ is guaranteed to be positive definite, thus it is safe to perform sigma point propagation as in (5.5) and (5.6). We obtain $\hat{\mathbf{z}}_{t+m|t}$, $\mathbf{P}^{\mathbf{zz}}_{t+m|t}$, $\mathbf{P}^{\mathbf{z}\grave{\mathbf{x}}}_{t+m|t}$, and:

$$\mathbf{P}^{\mathbf{z}\grave{\mathbf{x}}}_{t+m|t}\grave{\mathbf{P}}^{-1}_{t+m|t} = \left[\mathbf{H}^{\mathbf{x}}_{t+m|t},\, \mathbf{H}^{\mathbf{x}_j}_{t+m|t},\, \mathbf{L}_{t+m}\right].$$

We can apply the measurement update similar to an EKF:

$$\begin{aligned}\check{\mathbf{K}}_{t+m} &= \check{\mathbf{P}}_{t+m|t}{\mathbf{H}_{t+m|t}}^{\mathrm{T}}{\mathbf{P}^{\mathbf{zz}}_{t+m|t}}^{-1} \\ \check{\mathbf{x}}_{t+m|t+m} &= \check{\mathbf{x}}_{t+m|t} + \check{\mathbf{K}}_{t+m}\left(\mathbf{z}_{t+m} - \hat{\mathbf{z}}_{t+m|t}\right) \\ \check{\mathbf{P}}_{t+m|t+m} &= \check{\mathbf{P}}_{t+m|t} - \check{\mathbf{K}}_{t+m}\mathbf{H}_{t+m|t}\check{\mathbf{P}}_{t+m|t},\end{aligned}$$

where $\mathbf{z}_{t+m}$ is the actual sensor measurement. Both the main and augmented states will be corrected during measurement update. We note that entries in $\mathbf{H}_{t+m|t}$ that correspond to inactive augmented states are zero. This can be utilized to speed up the matrix multiplication.

The fusion of absolute measurements can simply be done by setting $\hat{\mathbf{x}}_{j_{t+m|t}} = \emptyset$ and applying the corresponding absolute measurement model (5.2).

As shown in Fig. 5.9, fusion of multiple relative measurements results in slow growing, but *unbounded* covariance in the global position. This is consistent with results in [46] that these global quantities are unobservable.

5.2.5 Delayed and Out-of-Order Measurement Update

When fusing multiple measurements, it is possible that the measurements arrive out-of-order to the filter, that is, a measurement that corresponds to an *earlier* state arrives *after* the measurement that corresponds to a *later* state. This violates the Markov assumption of the Kalman filter. Also, due to the sensor processing delay, measurements may lag behind the state propagation.

We address these two issues by storing measurements in a priority queue, where the top of the queue corresponds to the oldest measurement. A pre-defined a maximum allowable sensor delay t_d of 100 ms was set for our MAV platform. Newly arrived measurements that correspond to a state older than t_d from the current state (generated by state propagation) are directly discarded. After each state propagation, we check the queue and process all measurements in the queue that are older than t_d. The priority queue essentially serves as a measurement reordering mechanism (Fig. 5.1) for all measurements that are not older than t_d from the current state. In the filter, we always utilize the most recent IMU measurement to propagate the state forward. We, however, only propagate the covariance on demand. As illustrated in Fig. 5.1, the covariance is only propagated from the time of the last measurement to the current measurement.

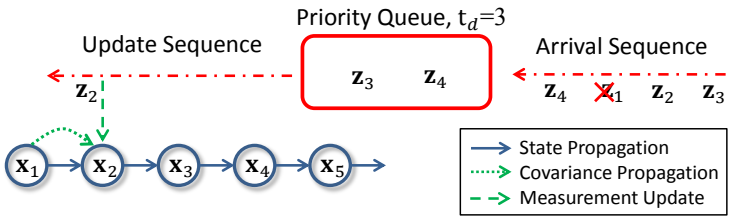


Figure 5.1: Delayed and out-of-order measurement update with priority queue. While $\mathbf{z}_4$ arrives before $\mathbf{z}_2$, $\mathbf{z}_2$ is first applied to the filter. $\mathbf{z}_4$ is temporary stored in the queue. $\mathbf{z}_1$ is discarded since it is older than t_d from the current state. The covariance is only propagated up to time where the most recent measurement is applied to the filter. The state is propagated till the most recent IMU input.

5.3 Handling Global Pose Measurements

As the vehicle moves through the environment, global pose measurements from GPS and magnetometer may be available. It is straightforward to fuse the GPS as a global pose measurement and generate the optimal state estimate. However, this may not be the best for real-world applications. A vehicle that operates in a GPS-denied environment may suffer from accumulated drift. When the vehicle gains GPS signal, as illustrated in Fig. 5.2(a), there maybe large discrepancies between the GPS measurement and the estimated state ($\mathbf{z}_5 - \mathbf{s}_5$). Directly applying GPS as global measurements will result in undesirable behaviors in both estimation (large linearizion error) and control (sudden pose change).

This is not a new problem and it has been studied for ground vehicles [75] under the term of local frame-based navigation. However, [75] assumes that a reasonably accurate local estimate of the vehicle is always available (e.g. wheel odometry). This is not the case for MAVs since the state estimates with only the onboard IMUs drifts away vastly within a few seconds. The major difference between dead reckoning with IMU and wheel odometry is that the former drifts temporally, while the latter only drifts spatially. However, we have relative exteroceptive sensors that are able to produce temporally drift-free estimates. As such, we only need to deal with the case that all relative exteroceptive sensors have failed. Therefore, our goal is to properly transform the global GPS mea-

surement into the local frame to bridge the gap between relative sensor failures.

Consider a pose-only graph SLAM formulation with $\mathbf{s}_k = [x_k^w,\, y_k^w,\, \psi_k^w]^{\mathrm{T}} \in \Theta$ being 2D poses. The SLAM module may run at a much lower rate than the UKF-based estimator. We optimize the pose graph given incremental motion constraints $\mathbf{d}_k$ from the multi-sensor UKF with only relative measurements (laser/visual odometry), spatial loop closure constraints $\mathbf{l}_k$, and absolute pose constraints $\mathbf{z}_k$ from GPS:

$$\min_{\Theta} \left\{ \sum_{k=1}^{M} \| h_i(\mathbf{s}_{k-1}, \mathbf{d}_k) - \mathbf{s}_k \|_{\mathbf{P}_k^{\mathbf{d}}} \right.$$
$$\left. + \sum_{k=1}^{L} \left\| h_l(\mathbf{s}_k, \mathbf{l}_k) - \mathbf{s}_{l(k)} \right\|_{\mathbf{P}_k^{\mathbf{l}}} + \sum_{k=1}^{N} \| \mathbf{z}_k - \mathbf{s}_k \|_{\mathbf{P}_k^{\mathbf{z}}} \right\}.$$

Similar to previous chapters (Ch. 3 and 4), The optimal pose graph configuration can be found with available solvers [16, 48], as shown in Fig. 5.2(b). The pose graph is disconnected if there are no relative exteroceptive measurements between two nodes. Let two pose graphs be disconnected between $k-1$ and k.

The pose graph SLAM provides the transformation between the non-optimized $\mathbf{s}_{k-1}$ and the SLAM-optimized $\mathbf{s}_{k-1}^{+}$ state. This transform can be utilized to transform the global GPS measurement to be aligned with $\mathbf{s}_{k-1}$:

$$\Delta_{t-1} = \mathbf{s}_{k-1} \ominus \mathbf{s}_{k-1}^{+}$$
$$\mathbf{z}_{k-1}^{-} = \Delta_{t-1} \oplus \mathbf{z}_{k-1},$$

where $\oplus$ and $\ominus$ are pose compound operations as defined in [104]. The covariance $\mathbf{P}_{t-1}^{\Delta}$ of Δ_{t-1} and subsequently the covariance $\mathbf{P}_{t-1}^{z^-}$ of $\mathbf{z}_{k-1}^{-}$ can be computed following [104]. This formulation minimizes the discrepancies between $\mathbf{z}_{k-1}^{-}$ and $\mathbf{s}_{k-1}$, and thus maintains smoothness in the state estimate. The transformed GPS $\mathbf{z}_{k-1}^{-}$, is still applied as an absolute measurement to the UKF (Fig. 5.3(a)).

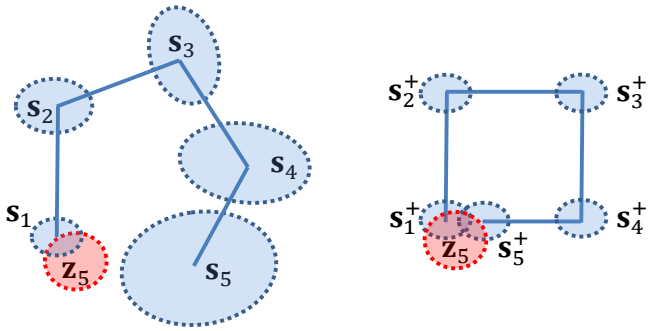


Figure 5.2: In Fig. 5.2(a), GPS signal is regained at $k = 5$, resulting in large discrepancies between the measurement $\mathbf{z}_5$ and the state $\mathbf{s}_5$. Pose graph SLAM produces a globally consistent graph (Fig. 5.2(b)).

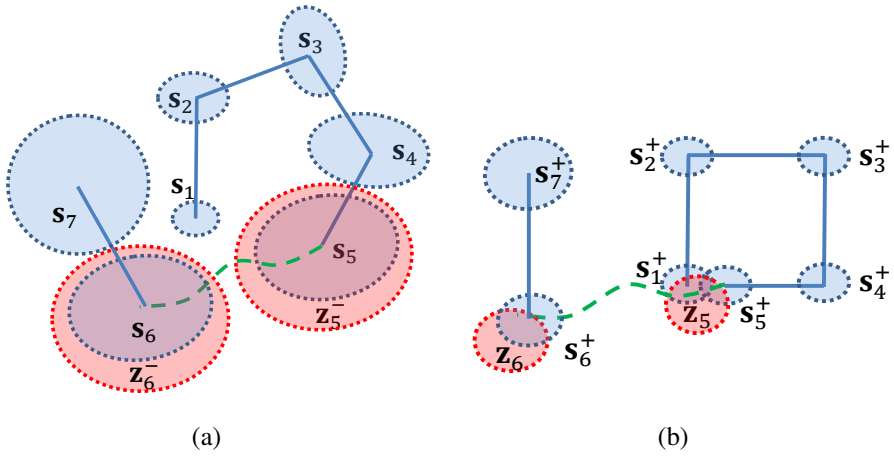


Figure 5.3: Fig. 5.3(a) illustrates the alternative GPS fusion, the discrepancy between transformed GPS measurement $\mathbf{z}_5^-$ and the non-optimized state $\mathbf{s}_5$ is reduced. Fusion of such indirect GPS measurement will lead to smooth state estimate (green dashed line).

5.4 Implementation Details

To verify the proposed sensor fusion algorithm, we develop a quadrotor MAV platform equipped with multiple sensors and sufficient computation power. The platform is discussed in Sect. 1.3 and shown in Fig. 1.2(c). The experimental platform is based on the Pelican quadrotor from Ascending Technologies, GmbH. This platform is natively equipped with an AutoPilot board consisting of an IMU and a user-programmable ARM7 microcontroller. The main computation unit onboard is an Intel NUC with a 1.8 GHz Core i3 processor with $8\,$GB of RAM and a $120\,$GB SSD. The sensor suite includes a u-blox LEA-6T GPS module, a Hokuyo UTM-30LX LiDAR and two mvBlueFOX-MLC200w grayscale HDR cameras with fisheye lenses that capture 752×480 images at $25\,$Hz. We use hardware triggering for frame synchronization. The onboard auto exposure controller is fine tuned to enable fast adaption during rapid light condition changes. A 3-D printed laser housing redirects some of the laser beams for altitude measurement. The total mass of the platform is 1.87kg. The entire algorithm is developed in C++ using ROS as the interfacing robotics middleware.

Fig. 5.4 shows the typical sensor setup of the platform. Note that depending on mission requirement and available payload, sensors can be added and/or removed by simply write down the state space model in scripting language. We now describes the measurement models of some typical sensors.

5.4.1 Absolute Measurements

Some onboard sensors are capable of producing absolute measurements (Sect. 5.1.1), here are their details:

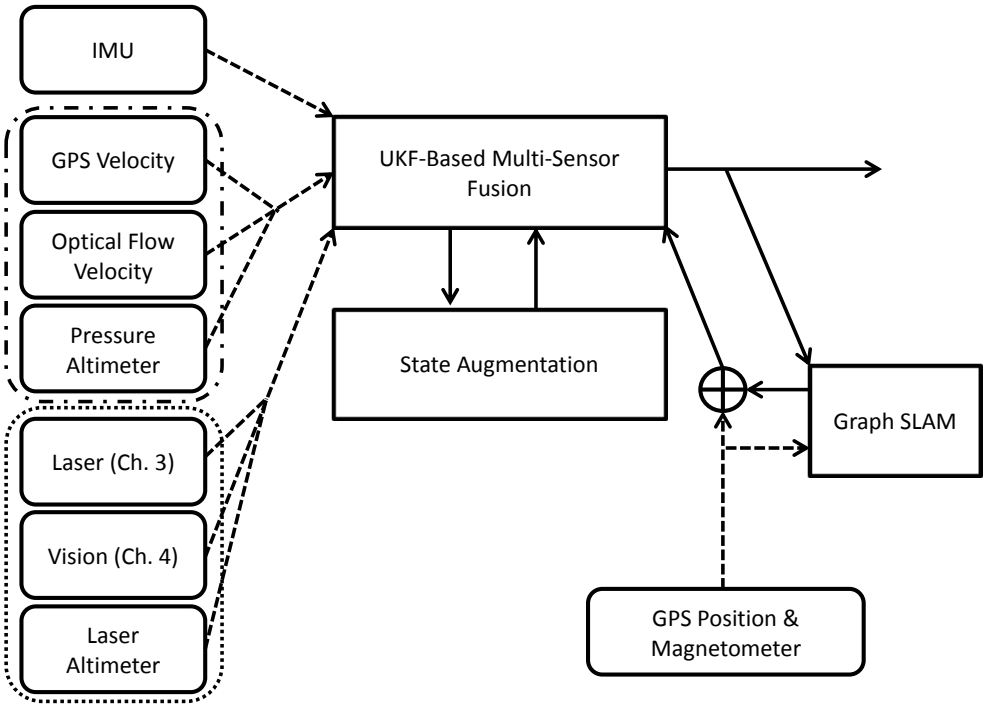


Figure 5.4: Setup of the multi-sensor MAV platform. Absolute sensors are grouped by dashed box, while relative sensors are grouped by dotted box. Laser- and vision-based state estimation that provide relative measurements are discussed in Ch. 3 (Fig. 3.1) and Ch. 4 (Fig. 4.1), respectively. Also note the special handling of GPS position measurement to ensure smoothness of the state estimate.

GPS And Magnetometer

$$\mathbf{z}_t = \begin{bmatrix} \begin{pmatrix} x_t^w \\ y_t^w \end{pmatrix} \\ \mathbf{R}_t^w \begin{pmatrix} \dot{x}_t^b \\ \dot{y}_t^b \end{pmatrix} \\ \psi_t^w \end{bmatrix} + \mathbf{n}_t.$$

Pressure Altimeter

$$\mathbf{z}_t = z_t^w + \mathbf{b}_{z_t}^w + \mathbf{n}_t.$$

Pseudo Gravity Vector

If the MAVs is near hover or moving at approximately constant speed, we may say that the accelerometer output provides a pseudo measurement of the gravity vector. Let $\mathbf{g}^w = [0,\ 0,\ g]^{\mathrm{T}}$, we have:

$$\mathbf{z}_t = \mathbf{R}_t^{w\mathrm{T}} \mathbf{g}^w + \mathbf{b}_{a_t}^b + \mathbf{n}_t.$$

5.4.2 Relative Measurement - Laser Odometry

We utilize the laser-based pose estimator described in Ch. 3. Observing that man-made indoor environments mostly contains vertical walls, we can make a 2.5-D environment assumption. With this assumption, we can make use of the onboard roll and pitch estimates to project the laser scanner onto a common ground plane. As such, 2D scan matching can be utilized to estimate the incremental horizontal motion of the vehicle. We keep a local map to avoid drifting while hovering. While our previous EKF-based sensor fusion (Sect. 3.2 treat laser odometry as absolute measurements, here we do the

measurement update in a principled way by treating them as relative measurements.

$$\mathbf{z}_{t+m} = \ominus_{2d} \begin{bmatrix} x_t^w \\ y_t^w \\ \psi_t^w \end{bmatrix} \oplus_{2d} \begin{bmatrix} x_{t+m}^w \\ y_{t+m}^w \\ \psi_{t+m}^w \end{bmatrix} + \mathbf{n}_{t+m},$$

where $\mathbf{p}_{2d_t} = [x_t^w,\, y_t^w,\, \psi_t^w]^{\mathsf{T}}$, $\oplus_{2d}$ and $\ominus_{2d}$ are the 2-D pose compound operations as defined in [104].

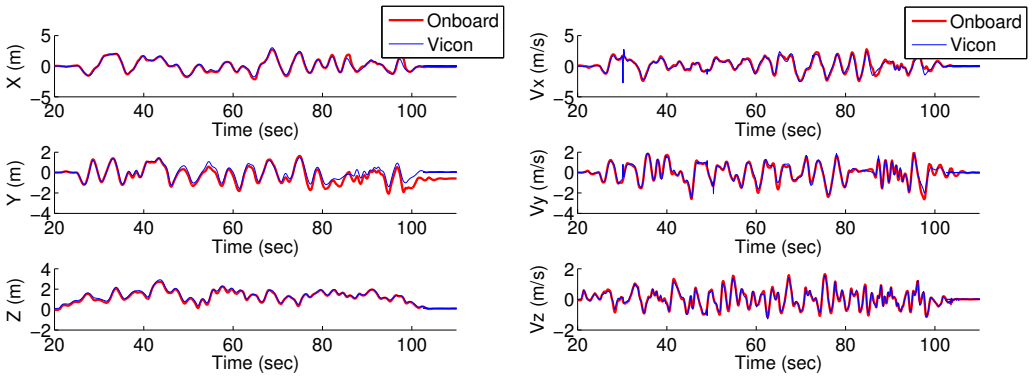
5.4.3 Relative Measurement - Visual Odometry

We test both our vision-based state estimation approach (Ch. 4) and a classic keyframe-based stereo visual odometry algorithm. For latter case, we choose to use light-weight corner features but run the algorithm at a high-rate ($25\,\text{Hz}$). Features are tracked across images via KLT tracker. Given a keyframe with a set of triangulated feature points, we run a robust iterative 2D-3D pose estimation [89] to estimate the 6-DOF motion of the vehicle with respect to the keyframe. New keyframes are inserted depending on the distance traveled and the current number of valid 3D points. Similar to laser odometry, we convert the absolute measurement update approach as in Sect 4.4 into a more principled relative measurement model.

$$\mathbf{z}_{t+m} = \ominus \begin{bmatrix} \mathbf{p}_t^w \\ \mathbf{\Phi}_t^w \end{bmatrix} \oplus \begin{bmatrix} \mathbf{p}_{t+m}^w \\ \mathbf{\Phi}_{t+m}^w \end{bmatrix} + \mathbf{n}_{t+m}$$

5.5 Experimental Results

Multiple experiments are conducted to demonstrate the robustness of our system. We begin with an quantitative evaluation in a lab environment equipped with a motion capture systems. We then test our system in two real-world autonomous flight experiments, including an industrial complex and a tree-lined campus.



(a) Position (b) Velocity

Figure 5.5: The MAV maneuvers aggressively with a maximum speed of $3.5\,\text{m/s}$ (Fig. 5.5(b)). The horizontal position also compares well with the ground truth with slight drift (Fig. 5.5(a)).

5.5.1 Evaluation of Estimator Performance

We would like to push the limits of our onboard estimator. Therefore, we have a professional pilot to aggressively fly the quadrotor with a $3.5\,\text{m/s}$ maximum speed and large attitude of up to 40°. The onboard state estimates are compared the ground truth from the motion capture system. Since there is no GPS measurement indoor, our system relies on a fusion of relative measurements from laser and vision. We do observe occasional laser failure due to large attitude violating the 2.5-D assumption (Sect. 5.4.2). However, the multi-sensor filter still tracks the vehicle state throughout (Fig. 5.5). We do not quantify the absolute pose error since it is unbounded. However, the body frame velocity (Fig. 5.5(b)) compares well with the ground truth with standard deviations of $\{0.1021,\ 0.1185,\ 0.0755\}^{\text{T}}$ (m/s) in x, y, and z, respectively.

5.5.2 Autonomous Flight in Indoor and Outdoor Environments

We tested our system in a challenging industrial complex. The testing site spans a variety of environments, including outdoor open space, densely filled trees, cluttered building

area, and indoor environments (Fig. 5.6). The MAV is autonomously controlled using the onboard state estimates. However, a human operator always has the option of sending high level waypoints or velocity commands to the vehicle. The total flight time is approximately 8 minutes, and the vehicle travels 445 meters with an average speed of 1.5 m/s. As shown in the map-aligned trajectory (Fig. 5.7), during the experiment, frequent sensor failures occurred (Fig. 5.8), indicating the necessity of multi-sensor fusion. Fig. 5.9 shows the evolution of covariance as the vehicle flies through a GPS shadowing area. The global x, y and yaw error is bounded by GPS measurement, without which the error will grow unbounded. This matches the observability analysis results. It should be noted that the error on body frame velocity does not grow, regardless of the availability of GPS. The spike in velocity covariance in Fig. 5.9 is due to the camera facing direct sunlight. Fig. 5.10 shows a more throughout plot of the impact of sensor availability on the state estimation uncertainty.

5.5.3 Autonomous Flight in Tree-Lined Campus

We also conduct experiments in a tree-lined campus environment, as shown in Fig. 5.12 and 5.11. Autonomous flight in this environment is challenging due to nontrivial light condition changes as the vehicle moves in and out of tree shadows. The risk of GPS failure is also very high due to the trees above the vehicle. Laser-based odometry only works when close to buildings. The total trajectory length is 281 meters.

Figure 5.6: Images from the onboard camera (left column) and corresponding images from an external camera (right column). Note the vast variety of environments, including open space, trees, complex building structures, and indoor environments. We highlight the position of the MAV with a red circle.

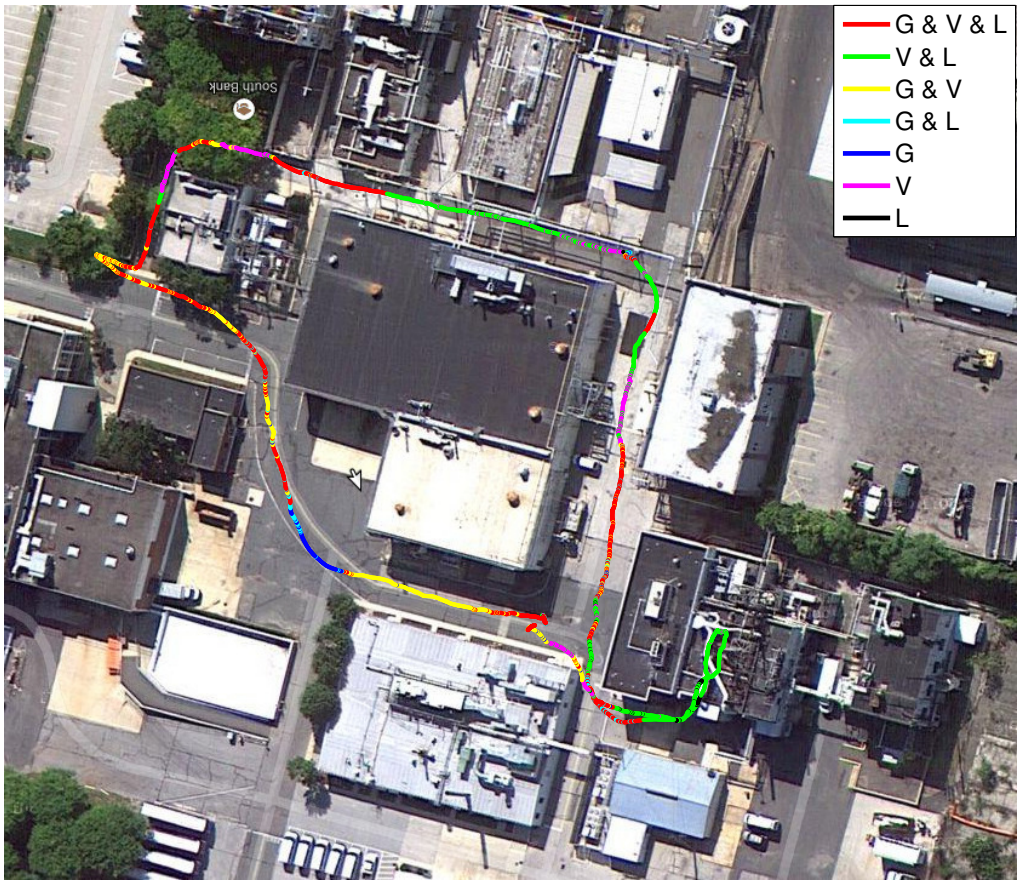


Figure 5.7: Vehicle trajectory aligned with satellite imagery. Different colors indicate different combinations of sensing modalities. G=GPS, V=Vision, and L=Laser.

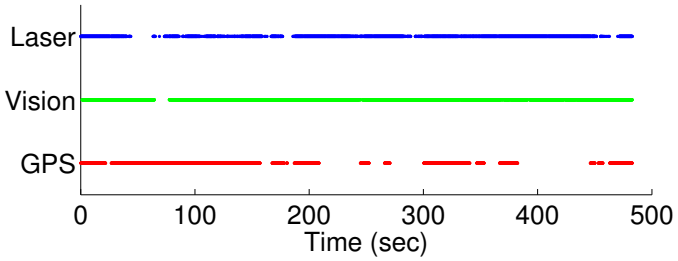


Figure 5.8: Sensor availability over time. Note that failures occurred to all sensors. This shows that multi-sensor fusion is a must for this kind of indoor-outdoor missions.

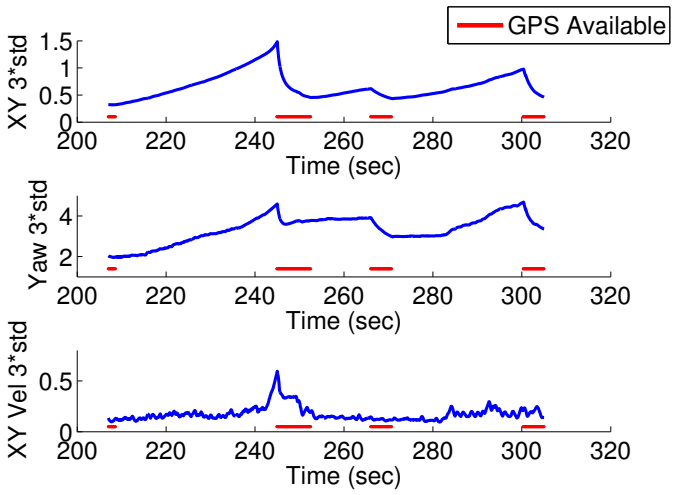


Figure 5.9: Covariance changes as the vehicle flies through a dense building area (between 200s - 300s, top of Fig. 5.7, green line). The GPS comes in and out due to building shadowing. The covariance of x, y, and yaw increases as GPS fails and decreases as GPS resumes. Note that the body frame velocity are observable regardless of GPS measurements, and thus its covariance remains small. The spike in the velocity covariance is due to the vehicle directly facing the sun. The X-Y covariance is calculated from the Frobenius norm of the covariance submatrix.

5.6 Benefits and Limitations

We note that the filtering framework is not the only way to fuse heterogeneous sensor measurements. As discussed in Sect. 2.4, it is straightforward perform multi-sensor fusion in a graph-based SLAM framework as each measurement simply adds another edge or factor node to the graph [11, 34, 88]. However, as results of sensor fusion are used directly for feedback control of the MAV, the large latency and computation complexity of graph-based approaches making them inappropriate for our application.

On the other hand, our modular filtering-based multi-sensor fusion framework is a loosely-coupled approach. We benefit from the the plug-and-ply feature of the methodology, which enables easy reconfigurations of sensors. The loosely-coupled approach also makes the sensor fusion framework computationally lightweight, which is essential for MAV feedback stabilization purposes.

However, our method relies on external modules to process raw sensor readings and

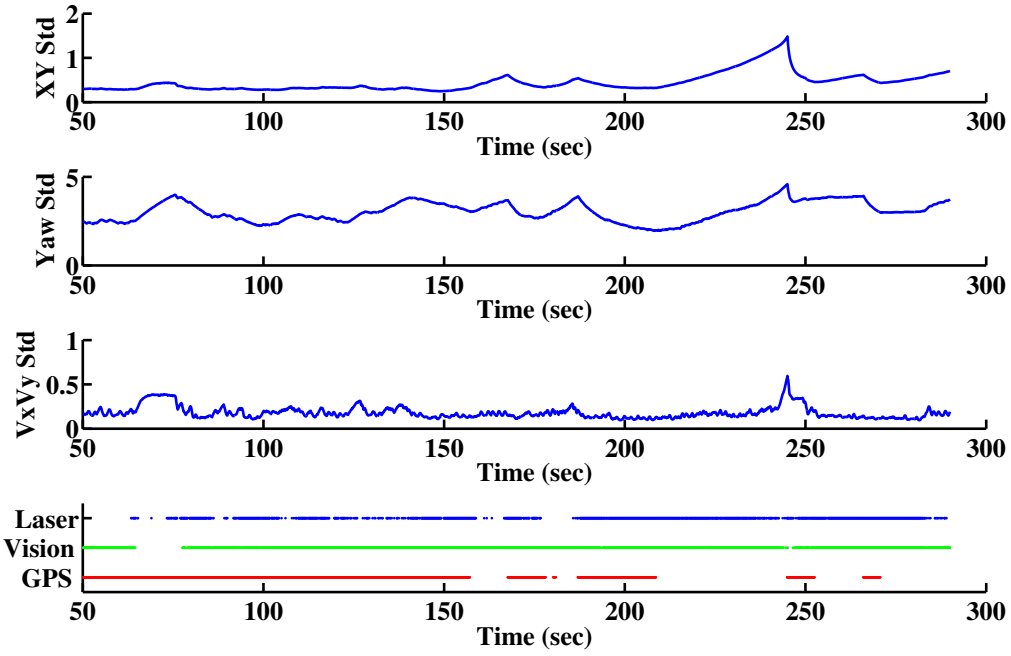


Figure 5.10: Impact of sensor availability on state estimation uncertainty. Around 65 sec, only GPS is available, resulting in high but bounded uncertainty in position, rotation, and velocity. Around 160 sec, the unavailability of both GPS and laser causes significant worse estimation performance. Between $200 - 250$ sec, the lack of GPS measurements causes unbounded increase in position and yaw.

Figure 5.11: Onboard (left) and external (right) camera images as the MAV autonomously flies through a tree-lined campus environment. Note the nontrivial light condition.

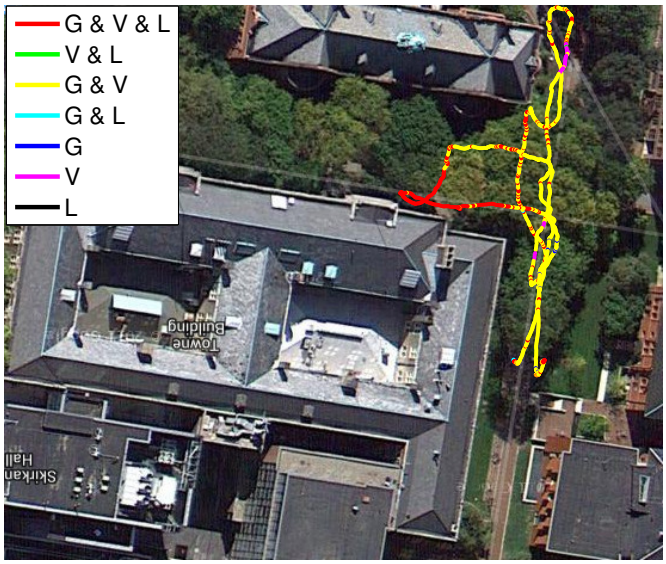


Figure 5.12: Vehicle trajectory overlaid on a satellite map. The vehicle operates in a tree-lined campus environment, where there is high risk of GPS failure during operation.

convert them into state estimates with associated covariance matrices. We also require external modules to report failures of sensors. The quality and consistency of those external modules are crucial to the overall fusion performance. Our sensor fusion framework ignores the cross-coupling between measurements, which may lead to suboptimal fusion results comparing to tightly coupled approaches [34]. Nonetheless, we stress that in applications such as MAV feedback control, processing speed is more important than overall accuracy. For these applications, our approach still provides a viable and easy-to-use solution to the complex sensor fusion problem.

5.7 Discussion

In this chapter, we present a modular and extensible approach to integrate noisy measurements from multiple heterogeneous sensors that yield either absolute or relative observations at different and varying time intervals. Our approach generates high rate state estimates in real-time for autonomous flight. The proposed approach runs onboard our

new 1.9 kg MAV platform equipped with multiple heterogeneous sensors. We demonstrate the robustness of our framework in large-scale, indoor and outdoor autonomous flight experiments that involves traversal through a industrial complex and a tree-lined campus.

However, we note in all experiments, we have to have the MAV start from stationary in order to initialize the filter. Also, our filter will still breakdown (or diverge) if all sensors fail, and once the filter breakdown, it may not be able to recover even if some or all sensors resumes operation. This motivates us to investigate into online initialization and failure recovery methodologies in the next Chapter (Ch. 6).

Chapter 6

Initialization and Failure Recovery for Monocular Visual-Inertial Systems

In this chapter, we focus on the initialization and failure recovery of estimators that are used for MAVs. In particular, we consider a monocular visual-inertial system (VINS) that consists of only an IMU and a camera. We are interested in this case because of two reasons. First, as we would like operate agile autonomous MAVs in confined environments, the platforms we use typically have very tight constrains on size, weight, and power (SWaP). Up to some point, a monocular VINS is the only viable setup due to its ultra light weight and small footprint. Second, this is a intellectually challenging problem in which initialization is of great importance. For platforms with comprehensive set of sensors such as the one we used in Ch. 5, almost all states are directly observable by onboard sensors, and initialization can simply be done with the first sensor reading. However, for monocular VINS, most of the critical navigation states such as initial velocity, attitude, as well as the metric scale are not directly observable, Providing a reasonable initial value of these states can be challenging without additional sensors or assumptions about the environments.

While there does exists closed-form initialization approaches for monocular VINS [17, 45, 56, 66, 67], as discussed in Sect. 2.6, most of them require either global orientation

to be known or are inoperable with noisy sensors due to the lack of good probabilistic models characterizing sensor behavior.

Another issue of monocular VINS is the scale ambiguity due to degenerate motion. It is well known that in order to have the scale be observable, accelerations in at least two axes are required [37, 41, 66]. However, for a MAV, degenerate motions such as hovering or constant velocity motions are unavoidable and have to be handled properly.

Based on the idea of reducing nonlinearity via frame transform in [61], this chapter addresses the development of a monocular VINS estimator that is capable of on-the-fly initialization and failure recovery. We first propose a linear sliding window formulation for monocular VINS that is able to estimate necessary navigation states (velocity and attitude) without any prior initial information (Sect. 6.1). We present a nonlinear optimization-based monocular VINS estimation that operates on the tangent space of the rotation group to refine the initial solution (Sect. 6.2). We combine both the linear and nonlinear formulations to form a complete system that is capable of recovery from failures (Sect. 6.4). Inspired by the observability condition analysis in [67], we address the issues of degenerate motion and scale unobservability by proposing a two-way marginalization scheme (Sect. 6.3). Finally, by combining with planning and control methodologies (Ch 7), we experimentally show that the proposed approach enables metric state estimation without initialization. We also show that our system is able to handle degenerate motion cases (Sect. 6.6). A diagram of the proposed monocular VINS approach is shown in Fig. 6.2.

6.1 Linear Sliding Window VINS Estimator

The key idea behind on-the-fly initialization and failure recovery for monocular VINS is the development of a linear sliding window estimator that turns all information from both the IMU and camera, in a fixed time interval, into estimates of initial velocity, gravity vector, and depth of features. We start by defining notations. We consider $(\cdot)^w$ as the

earth's inertial frame, $(\cdot)^b$ as the current IMU body frame, $(\cdot)^k$ as the camera frame while taking the k^{th} image, Note that IMU usually runs at a higher rate than the camera, and that multiple IMU measurements may exist in the interval $[k, k+1]$. We assume that the camera and the IMU is pre-calibrated such that the camera optical axis is aligned with the z-axis of the IMU. $\mathbf{p}_Y^X$, $\mathbf{v}_Y^X$, and $\mathbf{R}_Y^X$ are 3D position , velocity, and rotation of frame Y with respect to frame X. In particular, $\mathbf{p}_t^X$ represents the position of the body frame at time t with respect to frame X. Similar conversion follows for other parameters. $\mathbf{g}^w = [0,\ 0,\ g]^{\mathrm{T}}$ is the gravity vector in the world frame, and $\mathbf{g}^k$ is the earth's gravity vector expressed in the body frame of the k^{th} image.

6.1.1 Formulation

Given two time instants (corresponding to two image frames), the IMU propagation model for position and velocity, expressed in the world frame, can be written as:

$$\begin{aligned}\mathbf{p}_{k+1}^w &= \mathbf{p}_k^w + \mathbf{v}_k^w \Delta t + \iint_{t\in[k,k+1]} \left(\mathbf{R}_t^w \mathbf{a}_t^b - \mathbf{g}^w\right) dt^2 \\ \mathbf{v}_{k+1}^w &= \mathbf{v}_k^w + \int_{t\in[k,k+1]} \left(\mathbf{R}_t^w \mathbf{a}_t^b - \mathbf{g}^w\right) dt\end{aligned} \tag{6.1}$$

where $\mathbf{a}_t^b$ is the linear acceleration in the body frame, Δt is the time difference between k and $k+1$. It can be seen that the rotation between the world frame and the body frame is required in order to propagate the states with IMU measurements. This rotation can only be determined if the initial attitude of the vehicle is known, which is not the case when the vehicle is dynamically launched or recovering from estimator failure. However, as suggested in [61], if the reference frame of the IMU propagation model is attached to the first pose of the system (i.e. the first pose that we are trying to estimate), (6.1) can be rewritten as:

$$\begin{aligned}\mathbf{p}_{k+1}^0 &= \mathbf{p}_k^0 + \mathbf{R}_k^0 \mathbf{v}_k^k \Delta t - \mathbf{R}_k^0 \mathbf{g}^k \Delta t^2/2 + \mathbf{R}_k^0 \alpha_{k+1}^k \\ \mathbf{v}_{k+1}^{k+1} &= \mathbf{R}_k^{k+1} \mathbf{v}_k^k - \mathbf{R}_k^{k+1} \mathbf{g}^k \Delta t + \mathbf{R}_k^{k+1} \beta_{k+1}^k \\ \mathbf{g}^{k+1} &= \mathbf{R}_k^{k+1} \mathbf{g}^k\end{aligned} \tag{6.2}$$

where

$$\boldsymbol{\alpha}_{k+1}^{k} = \iint_{t\in[k,k+1]} \mathbf{R}_t^k \mathbf{a}_t^b dt^2$$
$$\boldsymbol{\beta}_{k+1}^{k} = \int_{t\in[k,k+1]} \mathbf{R}_t^k \mathbf{a}_t^b dt$$

where $\mathbf{R}_k^0$ is the change in rotation since the first pose (or since the 0^{th} image), and $\mathbf{R}_{k+1}^k$ is the incremental rotation between two images. Both can be obtained by combing the integral gyroscope measurements and relative epipolar constraints (Sect. 6.1.2). $\boldsymbol{\alpha}_{k+1}^k$ and $\boldsymbol{\beta}_{k+1}^k$ can be obtained solely with IMU measurements within $[k, k+1]$. We can see that the update equations for all the key quantities ($\mathbf{p}_k^0$, $\mathbf{v}_k^k$, $\mathbf{g}^k$) are now linear. It is thus expected that VINS system may be solved in a linear fashion, even without any knowledge of the initial condition.

6.1.2 Linear Rotation Estimation

The IMU propagation model in (6.2) can only be linear if good rotation estimates are provided. Although integrating gyroscope measurements will lead to reasonable rotation estimates, they still drift over time. Therefore, we utilize additional epipolar constraints to eliminate rotation drift. We wish to estimate R_k^0, $k = 0, \cdots, N$, subject to following conditions:

$$\mathbf{R}_0^0 = \mathbf{I}_3, \quad \mathbf{R}_j^0 = \hat{\mathbf{R}}_j^i \cdot \mathbf{R}_i^0$$

where $\hat{\mathbf{R}}_j^i$ is a rotation that is obtained by either integrating gyroscope measurements between two consecutive images, or by finding the essential matrices between the current image and past images. For each incoming image, we try to compute the essential matrix between it and all other images within the sliding window.

As in [65], the above system can be solved linearly by relaxing orthonormality constraints of the rotations. Specifically, for a pair of rotation matrices $\mathbf{R}_i^0$, $\mathbf{R}_j^0$, and their relative constraint $\hat{\mathbf{R}}_j^i$, we have:

$$\left[\mathbf{I}_3, -\hat{\mathbf{R}}_j^i\right] \begin{bmatrix} \mathbf{r}_i^k \\ \mathbf{r}_j^k \end{bmatrix} = 0 \quad k = 1, 2, 3 \tag{6.3}$$

where $\mathbf{r}_i^k$ is the k^{th} column of $\mathbf{R}_i^0$. The solution of the relaxed approximate rotation matrices can be found as the last three columns of the right singular matrix of the system (6.3), which forms approximate rotation matrices $\bar{\mathbf{R}}_i^0$. Given the singular value decomposition of the approximate matrix $\bar{\mathbf{R}}_i^0 = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$, The true rotation matrices can then be obtained by enforcing unit singular values, which gives: $\mathbf{R}_i^0 = \mathbf{U}\mathbf{V}^{\mathrm{T}}$. After this point, we assume that the rotation components within the sliding window is known and noise-free.

6.1.3 Linear Sliding Window Estimator

We apply a tightly-coupled, sliding window graph-based [103] formulation due to its constant computation complexity and its ability to incorporate constrains from multiple observations to refine its solution. The *full state* vector can be expressed as (the transpose is ignored for the simplicity of presentation):

$$\begin{aligned}
\mathcal{X} &= \left[\mathbf{x}_0^0,\, \mathbf{x}_1^0,\, \cdots \mathbf{x}_N^0,\, \lambda_0,\, \lambda_1,\, \cdots \lambda_M\right] \\
\mathbf{x}_k^0 &= \left[\mathbf{p}_k^0,\, \mathbf{v}_k^k,\, \mathbf{g}^k\right] \quad \text{for } k = 1, ..., N \\
\mathbf{p}_0^0 &= [0,\, 0,\, 0]
\end{aligned}$$

where $\mathbf{x}_k^0$ is the k^{th} *camera state*, N is the number of camera states in the sliding window, M is the number of all features that have been observed for at least twice and have sufficient parallax within the sliding window. λ_l is the depth of the l^{th} point feature from its first observation. We are able to use a one-dimensional representation for features due to the nature of the underlying image processing pipeline (Sect. 6.6.1). This saves significant amount of computation power.

Since the rotation is fixed as in Sect. 6.1.2, we can formulate the linear VINS by gathering all measurements from both the IMU and the monocular camera and solve for the maximum likelihood estimate by minimizing the sum of the Mahalanobis norm of all measurement errors:

$$\min_{\mathcal{X}} \left\{ (\mathbf{b}_p - \mathbf{\Lambda}_p \mathcal{X}) + \sum_{k \in \mathcal{D}} \left\| \hat{\mathbf{z}}_{k+1}^k - \mathbf{H}_{k+1}^k \mathcal{X} \right\|_{\mathbf{P}_{k+1}^k}^2 + \sum_{(l,j) \in \mathcal{C}} \left\| \hat{\mathbf{z}}_l^j - \mathbf{H}_l^j \mathcal{X} \right\|_{\mathbf{P}_l^j}^2 \right\} \tag{6.4}$$

where the measurement triplets $\{\hat{\mathbf{z}}_{k+1}^k,\, \mathbf{H}_{k+1}^k,\, \mathbf{P}_{k+1}^k\}$ and $\{\hat{\mathbf{z}}_l^j,\, \mathbf{H}_l^j,\, \mathbf{P}_l^j\}$ are defined in Sect. 6.1.4 and Sect. 6.1.5 respectively. $\mathcal{D}$ is the set of all IMU measurements. $\mathcal{C}$ is the set of all observations between any features and any camera states within the sliding window. $\{\mathbf{b}_p,\, \mathbf{\Lambda}_p\}$ is the *optional* prior for the system. This system can be solved by reorganizing in the following form:

$$\left(\mathbf{\Lambda}_p + \mathbf{\Lambda}_{\mathcal{D}} + \mathbf{\Lambda}_{\mathcal{C}}\right) \mathcal{X} = \left(\mathbf{b}_p + \mathbf{b}_{\mathcal{D}} + \mathbf{b}_{\mathcal{C}}\right) \tag{6.5}$$

where $\{\mathbf{\Lambda}_{\mathcal{D}},\, \mathbf{b}_{\mathcal{D}}\}$ and $\{\mathbf{\Lambda}_{\mathcal{C}},\, \mathbf{b}_{\mathcal{C}}\}$ are information matrices and vectors for IMU and camera measurements respectively.

It should be noted that since the cost is linear with respect to the states, the system in (6.5) can have unique solution *without* the prior (initial condition):

$$\left(\mathbf{\Lambda}_{\mathcal{D}} + \mathbf{\Lambda}_{\mathcal{C}}\right) \mathcal{X} = \left(\mathbf{b}_{\mathcal{D}} + \mathbf{b}_{\mathcal{C}}\right) \tag{6.6}$$

This is the key to enable dynamic launching and failure recovery of MAVs. However, as will be shown in Sect. 6.3, there are degenerate motions for the monocular VINS setup, which will render the scale unobservable using only measurements within the sliding window. In such case, it is desirable to marginalize out states that are about to be removed from the window and convert them a prior to (implicitly) propagate the scale.

6.1.4 IMU Measurement Model

Given the locally drift-free rotation, we can rewrite (6.2) as a linear function of the state $\mathcal{X}$:

$$\hat{\mathbf{z}}_{k+1}^k = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{k+1}^k \\ \hat{\boldsymbol{\beta}}_{k+1}^k \\ \hat{\mathbf{0}} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_0^k \left(\mathbf{p}_{k+1}^0 - \mathbf{p}_k^0\right) - \mathbf{v}_k^k \Delta t + \mathbf{g}^k \frac{\Delta t^2}{2} \\ \mathbf{R}_{k+1}^k \mathbf{v}_{k+1}^{k+1} - \mathbf{v}_k^k + \mathbf{g}^k \Delta t \\ \mathbf{R}_{k+1}^k \mathbf{g}^{k+1} - \mathbf{g}^k \end{bmatrix} = \mathbf{H}_{k+1}^k \mathcal{X} + \mathbf{n}_{k+1}^k \tag{6.7}$$

where $\mathbf{n}_{k+1}^k$ is the additive measurement noise. We estimate the gravity vector for each pose. The last block line in (6.7) represents prediction of the gravity vector. All variables

except the position component are independent of the accumulated rotation $\mathbf{R}_k^0$, making them insensitive to rotation error. The linear IMU measurement covariance has the form:

$$\mathbf{P}_{k+1}^k = \begin{bmatrix} {}^{\alpha\beta}\mathbf{P}_{k+1}^k & \mathbf{0} \\ \mathbf{0} & {}^{\mathbf{g}}\mathbf{P}_{k+1}^k \end{bmatrix}$$

Note that the terms $\hat{\boldsymbol{\alpha}}_{k+1}^k$ and $\hat{\boldsymbol{\beta}}_{k+1}^k$ and correlated since they both come from IMU measurements within $[k, k+1]$. Their joint covariance matrix ${}^{\alpha\beta}\mathbf{P}_{k+1}^k$ can be calculated using the pre-integration technique proposed in [61].

6.1.5 Camera Measurement Model

Let the l^{th} feature be first detected in the i^{th} frame. The observation of this feature in the j^{th} normalized image plane $[\hat{u}_l^j,\ \hat{v}_l^j]^{\mathrm{T}}$ can be expressed as:

$$\lambda_l^j \begin{bmatrix} \hat{u}_l^j \\ \hat{v}_l^j \\ 1 \end{bmatrix} = \mathbf{R}_0^j \left(\begin{bmatrix} \mathbf{p}_i^0 - \mathbf{p}_j^0 + \lambda_l \mathbf{R}_i^0 \begin{bmatrix} u_l^i \\ v_l^i \\ 1 \end{bmatrix} \end{bmatrix} \right) \tag{6.8}$$

where λ_l^j is the depth of the feature in the j^{th} frame. We use tracking, instead of a descriptor-based method, as the tool for data association (Sect 6.6.1). As such, the first observation *defines* the direction of a feature, and $[u_l^i,\ v_l^i]$ is noiseless. Note that (6.8) is now linear with respect to the state, but nonlinear to the image measurement since the depth is initially unknown. The unknown depth transforms into a unknown weighting factor to the measurement covariance. Still, we can rewrite (6.8) as:

$$\hat{\mathbf{z}}_l^j = \hat{\mathbf{0}} = \begin{bmatrix} -1 & 0 & \hat{u}_l^j \\ 0 & -1 & \hat{v}_l^j \end{bmatrix} \mathbf{R}_0^j \left(\begin{bmatrix} \mathbf{p}_i^0 - \mathbf{p}_j^0 + \lambda_l \mathbf{R}_i^0 \begin{bmatrix} u_l^i \\ v_l^i \\ 1 \end{bmatrix} \end{bmatrix} \right) = \mathbf{H}_l^j \mathcal{X} + \mathbf{n}_l^j$$

where $\mathbf{n}_l^j$ is the additive measurement noise. and the camera measurement covariance has the form:

$$\mathbf{P}_l^j = \lambda_l^{j^2} \bar{\mathbf{P}}_l^j$$

where $\bar{\mathbf{P}}_l^j$ is the feature observation noise in the normalized image plane. Note that although λ_l^j is initially unknown, we can initialize it as the average depth of the scene. In practice, we found the solution very insensitive to the initial value of λ_l^j as long as it is set to be *larger* than the actual depth.

6.2 Nonlinear Optimization

As the monocular VINS is initialized with the linear sliding window approach described Sect. 6.1, we are able to use a nonlinear optimization framework to jointly optimize both the translation and rotation components of the system to obtain more accurate results. Our nonlinear solver operates on the tangent space of the rotation group to avoid singularities and to better approximate rotation errors.

Note since a large number of parameters in the nonlinear optimization shares the same physical meaning as those in the linear formulation (Sect. 6.1), here we introduce slight a abuse of notations by reusing the symbols for the state vector and measurement matrices/jacobians.

6.2.1 Formulation

Similar to the linear formulation, we still use a tightly-coupled, sliding window graph-based [103] formulation for nonlinear optimization Here we do minor changes to the *full state* vector as (the transpose is again ignored for the simplicity of presentation):

$$\begin{aligned}
\mathcal{X} &= \left[\mathbf{x}_0^0,\, \mathbf{x}_1^0,\, \cdots \mathbf{x}_N^0,\, \lambda_0,\, \lambda_1,\, \cdots \lambda_M\right] \\
\mathbf{x}_k^0 &= \left[\mathbf{p}_k^0,\, \mathbf{v}_k^k,\, \mathbf{q}_k^0\right] \quad \text{for } k = 1, ..., N \\
\mathbf{p}_0^0 &= \left[0,\, 0,\, 0\right], \quad \mathbf{q}_0^0 = \left[0,\, 0,\, 0,\, 1\right]
\end{aligned}$$

where the definition of $\mathbf{x}_k^0$, $\mathbf{v}_k^k$, and λ_l remains the same. However, we change the gravity vector into the rotation quaternion ($\mathbf{q} = [q_x,\, q_y,\, q_z,\, q_w]$) of the camera with respect to the first camera state. The use of quaternion avoids singularities in rotation, as oppose to

Euler angle-based representations. We use the Hamilton notation for quaternions. Note now the dimension of the state vector is not the same as the dimension of the degree of freedom of the system as rotation, which has only 3-DOF, is over-parameterized by the four dimension quaternion.

Similar to (6.4), we aim to find a configuration of the state parameters that produce the maximum a priori estimates by minimizing the sum of the Mahalanobis norm of all measurement errors:

$$\min_{\mathcal{X}} \left\{ (\mathbf{b}_p - \mathbf{\Lambda}_p \mathcal{X}) + \sum_{k \in \mathcal{D}} \left\| r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^{k},\, \mathcal{X}) \right\|_{\mathbf{P}_{k+1}^{k}}^{2} + \sum_{(l,j) \in \mathcal{C}} \left\| r_{\mathcal{C}}(\hat{\mathbf{z}}_{l}^{j},\, \mathcal{X}) \right\|_{\mathbf{P}_{l}^{j}}^{2} \right\} \tag{6.9}$$

where $r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^{k},\, \mathcal{X})$ and $r_{\mathcal{C}}(\hat{\mathbf{z}}_{l}^{j},\, \mathcal{X})$ are residuals of IMU and camera measurements, which will be presented in Sect. 6.2.2 and Sect. 6.2.3 respectively.

Although the residuals for position, velocity, and feature depth can be easily defined:

$$\begin{aligned} \mathbf{p} &= \hat{\mathbf{p}} + \delta \mathbf{p} \\ \mathbf{v} &= \hat{\mathbf{v}} + \delta \mathbf{v} \\ \lambda &= \hat{\lambda} + \delta \lambda \end{aligned} \tag{6.10}$$

The residual of rotation is more involved. To this end, we use the perturbation of the tangent space of the rotation manifold as the minimum dimension representation of rotation residual. We first define the error quaternion term $\delta \mathbf{q}$ as the different between the estimated and true quaternions:

$$\mathbf{q} = \hat{\mathbf{q}} \otimes \delta \mathbf{q} \tag{6.11}$$

where $\otimes$ is the quaternion multiplication operator Since the error quaternion term is usually small, we can approximate it as:

$$\delta \mathbf{q} \approx \begin{bmatrix} \frac{1}{2} \delta \boldsymbol{\theta} \\ 1 \end{bmatrix} \tag{6.12}$$

from which we can use the three dimension error vector $\delta \boldsymbol{\theta}$ as the representation of rotation residual. Similarly, we can write the error term in the form of rotation matrix:

$$\mathbf{R} \approx \hat{\mathbf{R}} \cdot (\mathbb{I} + \lfloor \delta \boldsymbol{\theta} \times \rfloor) \tag{6.13}$$

where $\lfloor \delta\boldsymbol{\theta} \times \rfloor$ is the skew-symmetric matrix from $\delta\boldsymbol{\theta}$.

Following this definition, we operate on the error state representation during the optimization:

$$\delta\mathcal{X} = \left[\delta\mathbf{x}_0^0,\, \delta\mathbf{x}_1^0,\, \cdots \delta\mathbf{x}_N^0,\, \delta\lambda_0,\, \delta\lambda_1,\, \cdots \delta\lambda_M\right]$$
$$\delta\mathbf{x}_k^0 = \left[\delta\mathbf{p}_k^0,\, \delta\mathbf{v}_k^k,\, \delta\boldsymbol{\theta}_k^0\right] \quad \text{for } k = 1, ..., N$$

We linearize the cost function (6.9) with respect to $\delta\mathcal{X}$, and iteratively minimize the cost of the resulting linear system. Given the current best estimates the state $\hat{\mathcal{X}}$, we have:

$$\begin{aligned}\min_{\delta\mathcal{X}} \Bigg\{ &\left(\mathbf{b}_p - \boldsymbol{\Lambda}_p\hat{\mathcal{X}}\right) + \sum_{k\in\mathcal{D}} \left\| r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k,\, \hat{\mathcal{X}}) + \mathbf{H}_{k+1}^k \delta\mathcal{X} \right\|_{\mathbf{P}_{k+1}^k}^2 \\ &+ \sum_{(l,j)\in\mathcal{C}} \left\| r_{\mathcal{C}}(\hat{\mathbf{z}}_l^j,\, \hat{\mathcal{X}}) + \mathbf{H}_l^j \delta\mathcal{X} \right\|_{\mathbf{P}_l^j}^2 \Bigg\}\end{aligned} \tag{6.14}$$

where $\mathbf{H}_{k+1}^k$ and $\mathbf{H}_l^j$, which will also be defined in Sect. 6.2.2 and Sect. 6.2.3, are the jacobians of $r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k,\, \hat{\mathcal{X}})$ and $r_{\mathcal{C}}(\hat{\mathbf{z}}_l^j,\, \hat{\mathcal{X}})$ with respect to $\delta\mathcal{X}$, respectively. The system (6.14) can be rewritten and solved as:

$$\left(\boldsymbol{\Lambda}_p + \boldsymbol{\Lambda}_{\mathcal{D}} + \boldsymbol{\Lambda}_{\mathcal{C}}\right)\delta\mathcal{X} = \left(\mathbf{b}_p + \mathbf{b}_{\mathcal{D}} + \mathbf{b}_{\mathcal{C}}\right) \tag{6.15}$$

after which the state estimates can be updated as:

$$\hat{\mathcal{X}} = \hat{\mathcal{X}} \oplus \delta\mathcal{X} \tag{6.16}$$

where $\oplus$ is the compound operator that has the form of simple addition for position, velocity, and feature depth as in (6.10), but is formulated as quaternion multiplication for rotations as in (6.11) and (6.12).

6.2.2 IMU Measurement Model

We now present the formulation for the IMU measurement $\hat{\mathbf{z}}_{k+1}^k$, the measurement covariance matrix $\mathbf{P}_{k+1}^k$, the residual $r_{\mathcal{D}}(\hat{\mathbf{z}}_{k+1}^k,\, \mathcal{X})$, and the measurement jacobian $\mathbf{H}_{k+1}^k$.

It should be first noted that since there are multiple accelerometer and gyroscope measurements between two images, the IMU measurement $\hat{\mathbf{z}}_{k+1}^k$ is a composition of multiple IMU readings.

$$\hat{\mathbf{z}}_{k+1}^k = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{k+1}^k \\ \hat{\boldsymbol{\beta}}_{k+1}^k \\ \hat{\mathbf{q}}_{k+1}^k \end{bmatrix} = \begin{bmatrix} \iint_{t\in[k,k+1]} \hat{\mathbf{R}}_t^k \hat{\mathbf{a}}_t^b dt^2 \\ \int_{t\in[k,k+1]} \hat{\mathbf{R}}_t^k \hat{\mathbf{a}}_t^b dt \\ \int_{t\in[k,k+1]} \boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_t^b) \hat{\mathbf{q}}_t^k dt \end{bmatrix} \tag{6.17}$$

where

$$\hat{\mathbf{a}}_t^b = \mathbf{a}_t^b + {}^a\mathbf{n}_t$$
$$\hat{\boldsymbol{\omega}}_t^b = \boldsymbol{\omega}_t^b + {}^\omega\mathbf{n}_t$$

are accelerometer and gyroscope measurements that are corrupted with additive noise, and

$$\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_t^b) = \frac{1}{2} \begin{bmatrix} -\lfloor \hat{\boldsymbol{\omega}}_t^b \times \rfloor & \hat{\boldsymbol{\omega}}_t^b \\ -\hat{\boldsymbol{\omega}}_t^{b\mathrm{T}} & \mathbf{0} \end{bmatrix}$$

$\hat{\mathbf{R}}_t^k$ can be derived from $\hat{\mathbf{q}}_t^k$. Again, while error terms for $\hat{\boldsymbol{\alpha}}_{k+1}^k$ and $\hat{\boldsymbol{\beta}}_{k+1}^k$ are still additive, since $\hat{\mathbf{q}}_{k+1}^k$ is over-parameterized, we define its error terms as the perturbation from the true value:

$$\mathbf{q}_{k+1}^k \approx \hat{\mathbf{q}}_{k+1}^k \otimes \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}_{k+1}^k \\ 1 \end{bmatrix} \tag{6.18}$$

With the approximated rotation matrix composition of the error term (6.13), we can derive the continuous-time linearized dynamics of the error terms from (6.17) and (6.18):

$$\begin{bmatrix} \delta\dot{\boldsymbol{\alpha}} \\ \delta\dot{\boldsymbol{\beta}} \\ \delta\dot{\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\hat{\mathbf{R}}_t^k \lfloor \hat{\mathbf{a}}_t^b \times \rfloor \\ \mathbf{0} & \mathbf{0} & -\lfloor \hat{\boldsymbol{\omega}}_t^b \times \rfloor \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\alpha}_t^k \\ \delta\boldsymbol{\beta}_t^k \\ \delta\boldsymbol{\theta}_t^k \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -\hat{\mathbf{R}}_t^k & \mathbf{0} \\ \mathbf{0} & -\mathbb{I} \end{bmatrix} \begin{bmatrix} {}^a\mathbf{n}_t \\ {}^\omega\mathbf{n}_t \end{bmatrix} = \mathbf{F}_t \delta\mathbf{z}_t^k + \mathbf{G}_t \mathbf{n}_t$$

from which we can derived the first-order discrete-time covariance update equation in order to recursively compute $\mathbf{P}_{k+1}^k$ with the initial covariance $\mathbf{P}_k^k = \mathbb{I}$:

$$\mathbf{P}_{t+\delta t}^k = (\mathbb{I} + \mathbf{F}_t \delta t) \cdot \mathbf{P}_t^k \cdot (\mathbb{I} + \mathbf{F}_t \delta t)^{\mathrm{T}} + (\mathbf{G}_t \delta t) \cdot \mathbf{Q}_t \cdot (\mathbf{G}_t \delta t)^{\mathrm{T}}, \quad t \in [k, k+1] \tag{6.19}$$

where δt is the time between two IMU measurements, and $\mathbf{Q}_t$ is the diagonal covariance matrix for IMU measurements.

We can now define the measurement residual $r_{\mathcal{D}}(\hat{\mathbf{z}}^k_{k+1},\, \mathcal{X})$ with respect to the final error terms of the IMU measurement $\left[\delta\boldsymbol{\alpha}^k_{k+1},\, \delta\boldsymbol{\beta}^k_{k+1},\, \delta\boldsymbol{\theta}^k_{k+1}\right]^{\mathrm{T}}$. Following (6.2) and (6.18), we have:

$$r_{\mathcal{D}}(\hat{\mathbf{z}}^k_{k+1},\, \mathcal{X}) = \begin{bmatrix} \delta\boldsymbol{\alpha}^k_{k+1} \\ \delta\boldsymbol{\beta}^k_{k+1} \\ \delta\boldsymbol{\theta}^k_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^k_0 \left(\mathbf{p}^0_{k+1} - \mathbf{p}^0_k + \mathbf{g}^0 \frac{\Delta t^2}{2} \right) - \mathbf{v}^k_k \Delta t - \hat{\boldsymbol{\alpha}}^k_{k+1} \\ \mathbf{R}^k_0 \left(\mathbf{R}^0_{k+1} \mathbf{v}^{k+1}_{k+1} + \mathbf{g}^0 \Delta t \right) - \mathbf{v}^k_k - \hat{\boldsymbol{\beta}}^k_{k+1} \\ 2 \left[\hat{\mathbf{q}}^{k^{-1}}_{k+1} \otimes \mathbf{q}^{0^{-1}}_k \otimes \mathbf{q}^0_{k+1} \right]_{xyz} \end{bmatrix} \tag{6.20}$$

where the gravity vector of the first camera state $\mathbf{g}^0$ is solved by linear formulation (Sect. 6.1), and $(\cdot)_{xyz}$ extracts the vector part of a quaternion.

Using (6.13) and the fact that $\mathbf{R}^0_{k+1} = \mathbf{R}^0_k \cdot \mathbf{R}^k_{k+1}$, we can write:

$$\hat{\mathbf{R}}^0_{k+1} \cdot \left(\mathbb{I} + \lfloor \delta\boldsymbol{\theta}^0_{k+1} \times \rfloor \right) \approx \hat{\mathbf{R}}^0_k \cdot \left(\mathbb{I} + \lfloor \delta\boldsymbol{\theta}^0_k \times \rfloor \right) \cdot \hat{\mathbf{R}}^k_{k+1} \cdot \left(\mathbb{I} + \lfloor \delta\boldsymbol{\theta}^k_{k+1} \times \rfloor \right)$$

which can be converted into the following by ignoring higher order terms:

$$\delta\boldsymbol{\theta}^0_{k+1} = \hat{\mathbf{R}}^{k+1}_0 \cdot \hat{\mathbf{R}}^0_k \cdot \delta\boldsymbol{\theta}^0_k + \delta\boldsymbol{\theta}^k_{k+1}$$

which provides a simple linearized form of the propagation of rotation error terms. As such, the jacobian of the IMU measurement residual with respect to the error state can be obtained as:

$$\begin{aligned} \mathbf{H}^k_{k+1} &= \begin{bmatrix} \frac{\partial r_{\mathcal{D}}}{\partial \delta \mathbf{x}_k} & \frac{\partial r_{\mathcal{D}}}{\partial \delta \mathbf{x}_{k+1}} \end{bmatrix} \\ \frac{\partial r_{\mathcal{D}}}{\partial \delta \mathbf{x}_k} &= \begin{bmatrix} -\mathbf{R}^k_0 & -\Delta t \mathbb{I} & \lfloor \mathbf{R}^k_0 \cdot (\mathbf{p}^0_{k+1} - \mathbf{p}^0_k + \mathbf{g}^0 \frac{\Delta t^2}{2}) \times \rfloor \\ \mathbf{0} & -\mathbb{I} & \lfloor \mathbf{R}^k_0 \cdot (\mathbf{R}^0_{k+1} \mathbf{v}^{k+1}_{k+1} + \mathbf{g}^0 \Delta t) \times \rfloor \\ \mathbf{0} & \mathbf{0} & -\mathbf{R}^{k+1}_0 \cdot \mathbf{R}^0_k \end{bmatrix} \\ \frac{\partial r_{\mathcal{D}}}{\partial \delta \mathbf{x}_{k+1}} &= \begin{bmatrix} \mathbf{R}^k_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^k_0 \cdot \mathbf{R}^0_{k+1} & -\mathbf{R}^k_0 \cdot \mathbf{R}^0_{k+1} \lfloor \mathbf{v}^{k+1}_{k+1} \times \rfloor \\ \mathbf{0} & \mathbf{0} & \mathbb{I} \end{bmatrix} \end{aligned} \tag{6.21}$$

Equations (6.17), (6.19), (6.20), and (6.21) define all required quantities to specify the IMU measurement model.

6.2.3 Camera Measurement Model

The formulation of the camera measurement model is straightforward. The feature measurement is the observation of the feature in the normalized image plane:

$$\hat{\mathbf{z}}_l^j = \begin{bmatrix} \hat{u}_l^j \\ \hat{v}_l^j \end{bmatrix} \tag{6.22}$$

The residual term is the reprojection error, which is defined as:

$$\begin{aligned} r_{\mathcal{C}}(\hat{\mathbf{z}}_l^j,\, \mathcal{X}) &= \begin{bmatrix} \frac{{}^f x_l^j}{{}^f z_l^j} - \hat{u}_l^j \\ \frac{{}^f y_l^j}{{}^f z_l^j} - \hat{v}_l^j \end{bmatrix} \\ \mathbf{f}_l^j &= \begin{bmatrix} {}^f x_l^j \\ {}^f y_l^j \\ {}^f z_l^j \end{bmatrix} = \mathbf{R}_0^j \left(\mathbf{p}_i^0 - \mathbf{p}_j^0 + \lambda_l \mathbf{R}_i^0 \mathbf{u}_l^i \right) \end{aligned} \tag{6.23}$$

where $\mathbf{u}_l^i = [u_l^i,\, v_l^i,\, 1]^{\mathsf{T}}$. The residual covariance is the diagonal feature measurement noise matrix $\mathbf{P}_l^j$.

The jacobian can be obtained by utilizing (6.13) and apply chain rule on (6.23):

$$\begin{aligned} \mathbf{H}_l^j &= \begin{bmatrix} \frac{1}{{}^f z_l^j} & 0 & -\frac{{}^f x_l^j}{{}^f z_l^{j2}} \\ 0 & \frac{1}{{}^f z_l^j} & -\frac{{}^f y_l^j}{{}^f z_l^{j2}} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{f}_l^j}{\partial \delta \mathbf{x}_i} & \frac{\partial \mathbf{f}_l^j}{\partial \delta \mathbf{x}_j} & \frac{\partial \mathbf{f}_l^j}{\partial \delta \lambda_l} \end{bmatrix} \\ \frac{\partial \mathbf{f}_l^j}{\partial \delta \mathbf{x}_i} &= \begin{bmatrix} \mathbf{R}_0^j & \mathbf{0} & \mathbf{R}_0^j \cdot \mathbf{R}_i^0 \left\lfloor \lambda_l \mathbf{u}_l^i \times \right\rfloor \end{bmatrix} \\ \frac{\partial \mathbf{f}_l^j}{\partial \delta \mathbf{x}_j} &= \begin{bmatrix} -\mathbf{R}_0^j & \mathbf{0} & \left\lfloor \mathbf{R}_0^j \left(\mathbf{p}_i^0 - \mathbf{p}_j^0 + \lambda_l \mathbf{R}_i^0 \mathbf{u}_l^i \right) \times \right\rfloor \end{bmatrix} \\ \frac{\partial \mathbf{f}_l^j}{\partial \delta \lambda_l} &= \mathbf{R}_0^j \cdot \mathbf{R}_i^0 \mathbf{u}_l^i \end{aligned} \tag{6.24}$$

Equations (6.22), (6.23), and (6.24) define all required quantities to specify the camera measurement model.

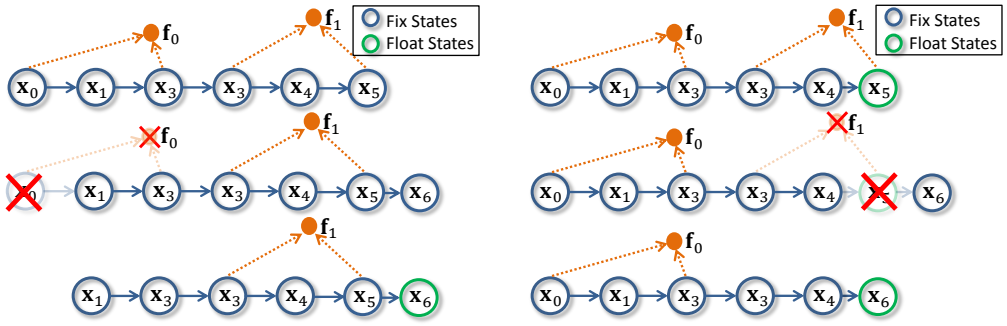
6.3 Handling Scale Ambiguity via Two-Way Marginalization

It is well known that in order to have the scale of a monocular VINS be observable, the IMU has to excite nonzero accelerations in at least two axes [37, 41, 66]. When the vehicle is undergoing degenerate motions, such as constant velocity or hovering, and without any prior information, it can be verified that the position and velocity components of the solution of (6.6) in this situation can be scaled arbitrary without violating any constraints. Unfortunately, zero acceleration motion is unavoidable for a hover-capable MAV and it must be handled properly.

If the vehicle first undergoes generic motions with sufficient excitation in acceleration with the time interval $[0, n]$, and then enters constant velocity motion during the interval $[n+1, n+N]$, the scale can only be observable if the camera states correspond to generic motion are included in the sliding window. This is unrealistic if available computation only allows N camera states in the sliding window. However, if we can provide an initial estimate of $\mathbf{x}_{n+1}^0$, we will be able to propagate (not observe) the scale from n to $n+1$. Naturally, this can be done by proper marginalization of $\mathbf{x}_n^0$ as it is removed from the window at the $(n+N)^{th}$ step.

For hovering, as proved in [47], if the vehicle first undergoes generic motions with sufficient acceleration excitation during $[0, N-1]$, and then enters a hover start from the N^{th} image, the scale observability can be preserved by using a last-in-first-out (LIFO) sliding window scheme. [47] performs state-only measurement update during hovering, and covariance is updated only once as the vehicle exits hovering. This is due to the fact that features are not kept in the state, but instead marginalized out as the covariance update is performed. However, this approach will lead to pessimistic covariance as observations obtained during hovering is not used to update the covariance.

Based on the previous discussions, we propose a novel two-way marginalization scheme to handle both constant velocity and hovering cases. The pseudo code is shown



(a) (b)

Figure 6.1: Fig 6.1(a) shows the structure of the full state before, during, and after marginalizing a recent camera state ($\mathbf{x}_5$) after a newer camera state $\mathbf{x}_6$ is added. Similar marginalization process of the oldest camera state is shown in Fig. 6.1(b).

in Algorithm 1. Consider the full state vector $\mathcal{X} = [\mathbf{x}_0^0, \cdots \mathbf{x}_{N-1}^0 \,|\, \lambda_{\mathcal{L}}]$, where $\lambda_{\mathcal{L}}$ is the set of all features have been added into the sliding window. We add the next camera state ($\mathbf{x}_N^0$) to the sliding window if any of the following two criteria are satisfied:

1. The time between two images Δt is larger than δ.

2. After compensating the relative rotation, the average parallax of all common features between the most recent two images is larger than ε.

The first condition ensures that the error in the integrated IMU measurement between two camera states is bounded, while second condition ensures that the new camera state is added when translation motion of the vehicle with respect to the scene is significant. We require that all newly added features $\lambda_{\mathcal{L}^+}$ to have at least two observations and with sufficient parallax to ensure successful triangulation (Line 1). The system is then solved with all available measurements within the sliding window plus any available prior (Line 2).

We keep a variable $s = \textit{float}/\textit{fix}$ to indicate whether we should marginalize out the second newest camera state ($\mathbf{x}_{N-1}^0$) or the oldest one ($\mathbf{x}_0^0$). To marginalize a chosen camera state $\mathbf{x}_k^0$, we first remove the camera state and all features $\lambda_{\mathcal{L}^-}$ that are first observed

by it (Lines 5 and 13). We then construct a new prior based on all measurements related to the removed states (Lines 4 and 12):

$$\mathbf{\Lambda}_p = \mathbf{\Lambda}_p + \sum_{k \in \mathcal{D}^-} \mathbf{H}_{k+1}^{k^T} \mathbf{P}_{k+1}^{k^{-1}} \mathbf{H}_{k+1}^{k} + \sum_{(l,j) \in \mathcal{C}^-} \mathbf{H}_{l}^{j^T} \mathbf{P}_{l}^{j^{-1}} \mathbf{H}_{l}^{j} \tag{6.25}$$

where $\mathcal{D}^-$ and $\mathcal{C}^-$ are sets of removed IMU and camera measurements respectively. The marginalization can the be carried out via Schur Complement [103].

The value of s is reevaluated after the marginalization based solely on the parallax between two most recent remaining images (Lines 6-9 and 14-17). Intuitively, our approach will keep removing the recent camera states if the vehicle has small or no motion. Keeping older camera states in this case will preserve acceleration information that is necessary to recover the scale, while still storing all information provided by the marginalized states as prior. On the other hand, if the vehicle is under fast constant speed motion, older camera states will be removed and converted into priors for the subsequent estimates. We do note that the scale in this case is subject to drifting. However, without global loop closure, marginalization is the best that can be done to propagate the scale information forward, while still maintaining constant computation complexity. Fig. 6.1 illustrates different working scenarios of the proposed marginalization approach.

6.4 Initialization and Failure Recovery

Our linear VINS formulation (Sect. 6.1) naturally allows on-the-fly initialization and failure recovery. In fact, these two tasks are identical as both involve solving the linear system (6.5) with no priors.

After initialization with the linear formulation, we use the nonlinear optimization approach (Sect. 6.2) to achieve better accuracy. We maintain two subsystems, which corresponds to two arrays of image/IMU measurements and camera/feature states. The first subsystem is solved by the nonlinear solver and two-way marginalization is used to

Algorithm 1 Two-Way Marginalization

Require:

$$\mathcal{X} \leftarrow \left[\mathbf{x}_0^0,\, \cdots \mathbf{x}_{N-1}^0 \,|\, \lambda_{\mathcal{L}}\right]$$

$$s \leftarrow \textit{float}\ \textbf{or}\ \textit{fix}$$

$$\{\mathbf{\Lambda}_p,\, \mathbf{b}_p\} \leftarrow \text{Prior Information}$$

Ensure: $\Delta t > \delta$ **or** Parallax$(\mathbf{x}_{N-1}^0\,,\, \mathbf{x}_N^0) > \varepsilon$

$\mathcal{X} \leftarrow \mathcal{X} \cup [\mathbf{x}_N^0 \,|\, \lambda_{\mathcal{L}^+}]$

Solve $\mathcal{X}$ using (6.14) and (6.15)

if $s =$ *float* **then**

  $\{\mathbf{\Lambda}_p,\, \mathbf{b}_p\} \leftarrow$ Marginalization$(\mathbf{x}_{N-1}^0,\, \lambda_{\mathcal{L}^-})$

  $\mathcal{X} \leftarrow \mathcal{X} \setminus [\mathbf{x}_{N-1}^0 \,|\, \lambda_{\mathcal{L}^-}]$

  if Parallax$(\mathbf{x}_{N-2}^0\,,\, \mathbf{x}_N^0) < \varepsilon$ **then**

    $s \leftarrow \textit{float}$

  else

    $s \leftarrow \textit{fix}$

  end if

else

  $\{\mathbf{\Lambda}_p,\, \mathbf{b}_p\} \leftarrow$ Marginalization$(\mathbf{x}_0^0,\, \lambda_{\mathcal{L}^-})$

  $\mathcal{X} \leftarrow \mathcal{X} \setminus [\mathbf{x}_0^0 \,|\, \lambda_{\mathcal{L}^-}]$

  if Parallax$(\mathbf{x}_{N-1}^0\,,\, \mathbf{x}_N^0) < \varepsilon$ **then**

    $s \leftarrow \textit{float}$

  else

    $s \leftarrow \textit{fix}$

  end if

end if

return $\{\mathcal{X},\, \mathbf{\Lambda}_p,\, \mathbf{b}_p,\, s\}$

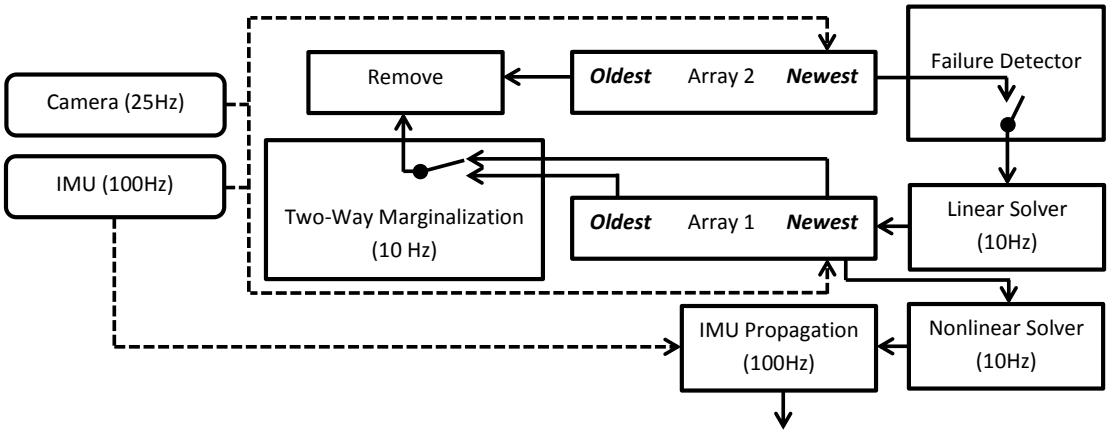


Figure 6.2: Block diagram of the proposed failure recovery approach. Note the switching mechanism for two-way marginalization and the failure detector. After initialization, the nonlinear optimizer is used to provide state estimates for navigation.

avoid scale ambiguity. In this subsystem, the newest two camera states may be separated arbitrarily far in time if the vehicle is hovering. On the other hand, the second subsystem refreshes independent of the vehicle motion. It always maintains a queue structure where the oldest camera state and its corresponding measurements are removed as a new image comes in. This way, we make sure that the IMU measurements in the queue always have bounded error.

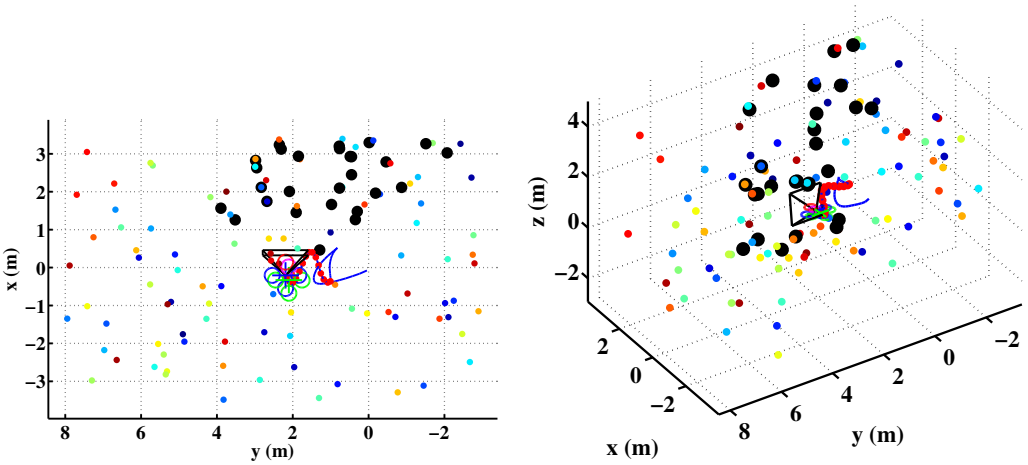
When the system is in normal operation, only the first subsystem is solved and the second subsystem only collects data. When failure occurs, the first array, as well as all prior information, are discarded. Instead, we repeatedly try to find a valid solution using measurements within the second subsystem using the linear approach. Once a valid solution is found, we put all measurements back to the first array and resume normal operation with the nonlinear solver. The block diagram of this failure recovery mechanism is shown in Fig. 6.2. We detect failure/validate solution by looking into the convergent property of the nonlinear solver. These failure modes can be the result of bad data association or insufficient feature observations, both are major source of failures in featureless environments.

6.5 Simulation Results

The proposed linear initialization approach is first verified in simulations. We implemented a comprehensive simulator that consists of both realistic VINS sensor data simulation and quadrotor dynamics simulation. The quadrotor dynamics matches with the actual experimental platform as shown in Fig. 1.2(d). The simulated environment is randomly populated with 100 features. The simulated camera that runs at 10 Hz has VGA resolution and 90 degrees field of view without lens distortion. Feature observations are corrupted with Gaussian noise with 1 pixel standard deviation. No outliers are presented in the simulation. The simulated IMU runs at 100 Hz. The additive accelerometer and gyroscope noise has 0.1 m/s^2 and 0.1 rad/s standard deviation, respectively, which matches the noise characteristics of a realistic IMU. We use 15 camera states in the sliding window estimator. In the simulation, the linear rotation estimation (Sect. 6.1.2) is disabled to illustrate the case of insufficient features in the environment.

The goal of the simulation is to verify that the initial values given by the linear approach is close to the global minimum, and such that the nonlinear optimization is able to converge to the correct solution. We run multiple trials. In each trial, features and waypoints are randomly generated. The simulated platform then tracks minimum jerk trajectories (Sect. 7.3) generated from those waypoints using the ground truth states for feedback control. A screenshot of a typical simulation trial is shown in Fig 6.3.

We are interested in the changes to the cost of the nonlinear VINS system (6.9). We examine the cost function by performing a dense sampling (more than 10^6 samples for $\lambda \in [0, 1]$) of the convex combination of the linear initial estimates and the ground truth. Fig. 6.4 shows the linear initialization is very close to the global optimal in the sense that that the cost variation is convex between the linear initial estimates and the ground truth. Due to the convexity, it can be verified that the global minimum of the system, which lies very close to the ground truth, can be obtained via classic iterative nonlinear optimization techniques such as the Gauss-Newton algorithm.



(a) (b)

Figure 6.3: 2D and 3D screenshots of the VINS simulation environment. Features are shown in color. Estimated features are shown as large black dots.

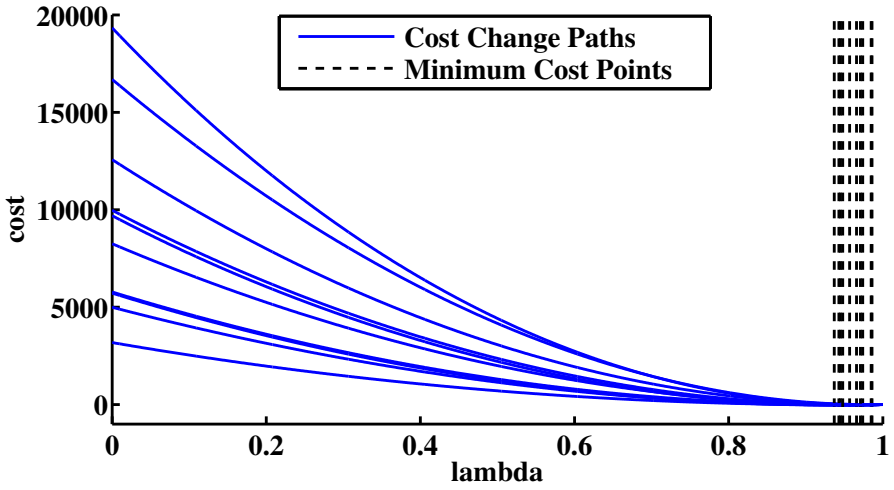


Figure 6.4: Changes in the nonlinear VINS cost function (6.9) within the region between the linear initial estimates ($\lambda = 0$) and the ground truth ($\lambda = 1$). The cost of each curve is offset by the cost of the ground truth. The globally minimum cost are indicated by dashed lines. Note that due to noisy sensor data, the state parameters that lead to the minimum cost are close to, but not equal to the ground truth. Differences in cost are caused by differences in the number of observed features.

6.6 Experimental Results

The configuration of the experimental platform is discussed in Sect. 1.3 and shown in Fig. 1.2(d). Planning, trajectory generation and control methodologies are presented in Ch. 7. The platform is based on the Pelican quadrotor from Ascending Technologies, GmbH. This platform is natively equipped with an AutoPilot board consisting of an IMU and a user-programmable ARM7 microcontroller. The main computation unit onboard is an Intel NUC with a 1.8 GHz Core i3 processor with 8 GB of RAM and a 120 GB SSD. The only additions to onboard sensing are a mvBlueFOX-MLC200w grayscale HDR camera with standard lens that capture 752×480 images at 25 Hz, and a Microstrain 3DM-GX2 IMU. The use of an additional IMU is not for getting better IMU measurements, but for assembling a sensor suite that is rigidly configured and with good time synchronization. The total mass of the platform is 1.28kg, which leads to a thrust to weight ratio of approximately two. The entire algorithm is developed in C++ using ROS as the interfacing robotics middleware.

6.6.1 Real-Time Implementation

Although the onboard camera captures images at 25 Hz, it is both computationally infeasible and unnecessary to perform the optimization (Sect. 6.1 and 6.2) at such a high rate. The system starts with one camera state in the sliding window, and a fixed number of corner features detected in that camera image. Features are tracked in the high-rate image sequence until the next camera state is added to the sliding window (Sect. 6.3). The pre-integrated IMU measurement (Sect. 6.1.4) is also computed as new camera state is added.

We utilize multi-thread implementation to achieve real-time operation. Three threads run concurrently. The first thread is the image processing front end that we just described. The second thread is the main VINS optimizer (Sect. 6.1 or 6.2) and the marginalization module (Sect. 6.3). Finally, due to computation constraint, our VINS system runs

Module	Time (ms)	Rate (Hz)	Thread
Feature Tracking	5	25	1
Add New Camera State and Features	9	10	1
Linear Sliding Window Estimator	12	10	2
Nonlinear Sliding Window Estimator	35	10	2
Marginalization	17	10	2
IMU Forward Propagation	1	100	3

Table 6.1: Computation breakdown of the system with 30 camera states and 200 features.

at 10 Hz with approximate processing latency of 35 ms. This is not sufficient for autonomous control of MAVs. We therefore implement a third thread to propagate the latest solution from the optimizer forward using the high-rate IMU measurements. The output of this thread is used directly as the feedback for the trajectory tracking controller. A breakdown of computation time of each components in our system is shown in Table. 6.1. It suggests that our algorithm is able to run stably onboard.

6.6.2 Implementation Details and Choice of Parameters

We maintain $N = 30$ camera states and $M = 200$ features in the sliding window. These choices reflects maximum utilization of the available computation power. For each incoming image, we try to detect 400 features with a minimum separation of 30 pixels using Shi-Tomasi corners [102] and track them using the KLT tracker [59]. The enforcement of feature separation is to avoid numerical issues due to poorly distributed features. At this point, we apply RANSAC with epipolar constraints for outlier rejection with a threshold of 1 pixel.

All tracked features are used for rotation estimation (Sect. 6.1.2) during the linear initialization phase. However, we enforce each tracked feature to reach a parallax of at least 30 pixels (distance in the image frame after rotation compensation) before its added

into the sliding window for depth estimation. The choice of the minimum parallax is for avoiding numerical issues in triangulation of noisy feature observations with small baseline. During the two-way marginalization (Sect. 6.3), all features that are first observed in the marginalized frame are also marginalized and removed. All features measurements that are made in the removed frame are also marginalized. We marginalize and remove additional feature if the total number of features goes beyond 200.

We determine the addition of frames to the sliding window based on $\delta = 0.1s$ (Algorithm 1), which implies that the sliding window estimator runs at $10\,\text{Hz}$. We decide whether a frame/camera state is *float*/*fix* based on a parallax threshold of $\varepsilon = 30\,\text{pixels}$. This threshold is set to satisfy two requirements: 1) There is sufficient parallax to ensure feature triangulation between frames; and 2) There is sufficient overlap between images to ensure feature matching. To deal with motions that is mostly rotation, we additionally force setting a frame to *fix* if the total number of matched features drops below 50.

In the nonlinear optimization, we propagate camera states using pre-integrated IMU measurements and triangulate the depth of each newly added features using current pose estimates. Although we assume all measurements are corrupted by Gaussian noise, we use the Huber kernel [33] for increased robustness against feature measurement outliers. The cutoff threshold for the Huber kernel is set to be 2 pixels. In practice, since gross outliers will result in very low weights due to the Huber kernel, which may lead to singularities in dimensions correspond to poorly observed features. In this case, we remove features from the optimization if their conditional information, which correspond to diagonal terms in the information matrix ($\mathbf{\Lambda}_{\mathcal{C}}$ in (6.15)), drops below 1. In practice, this threshold mainly depends on the stability of the underlying matrix solver, it can be set to some small positive number as far as numerical issues are avoided.

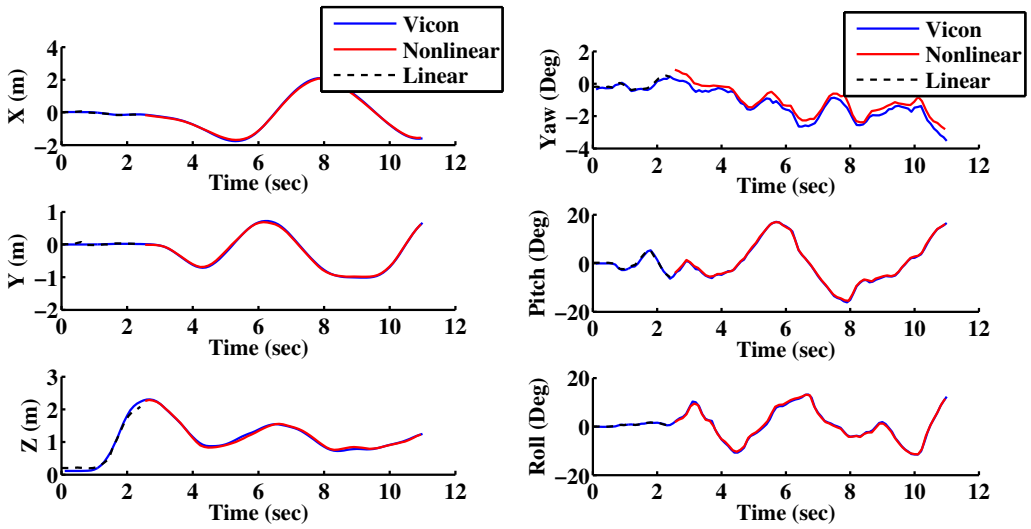
In the linear estimator, biases of the IMU is considered as zero. In the nonlinear optimization, it is straightforward to include bias parameters for each camera state. However, we experimentally found that the bias for our IMU is almost always zero. For the sake of saving computation power, we do not include bias terms in the optimization.

6.6.3 Initialization Performance

We present two experiments to validate the proposed linear initialization approach as well as the performance of the overall system in a lab space equipped with motion capture system.

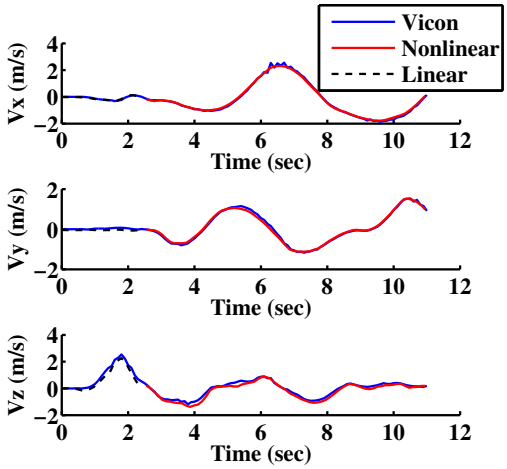
In the first experiment, the MAV takes off from stationary, but without any knowledge of the environment. Due to the lack of direct scale measurement of monocular VINS, the initialization of the estimator can only be performed after the MAV takes off. In this case, since we know that the MAV starts from stationary (although the MAV does not know this information), and the oldest pose in the sliding window is still on the ground when the estimator initialized, we are able to align the ground truth pose with the onboard pose estimates. As shown in Fig. 6.5, the linear initialization approach successfully generates an initial estimate that is close to the ground truth, after which the nonlinear optimization takes over and the proposed system is able to track the pose and velocity of the MAV accurately with minimum drift.

In the second experiment, we highlight the ability of the proposed approach to perform on-the-fly initialization. The estimator start when the MAV is already flying at high-speed. As shown in Fig. 6.6, we are able to recover the body frame velocity as well as the gravity vector, which is represented by roll and pitch angles in the plot. Note that these are actually the critical navigation states to enable stabilization of the MAV. We do not show comparisons of position and yaw since we have no alignment between the ground truth and the onboard pose estimation. Since in this case the MAV is already in rapid motion, the linear estimator immediately obtains a good solution and then pass the task to the nonlinear optimizer. Table. 6.2 shows the refinement of the nonlinear optimizer towards to the ground truth, which matches the simulation results in Fig. 6.4. For the same case, Fig. 6.7 shows the convergence of the nonlinear estimator in each iteration.



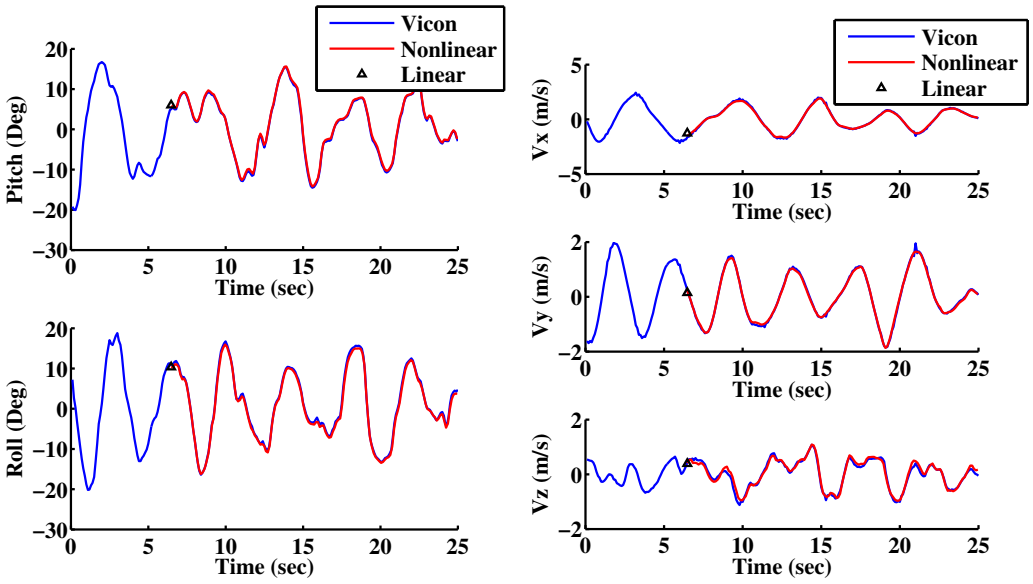
(a) Position

(b) Orientation



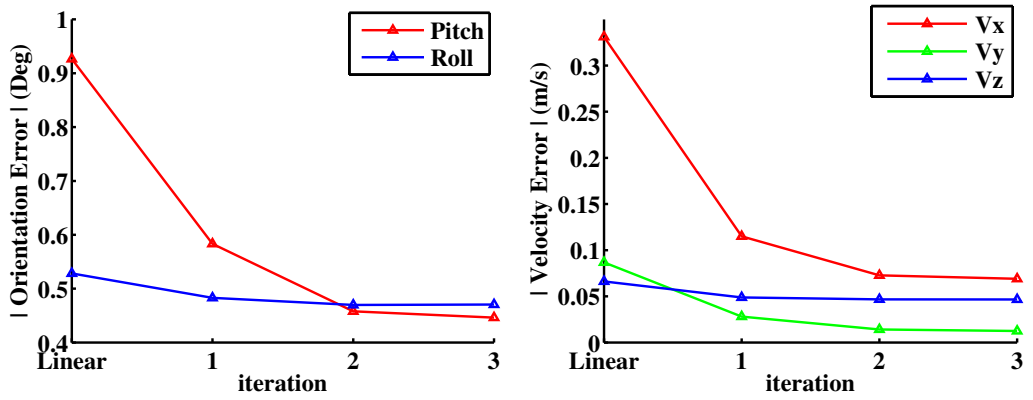
(c) Velocity

Figure 6.5: The MAV takes off from stationary but without any knowledge of the environment. The linear initialization can only be performed when the MAV starts flying, after which the nonlinear optimizer takes over and perform accurate tracking of the vehicle motion.



(a) Orientation (b) Velocity

Figure 6.6: On-the-fly initialization from nontrivial velocity and attitude. Since the MAV is already in rapid motion when the estimator starts, the linear estimator immediately obtain a good initial estimate and then hands off to the nonlinear optimizer. Note the accurate tracking performance even with significant acceleration and angular velocity. The convergence of the nonlinear optimizer towards the ground truth given the linear estimate (triangle in the plot) is show in Fig. 6.7 and Table 6.2.



(a) Orientation (b) Velocity

Figure 6.7: Convergence of the nonlinear optimizer towards the ground truth in three iterations given linear initial estimates during the experiment in Fig. 6.6.

	v_x (m/s)	v_y (m/s)	v_z (m/s)	Pitch (deg)	Roll (deg)
Vicon	-1.601	0.234	0.455	5.069	10.87
Nonlinear	-1.532	0.222	0.408	5.515	10.40
Linear	-1.270	0.147	0.389	5.559	10.34

Table 6.2: Convergence of the nonlinear optimizer towards the ground truth given linear initialization results.

6.6.4 Autonomous Hovering

One major issue of using a monocular VINS sensor suite for autonomous MAVs is the lack of direct measurement of metric scale. We use this experiment to highlight the effectiveness of our two-way marginalization (Sect. 6.3) scheme. After the on-the-fly initialization is completed, the MAV is commanded to hover using its onboard state estimates for feedback control. As shown in Fig 6.8, since the two-way marginalization always removes newer camera states while hovering, previous feature observations that have sufficient parallax, as well as IMU measurements that have sufficient accelerations are kept. As such, almost-drift-free estimates in full 6 degree-of-freedom is obtained. Very minor drift can be found in the error plot. This is due to small drifts in the KLT tracker, which may be avoided with descriptor-based feature matching. However, we defer this implementation as future work. During hovering, the onboard position estimate compares well with the ground truth with standard deviation of $\{0.0099,\ 0.0124,\ 0.0081\}$ meters. The precision of hovering using such onboard estimates for feedback control has the standard deviation of $\{0.0282,\ 0.0307,\ 0.0161\}$.

Recently, [58], presents a system that utilize the state estimates from the Google Tango[1] for feedback control of a quadrotor. Their data shows that the Tango state estimates has the standard deviation of $\{0.0165,\ 0.0320,\ 0.0274\}$ meters in position. This suggests that our approach is comparable or even better than the Tango while the platform

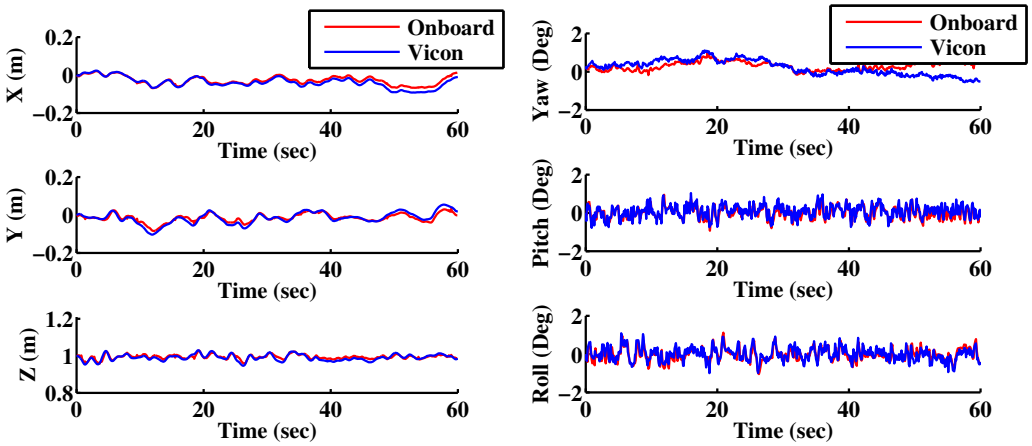
[1] http://www.google.com/atap/projecttango

is hovering, despite the fact that the Tango has the advantage of having a RGB-D camera for direct scale observation. We suspect this is due to the fact that the Tango is aimed for handheld application and does not take the mechanical vibrations from the quadrotor into account.

We also compare the methodology as proposed in [47] with our two-way marginalization scheme. Since the source code of [47] is not available, we compare the theoretical equivalent by making a slight change in our sliding window formulation. During marginalization of a *float* frame, we do not convert image measurement made in that frame into prior, but rather just drop them. This is equivalent to the state-only measurement update during hovering as image measurements do not contribute to the update to the covariance matrix (or information matrix in our case). While the position and orientation estimates remain close for both approaches, as shown in Fig. 6.9, this approach will result in pessimistic covariance/information, which reflects as fluctuation the velocity due to weaker "bound" from the initial guess.

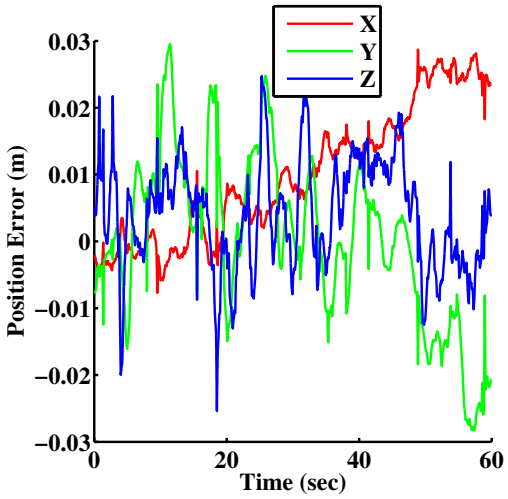
6.6.5 Autonomous Trajectory Tracking

In this experiment, we test the performance of autonomous trajectory tracking while using the onboard state estimate for feedback control. The MAV is commanded to track a figure eight pattern with each circle being 0.9 meters in radius, as shown in Fig. 6.10(d). Trajectory generation and control methodologies are introduced in Sect. 7. We conduct multiple trials of the same desired path but with different velocities. The desired velocity is given as a parameter for the minimum jerk trajectory generator (Sect. 7.3), although the actual velocity problem varies during the flight. Important performance metrics are shown in Table. 6.3. With different desired velocity, the maximum roll and pitch angles increases due to higher linear acceleration. The time required to complete the path is shortened accordingly. Meanwhile, we observe almost no difference in position and yaw drifting, which suggests that our approach is able to handle fast motions without

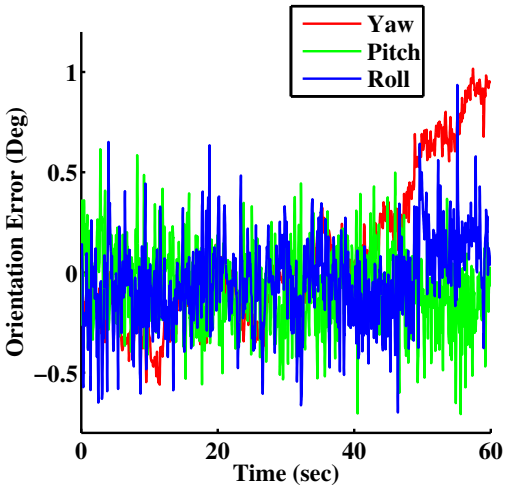


(a) Position

(b) Orientation



(c) Position Error



(d) Orientation Error

Figure 6.8: Hover performance of the MAV using onboard state estimates for feedback control comparing with ground truth. Note that there is almost no drift in all directions.

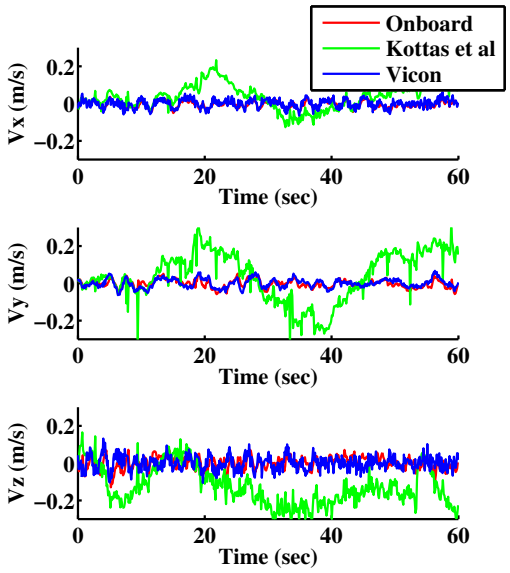
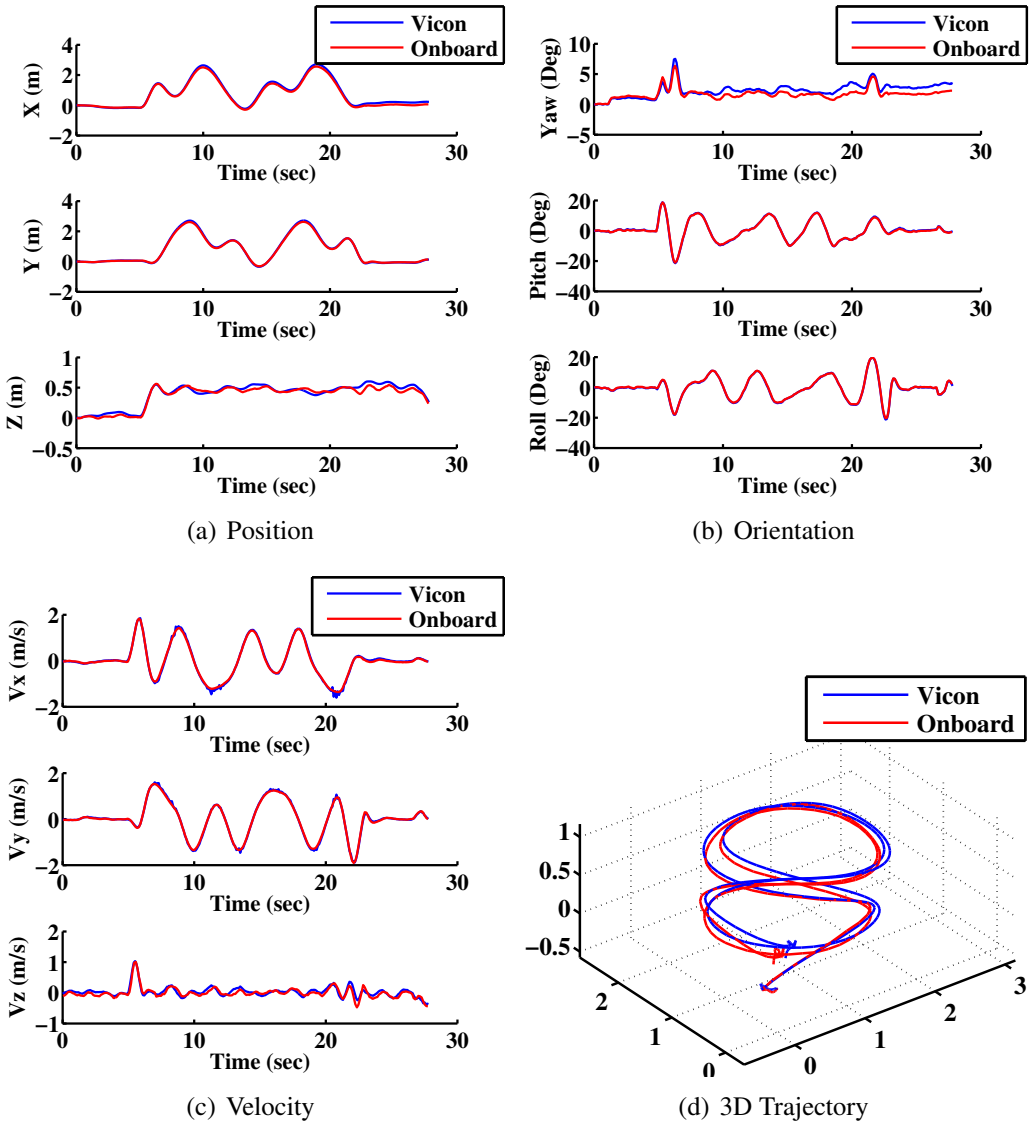


Figure 6.9: Comparison of velocity estimation during hovering with two-way marginalization and the method proposed in [47]. The state-only measurement update produces pessimistic covariance, which results in fluctuation in velocity estimates.

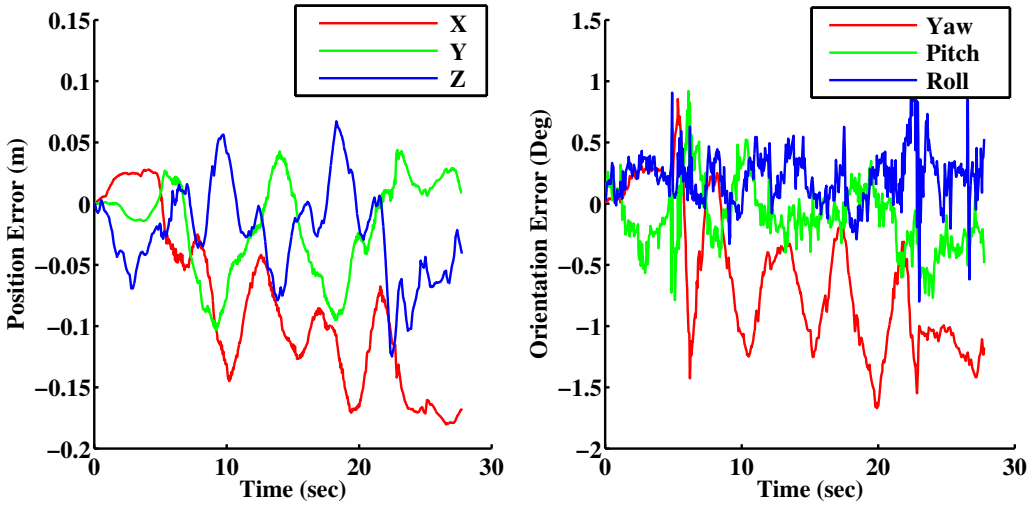
degeneration of performance. We do observe a slight increase in the standard deviations in velocity and roll and pitch, however, we do not observe any downgrade in terms of flight performance. The onboard velocity estimates are also comparable to those from the Google Tango powered flying robot [58], which reports a velocity standard deviation of $\{0.0651,\ 0.0533,\ 0.0788\}$ m/s. In Fig. 6.10, we highlight the fastest trial with a maximum velocity of 2 m/s and maximum roll/pitch angles close to 30 degrees. Error plots of the same trial is shown in Fig. 6.11. We note that due to numerical differentiation, the Vicon ground truth is actually noisier than the onboard state estimate as shown at 12 sec and 21 sec in Fig. 6.10(c). As such, the velocity error is expected to be smaller than the values reported on the plot.



(a) Position (b) Orientation

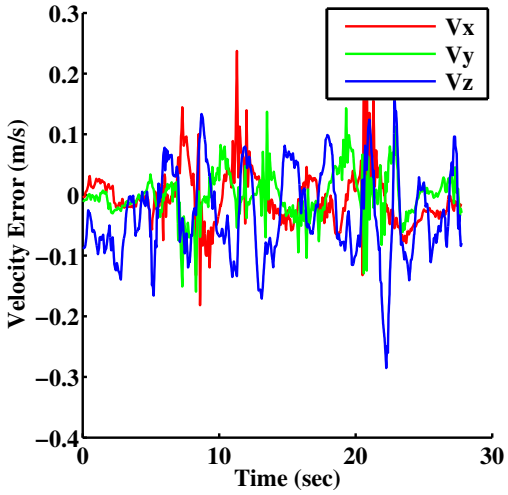
(c) Velocity (d) 3D Trajectory

Figure 6.10: Performance of autonomous trajectory tracking using onboard monocular VINS estimates for feedback control. The maximum speed is 2 m/s, while the maximum roll/pitch angle is 28 degrees. The onboard velocity and attitude estimates track the ground truth accurately. Small drifts position and yaw, which are known to be unobservable, are observed.



(a) Position Error

(b) Orientation Error



(c) Velocity Error

Figure 6.11: Error plots of trajectory tracking as in Fig. 6.10, illustrating slow drifts in position and yaw, but no drift in velocity and roll and pitch.

Trial	1	2	3	4	5	6	7
Duration (sec)	35.88	27.20	25.50	25.19	20.77	18.14	17.84
Max Velocity (m/s)	1.035	1.271	1.391	1.397	1.694	1.953	1.955
Max Roll/Pitch (deg)	6.047	9.089	11.51	13.34	20.71	26.19	28.04
Position Drift (m)	0.073	0.141	0.039	0.091	0.127	0.078	0.162
Yaw Drift (deg)	0.140	1.227	0.622	0.825	1.756	0.901	1.174
Velocity StdDev (m/s)	0.055	0.053	0.062	0.066	0.068	0.067	0.063
Roll/Pitch StdDev (deg)	0.160	0.190	0.251	0.204	0.311	0.332	0.262

Table 6.3: Performance metrics of autonomous trajectory tracking with varying speeds. Note that the position and yaw drifts are shown in magnitude only. The velocity and roll/pitch standard deviations are shown as maximum values between all dimensions (3 dimensions for velocity, 2 dimensions for roll/pitch).

6.6.6 Autonomous Flight in Indoor Environments

In this experiment, we show autonomous flight in the hallway of a typical office building using the proposed monocular VINS state estimates for feedback control. Fig. 6.12 shows images from the onboard camera during the experiment. Since no ground truth is available, we run multiple trials to verify the accuracy, consistency, and repeatability of the proposed approach. As shown in Fig. 6.14, we conduct two autonomous flight and one handheld experiments. Statistics of each trial is shown in Table. 6.4.

During autonomous flight experiments, the MAV is stabilized using onboard state estimates. The operator have the freedom of sending waypoints to the MAV, from which a minimum jerk trajectory will be generated and tracked by the vehicle (Sect. 7.3). The operator may also control the velocity of the MAV directly via the kinematic controller (Sect. 7.1). In both autonomous flight experiments, the MAV was able to fly stably and return to its starting position with only a small drift in the estimated poses.

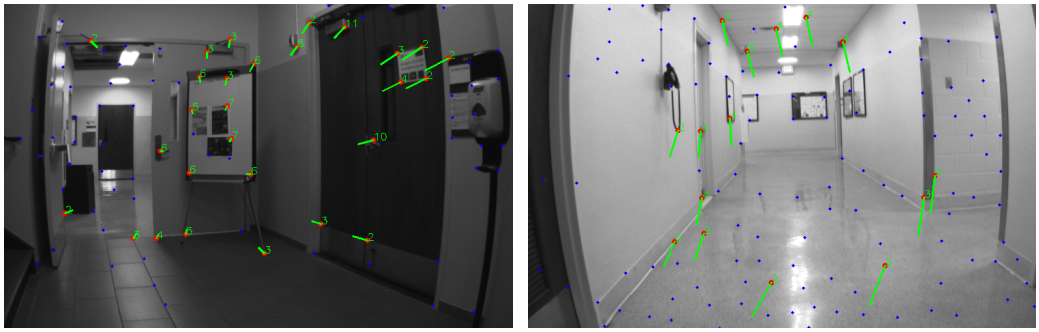
We do note that the estimation accuracy in autonomous flight is slightly worse than

Trial	Autonomous 1	Autonomous 2	Handheld
Duration (sec)	281.4	303.6	78.72
Distance (m)	81.39	87.52	63.38
Position Drift (m)	0.634	0.944	0.364
Position Drift %	0.78%	1.08%	0.58%
Yaw Drift (deg)	1.83	2.703	1.79
Yaw Drift %	0.51%	0.75%	0.50%

Table 6.4: Statistics of autonomous indoor flight and handheld experiments. Note that the handheld experiment runs much faster and with much less small motion (hovering), therefore both duration and distance are reduced. Drift is measured by putting the MAV back to the starting pose after the experiment.

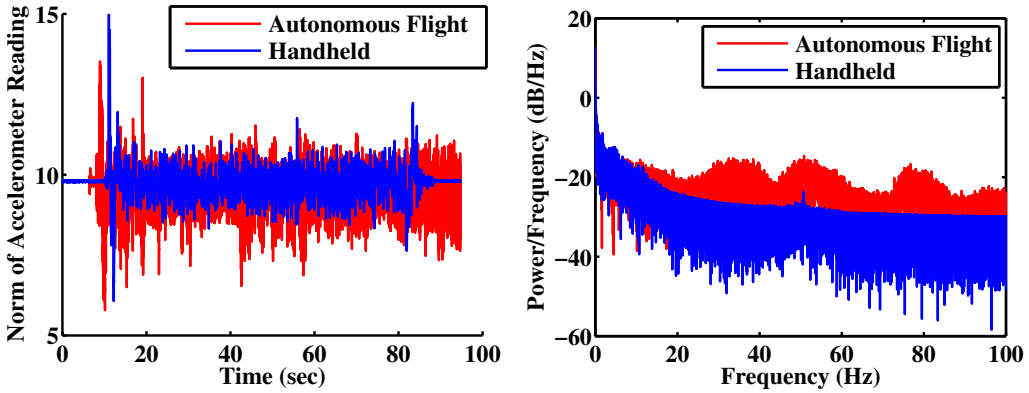
the handheld experiment. This is due to much noisier IMU measurements caused by mechanical vibrations from the motors while flying. A comparison in both time and frequency domain of IMU measurements during both in flight and handheld cases is shown in Fig. 6.13. In all trials, the onboard state estimator behaves consistently in the sense that the overall trajectory is well aligned. Note that since the visual scale is not directly observable in the monocular VINS setting, consistent scale in all trials indicates that the scale is correctly estimated.

For comparison, we also present the state estimates from the Tango in the same environment (Fig. 6.15). Since we do not have direct access to the Tango data, we can only obtain an approximation of the drift by counting the grids. These results suggest that our results are on par with those from the Tango, despite that the Tango has the advantage of having a RGB-D camera for direct scale observation.



(a) (b)

Figure 6.12: Images from the onboard camera during autonomous flight in indoor environments. red dots represent features that have valid depth from sliding window optimization, green lines shows the tracking of such features between the current and the latest *fix* frame. Blue dots are features that are being tracked but have not be added into the optimization. The criteria of choosing features are discussed in Sect. 6.6.2.



(a) (b)

Figure 6.13: Comparison of IMU noise characteristics in both time and frequency domains for in flight and handheld cases. Plots shows the variation of the magnitude of the IMU measurement across all axes. Note how the mechanical vibrations worsen IMU performance while in flight.

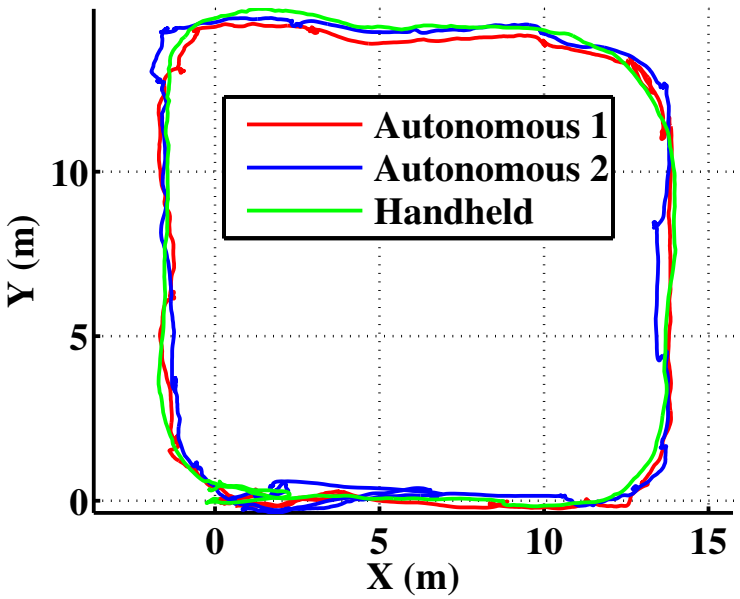


Figure 6.14: Trajectories of two autonomous flight and one handheld experiments. The onboard estimator is able to deliver repeatably results in the sense that the drift in all trials are similar, and the scale estimate is correct. Keep in mind that the visual scale is not directly observable in the monocular VINS setup, a consistent scale implies correct scale estimation.

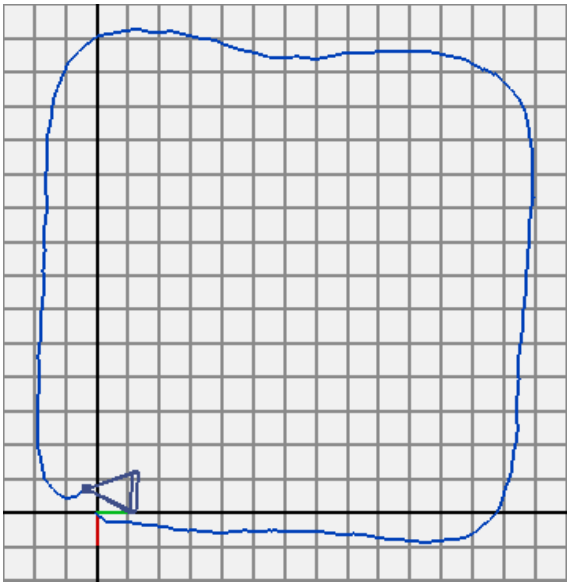


Figure 6.15: State estimation from the Google Tango in the same environment as Fig. 6.14 and with similar trajectory. Each grid in the graph is one meter wide. The final position drift is approximately $0.8\,\text{m}$.

6.6.7 State Estimation in Large-Scale Environments

In the final experiment, we push the limit of the proposed approach in complex large-scale indoor environments with challenging changes in lighting and texture conditions. A feature-rich and a relatively feature-less parts of the environment are shown in Fig. 6.18. In this experiment, the MAV is handheld due to 1) the length of the trajectory is higher than the feasible flight distance; and 2) failures are anticipated due to the challenging environmental conditions.

Fig. 6.16 shows the estimated paths of two trials conducted in the same environment but with different lighting conditions. Failures are highlighted with circles. Note that these trajectories are not perfectly aligned due to small errors in the initial placement of the vehicle. Both trials have position drift of approximately 5 meters. We do not list the exact number here as failures occurred during both trials, and as such the drift in position and yaw does not reflect the performance since the drift is dominated by the time that failure occurred. The earlier the failure, the longer the "lever arm" to amplify the error, resulting in larger drift. For comparison, we also show the trajectory obtained by the Google Tango (Fig. 6.17) in the same environment and lighting condition as the first trial and with similar trajectories. We note similar drifting behaviors between the proposed approach and the Tango. However, we need to keep in mind that the Tango has a RGB-D sensor to correct the visual scale, while our system does not has such advantage.

We now analyze typical cases of failures or performance downgrade while operating in complex environments. In this experiment, failures are considered as cases where the state performance is significantly downgraded due to the lack of features. However, in all failure cases, the nonlinear estimator was eventually able to recover as sufficient features are observed later. As highlighted in Fig. 6.16, failures cause significant discontinuities in the state estimates. We find that failure is almost alway caused by the lack of feature measurements, although the cause of insufficient feature measurement varies.

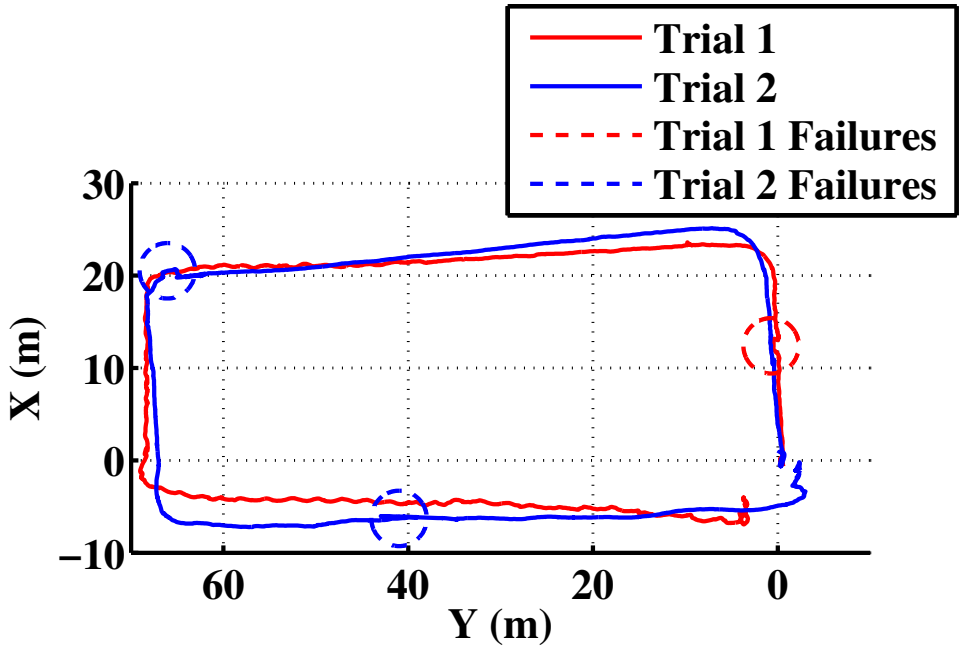


Figure 6.16: Estimated paths of two trials of state estimation experiment in large-scale environments. Failures in different trials are highlighted with circles of the corresponding color.

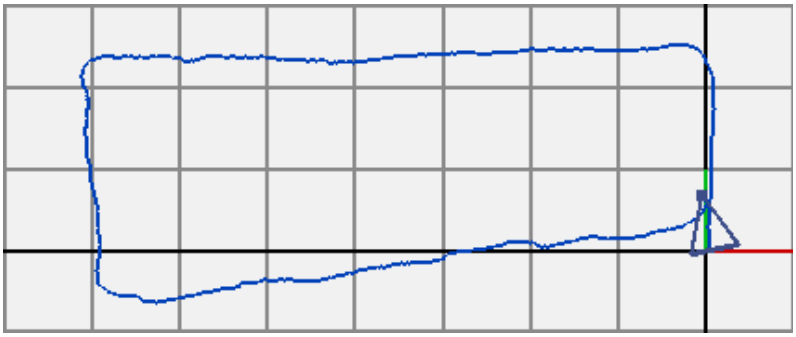
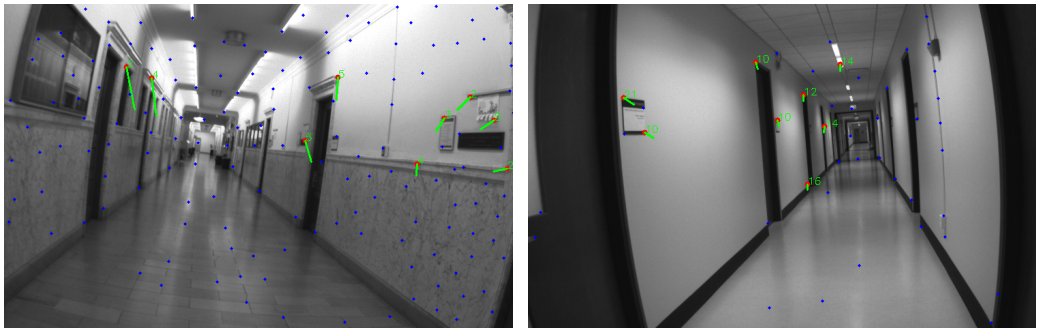


Figure 6.17: State estimation from the Google Tango in the same environment as Fig. 6.16. Each grid in the graph has the width of ten meters. The final position drift is approximately 6.0 m.



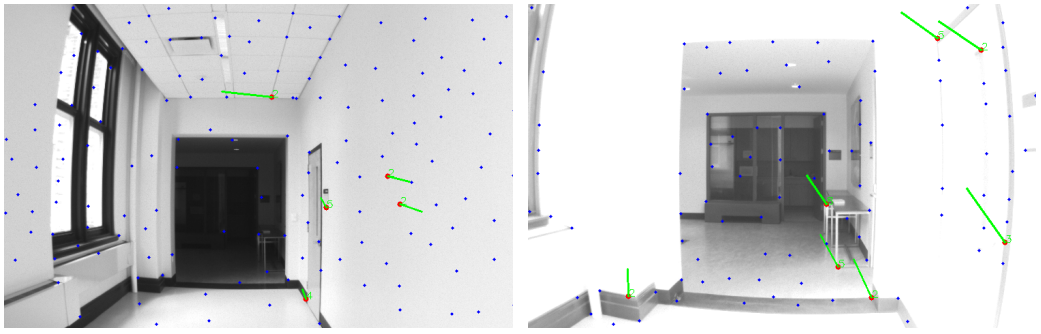
(a) (b)

Figure 6.18: Onboard images from large-scale environments, showing both feature-right (Fig. 6.18(a)) and relatively featureless (Fig. 6.18(b)) parts of the environment.

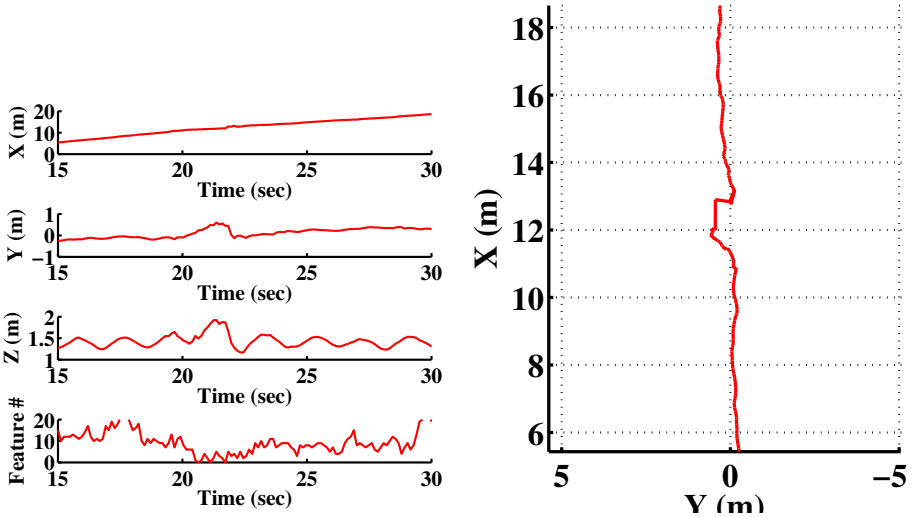
Fig. 6.19 show a challenging situation where the MAV moves from very bright but featureless area to very dark but feature-rich area. During the transition, the dynamic range of the scene is extremely high, causing no features being detected in the dark area (Fig. 6.19(a)). As the platform moves towards to the dark area, the bright area is oversaturated, which again causes lose of feature tracking (Fig. 6.19(b)). The number of features that are observed in the current frame and are used for the optimization frequently drop to zero (Fig. 6.19(c)), causing large noise in the state estimate.

In the second case, the platform perform a quick turn into a bright area when it is very close to a wall (Fig. 6.20). In such case, the platform quickly loses track of existing features due to rapid changes in the image. However, during a rapid rotation, there is limited baseline, and new features cannot be triangulated and added into the optimization immediately, again causing insufficient features in the current frame (Fig. 6.20(c)). The estimator was able to recover after sufficient parallax is obtained for the new features.

The last failure case is due to poor lighting conditions. As shown in Fig. 6.21, the platform first moves in a very dark corridor, which leads to in insufficient number of features and causes drift in visual scale. As the platform moves into brighter regions, corrections in the visual scale cause large discontinuities in the state estimates (Fig. 6.21(c)).

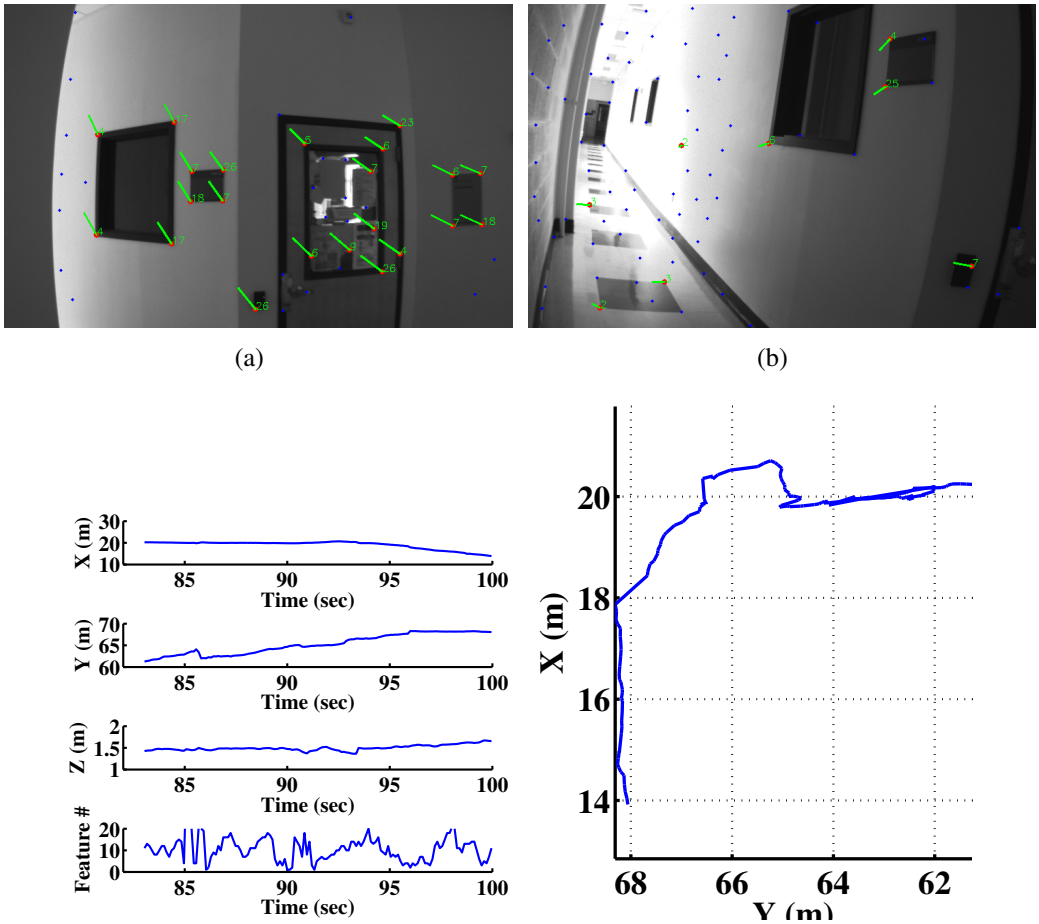


(a) (b)



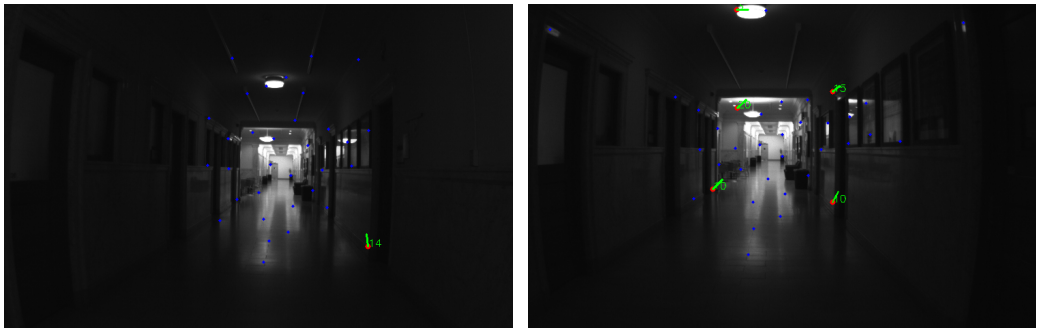
(c) Number of optimized features (d) Discontinuities in state estimates

Figure 6.19: Failure of the estimator due to high dynamic range of the scene. Occasionally, there is no feature in the current frame that is used for optimization (Fig. 6.19(c)).

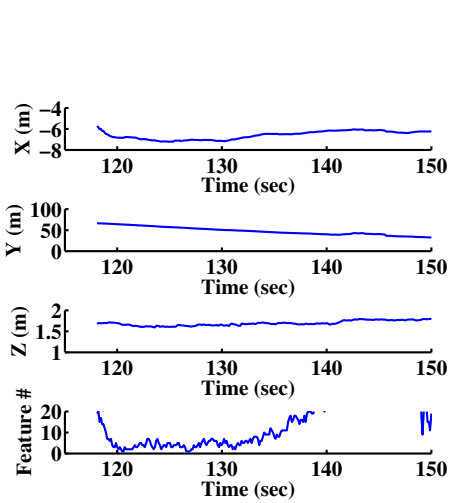


(c) Number of optimized features (d) Discontinuities in state estimates

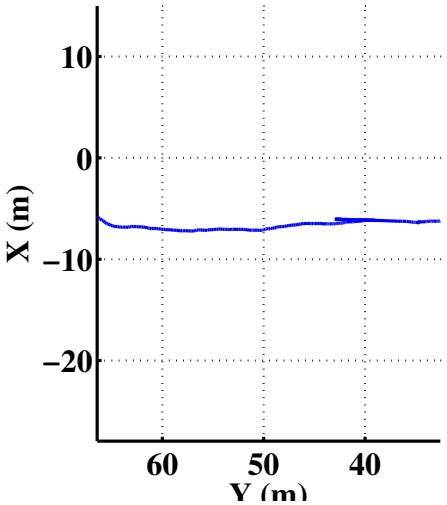
Figure 6.20: Failure of the estimator due to rapid rotation, which causes insufficient parallax for feature triangulation, and it is reflected as very small or even zero number optimized features in the current frame (Fig. 6.20(c)).



(a) (b)



(c) Number of optimized features



(d) Discontinuities in state estimates

Figure 6.21: Failure of the estimator due to poor lighting condition. The number of optimized features are constantly small in poorly lighted regions (Fig. 6.21(c)). The large correction in visual scale as the platform moves into brighter regions causes large discontinuities in the state estimates (Fig. 6.21(c)).

6.7 Discussion

In this chapter, we propose a novel approach for initialization and failure recovery for monocular visual-inertial system. We show that even with one camera, which is unable to provide direct distance measurement, it is still possible to recover all critical navigation states such as velocity and attitude without any prior knowledge of the system. We then propose a nonlinear solver that operates on the tangent space of the rotation manifold for continued high precision state estimation without any rotation singularities. We also address the issues of degenerate motions by proposing a two-way marginalization. We integrate all modules into a complete fail-safe system and demonstrate high-speed autonomous flight with average speed up to $2\,\text{m/s}$ in complex indoor environments. We present both autonomous flight and handheld experimental results in a variety of environments to demonstrate the performance of the proposed approach. From this chapter, we believe that the monocular VINS setup has great potential towards to miniature MAV platforms with tight SWaP constrains.

Chapter 7

Planning and Control

This chapter considers the adaption of planning and control methodologies to meet the need of aerial navigation. We first show control architectures that enable efficient human-robot interaction, and then focus on a system architecture that guarantees smoothness in both state estimation and control during planning and replanning.

7.1 Feedback Control

Given onboard state estimates, we command the MAV to track desired trajectories with a position tracking controller with nonlinear error metric [51] due to its superior performance in highly dynamical motions that involve large angle changes and significant accelerations. The onboard state estimates from the either the sensor fusion module (Ch. 5) or the monocular VINS system (Ch. 6) are used directly as the feedback for the controller. In our implementation, the attitude controller runs at $1\,\text{kHz}$ on the ARM processor on the vehicle's AutoPilot board, while the position tracking control operates at $100\,\text{Hz}$ on the onboard computer.

Although our goal is to develop a fully autonomous vehicle, at some point during the experiment, the human operator may wish to have simple, but direct control of the vehicle. As such, we developed a finite state machine-based hybrid-system controller

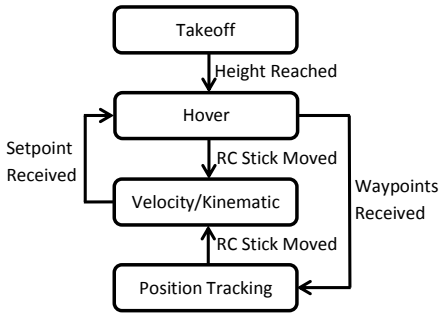


Figure 7.1: A finite state machine-based approach enables the operator to interact with the vehicle during experiments.

(Fig. 7.1) to allow human-robot interaction. There are four modes in this controller. The normal mode is position tracking mode. At any time, the operator is able to send inputs via a remote control. These commands are interpreted by the vehicle as kinematic velocity commands (where no commands result in hover state). We experimentally tested that the velocity control mode is easy to use in the sense that an untrained operator is able to control the vehicle without direct line-of-sight using only the 1 Hz image feed and the real-time 3D map visualization. The hover mode serves as an idle state, where the vehicle is waiting for commands from the operator.

7.2 High Level Planning

As the SLAM module generates globally consistent poses of the MAV and maps of the environment (Sect 3.3, 4.3.3, and 5.3), we wish to generate collision-free paths that enable the MAV to navigate autonomously within this map. Let the optimized pose in the SLAM frame denoted as $(\mathbf{p}_j^w, \mathbf{R}_j^w)$. The pose correction from the visual SLAM $\mathbf{d}_j^{we}$, which serves as the transform between the estimator frame and the SLAM frame, is formulated such that:

$$(\mathbf{p}_j^w, \mathbf{R}_j^w) = \mathbf{d}_j^{we} \oplus (\mathbf{p}_j^e, \mathbf{R}_j^e) \tag{7.1}$$

where $\oplus$ is the pose update function defined in [104] and $(\mathbf{p}_j^e,\, \mathbf{R}_j^e)$ is the corresponding pose in the estimator frame. In contrast to traditional approaches, we do not use $(\mathbf{p}_j^w,\, \mathbf{R}_j^w)$ as a global pose measurement for correcting the drift in the estimator (Sect. 5.2). Instead, we feed $\mathbf{d}_j^{we}$ into the trajectory generator (Sect. 7.3) and compute trajectories that are guaranteed to be smooth even if there are large discontinuities in the SLAM pose estimate (i.e. $\|\mathbf{d}_j^{we} \ominus \mathbf{d}_{j-1}^{we}\|$ is large, see Fig. 4.9(c))) due to loop closures or GPS corrections.

The key difference between the estimator frame and the SLAM frame is that the estimator frame drifts over time due to the lack of global measurements, while the SLAM frame remains globally consistent if loop closures of GPS measurements are available. Note that even if GPS measurements are available to the estimator, we still construct the drifting estimator frame via indirect GPS fusion (Sect. 5.3) in order to generate smooth state estimates, which are crucial for feedback control (Sect. 7.1).

We employ a two-stage planning approach. On a higher level, given user-specified waypoints in the SLAM frame, and treating the MAV as a cylinder, a high level path that connects the current vehicle position and the desired goal, which consists a sequence of desired 3D positions and yaw angles, is generated using the RRT* [40] implementation in the *Open Motion Planning Library* (OMPL) [107].

The resulting path is simplified by finding longest collision-free segments, which will eliminate intermediate waypoints. The resulting waypoints $\mathbf{g}_k^w$ are sent to the trajectory generator (Sect 7.3) for further refinement. The path is checked for possible collisions at the same frequency as the SLAM module.

7.3 Minimum Jerk Trajectory Generation

We first transform all waypoints from the high level planner into the estimator frame using the latest pose correction from SLAM (7.1):

$$\mathbf{g}_k^e = \ominus \mathbf{d}_j^{we} \oplus \mathbf{g}_k^w.$$

If the MAV flies through all transformed waypoints using state estimates in the estimator frame for feedback control, it will also fly through the same set of waypoints in the SLAM frame. Moreover, if there are large scale loop closures (i.e. large changes in $\mathbf{d}_j^{we}$), the set of waypoints that the vehicle is heading towards to will change significantly. However, if we are able to regenerate smooth trajectories with initial conditions equal to the current state of the vehicle, the transition between trajectories will be smooth and no special handling is needed within the onboard state estimator and the controller.

We wish to ensure that the MAV smoothly passes through all waypoints, while at the same time maintaining a reliable state estimate. A crucial condition that affects the quality of vision-based state estimates is the tracking performance. The effect of translation on the movement of a image features depends on the distances of the features, while rotation has uniform effect on all features, regardless of distance. Meanwhile, due to the underactuate nature of our quadrotor MAV platforms (4-DOF control versus 6-DOF pose), all translation motions are results of attitude changes. As shown Fig. 7.2 that fast translation has little effect on the tracking performance. However, fast rotation can blur the image easily, causing the failure of the feature tracker. This observation motivates us to design trajectories that minimize angular velocities in roll and pitch.

By differentiating the equation of motion of a quadrotor [71], it can be seen that the angular velocity of the body frame is affinely related to the jerk, which is the derivative of the linear acceleration. As such, we generate trajectories that minimize the jerk of the quadrotor in horizontal directions. For the vertical direction, we wish to minimize the RPM changes of the motors, which again correspond to the jerk. In order to avoid large deviations from the high level path, intermediate waypoints are added shortly before and after a waypoint if the angle between the two line segments that connect this waypoint exceeds a threshold. We utilize a polynomial trajectory generation algorithm [83], which is derived from [69], that runs onboard the vehicle with a runtime on the order of 10 ms. Optimal trajectories can be found by solving the following unconstrained quadratic

programming:

$$\min_{\mathbf{y}} \ \mathbf{y}^{\mathrm{T}} Q \mathbf{y}$$

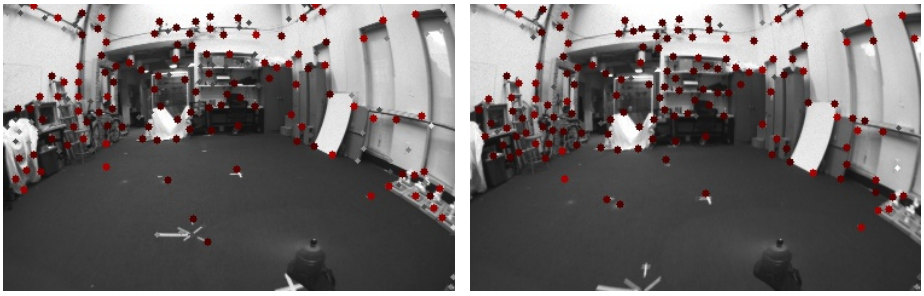
where $\mathbf{y}$ is a collection of desired derivative values at each waypoint that can be either free or fixed. We fix the position, velocity and acceleration at the first waypoint to be current state of the vehicle in order to maintain smooth trajectories during replanning and loop closures. The velocity and acceleration are set to be zero for the last waypoint. For all other waypoints, only position is fixed and the trajectory generator will provide the velocity and acceleration profile. The coefficients of the polynomial trajectories $\mathbf{s}$ can be found via a linear mapping $\mathbf{s} = M\mathbf{y}$.

A limitation of the above trajectory generation approach is the necessity of predefining the travel time between waypoints. Due to computational constraints, we do not perform any iterative time optimization [69, 83] to find the optimal segment time, but rather use a heuristic that approximates the segment time as a linear trajectory that always accelerates from and decelerates to zero speed with a constant acceleration at the beginning and end of a segment, and maintains constant velocity in the middle of a segment. This simple heuristic can help avoid excessive accelerations during short segments, and is a reasonable time approximation for long segments.

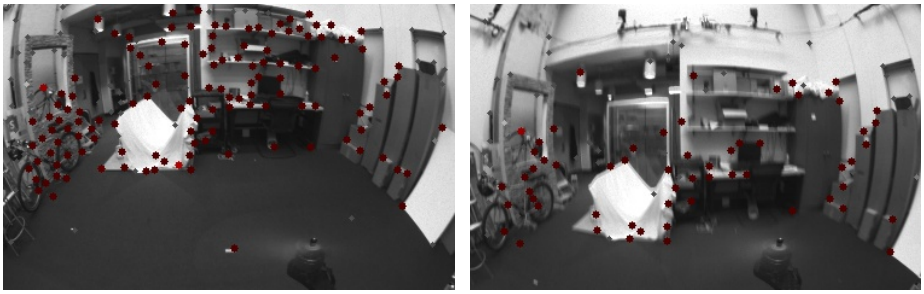
Fig. 7.3 shows in simulation a quadrotor tracking a smooth trajectory generated from a sequence of waypoints. A change of waypoints and trajectory regeneration take place at $20\,\mathrm{sec}$. The regenerated trajectory smoothly connects to the initial trajectory and the quadrotor is able to smoothly switch waypoints.

7.4 Experimental Results

We now present a set of experiment conducted inside a lab space with ground truth motion capture system to verify the effectiveness of the integrated high level planning and trajectory (re)generation. In this experiment, the platform setup is identical to the one

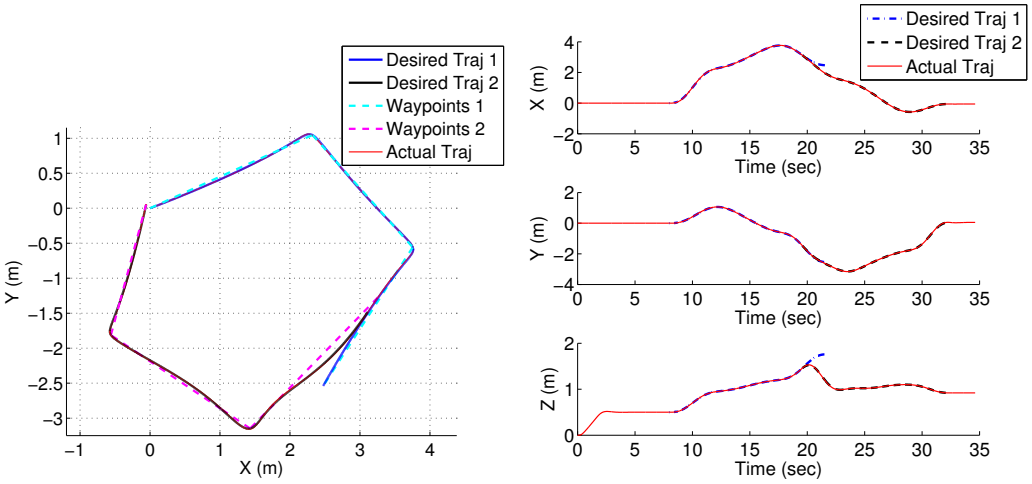


(a) Before translation (b) After translation



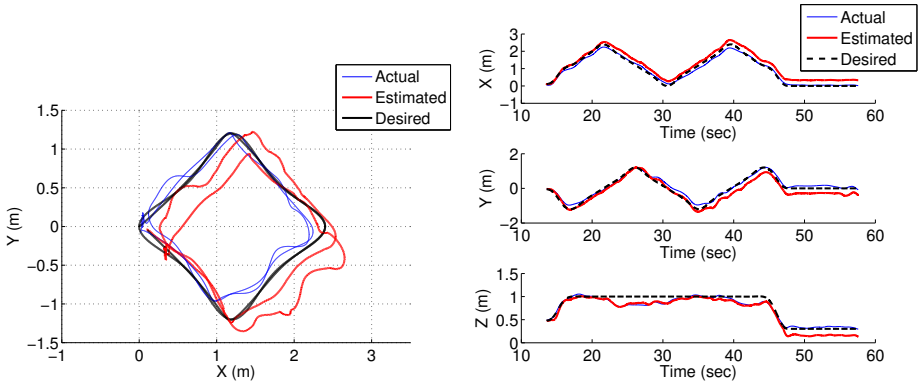
(c) Before rotation (d) After rotation

Figure 7.2: Effects on feature tracking performance due to fast translation (Figs. 7.2(a)–7.2(b)) and fast rotation (Figs. 7.2(c)–7.2(d)). The number of tracked features significantly decreases after rotation.



(a) Trajectory (b) Position

Figure 7.3: A simulated quadrotor tracks a smooth trajectory generated from a sequence of waypoints. Trajectory regeneration takes place after a change of waypoints at 20 sec.



(a) Desired, estimated, and actual trajectories

(b) Desired, estimated, and actual position

Figure 7.4: Autonomous tracking of smooth trajectories generated from a rectangle pattern.

used in Ch. 4. We have a vision-based state estimator and a UKF to produce 100 Hz slow drifting pose estimates that are sufficient to stabilize the MAV. In addition, a visual SLAM (Sect. 4.3.3) detects visual loop closures and produce globally consistent pose estimates. This causes discrepancy between estimator and SLAM frames, which validate the necessity of the proposed planning approach.

In this experiment, the MAV autonomously follows smooth trajectories generated from a rectangle pattern at approximately 1 m/s. The ground truth is used to quantify the global tracking performance. As seen from Fig. 7.4(a) and Fig. 7.4(b), there is slow position drift in vision-based state estimates. However, global corrections from the visual SLAM enables globally consistent operation. Note that the MAV is controlled using onboard state estimates, although global loop closure is clearly incorporated into the system. Due to corrections from the visual SLAM, the desired smooth trajectory in the estimator frame regenerates and changes over time. It can be seen from Fig. 7.4(b) that the actual position of the MAV converges to the desired position, with a standard deviation of $\{\sigma_x, \sigma_y, \sigma_z\} = \{0.1108, 0.1186, 0.0834\}$ (m), indicating globally consistent tracking.

Chapter 8

Autonomous Three-Dimensional Environment Coverage

We utilize our state estimation and mapping approaches introduced in previous chapters (Ch. 3, 4, 5, 6), and combine with planning and control methodologies (Ch. 7) to form a navigation that is able to guide the MAV autonomously fly to a predefined goal in an incrementally mapped environment while avoiding collisions. However, there are still questions of how this goal should be defined. Intuitively, we can have the human operator to select goals in the current map. However, there are missions that we would like to have the human operator completely out of the loop. One example is autonomous environment coverage, which is also called exploration in robotics. In such missions, we would like to have the vehicle start without any prior knowledge of the environment, and return the complete map without any human interaction.

In this chapter, we study a core component in the exploration mission, which is a way to incrementally identify regions or goals that, if visited by the vehicle, will eventually lead to full coverage of the environment. Specifically, we consider the case of exploration of 3D environments consists of multiple floors with a MAV. Central to this methodology is a stochastic differential equation-based exploration algorithm to enable exploration in three-dimensional indoor environments. We are able to address computation, memory,

and sensor limitations by using a map representation which is dense for the known occupied space but sparse for the free space. We determine regions for further exploration based on the evolution of a stochastic differential equation that simulates the expansion of a system of particles with Newtonian dynamics. The regions of most significant particle expansion correlate to unexplored space. After identifying and processing these regions, the autonomous MAV navigates to these locations to enable fully autonomous exploration.

We begin by motivate and detail the key concepts of our approach (Sect. 8.1). The methodology and algorithm details are provided in Sect. 8.3. We present numerical simulations that compare our method to a baseline frontier-based exploration approach in Sect. 8.5.1. We then discuss simulation and experimental results obtained from our autonomous MAV exploring complex indoor environments in Sects. 8.5.2-8.5.3.

8.1 Motivation

Consider a robot starting in a completely unknown environment. As the robot acquires sensor information, it is able to build a map representing the unoccupied and occupied spaces of the environment. Concurrently, we identify those sensed regions as known and therefore explored. The goal of this work is to identify regions that are presently unknown and guide the robot to those unknown locations, in the process exploring and expanding the map representation of the environment.

The development of our approach results from the evaluation of existing methods, such as frontier-based exploration [116] with naive extensions to three dimensions. These methods often fail to accurately capture the differences between unoccupied and unknown space because sensors provide incomplete information about the surrounding three-dimensional environment. We frequently observed cases where unknown and unoccupied regions of space were nearly co-located and sparsely-filled regions of the environment that should clearly be identified as unoccupied. Frontier-based exploration per-

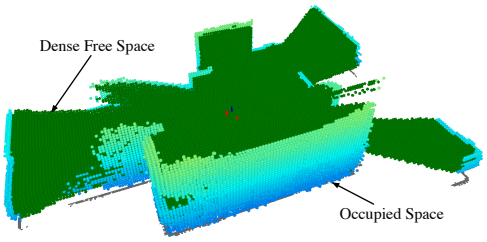
forms poorly in this case as it relies on the boundary between unoccupied and unknown space to determine the next exploratory step of the vehicle (Fig. 8.1), which results in a myopic strategy that increases local coverage while reducing the rate of expansion of the map. More recent methods such as those proposed in [18, 91] proved to be too computationally expensive for our system to enable real-time operation. Additionally, while in two dimensions, maintaining a dense map that contains a representation of known and unknown regions is computationally tractable, attempting the same in three dimensions quickly becomes intractable for systems with limited memory and computational capabilities because of the amount of sensor information.

Therefore, we pursue an approach that does not require a dense representation of the free space. In doing so, we address some of the issues resulting from incomplete sensor information. Fundamental to our approach is the observation (and assumption) that unstructured or uncluttered regions of the map generally correlate with unexplored regions of the indoor environment. Therefore, we wish to identify these regions as locations for further exploration. Following the literature, we call these regions *frontiers* as they serve to differentiate between the known and unknown regions of the environment.

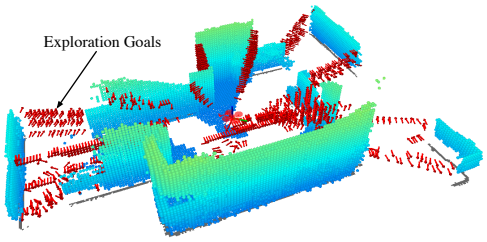
8.2 Overview

8.2.1 Notes on Notation

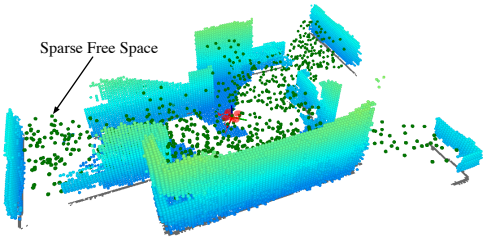
This chapter considers a relatively isolated topic comparing to the rest of this thesis (Ch. 3- 6). Therefore, it is appropriate to reuse some of the notations in this chapter to simplify presentation. While reading this chapter, the readers should stay with only the notations defined within it.



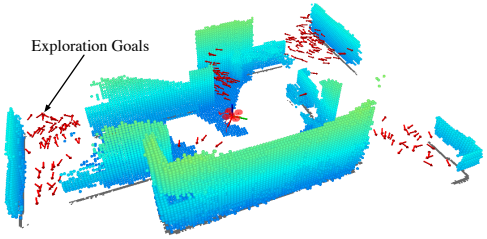
(a) Dense free space



(b) Goals found by traditional frontier-based approach



(c) Sparse free space



(d) Goals found by the SDEE algorithm

Figure 8.1: Challenges with traditional frontier-based exploration methods in 3D. Figure 8.1(a) depicts experimental data showing the observed free and occupied space given a dense voxel grid representation. Frontier-based exploration methods look toward the boundary between free and unexplored space. However, in three dimensions, free space observations are often incomplete due to occlusions and limitations on sensor field-of-view and resolution, yielding frontiers that result in poor exploration performance (Fig. 8.1(b)). The frontiers resulting from the proposed method are shown in Fig. 8.1(d).

8.2.2 Approach

The exploration algorithm begins when the robot starts sensing the environment and building a map of the occupied space. The free space, which is identified from sensor observations, is represented by an appropriately sampled set of virtual particles (Sect. 8.3.1). The particle set is resampled based on the local particle density and the current particle set to ensure an accurate representation of the environment free space (Sect. 8.3.2). These virtual particles are subject to simulated disturbance forces that are stochastic in nature, forcing them to disperse through the known and unknown space with the motion given by Newtonian dynamics. The evolution of these particles is given by a stochastic differential equation (SDE) that considers collisions with the known occupied space defined by the current map (Sect. 8.3.3). The simulation of the dynamics of the system of particles and the expansion of the boundary of the set of particles allow us to identify exploration frontiers (and therefore goals) based on the particle dispersion (Sect. 8.3.4). The frontier goals are queued by the robot and removed from the queue as the robot visits the goals (Sect. 8.3.5). The MAV navigates to these locations while incorporating sensor information into the map and defining new particles based on the sensor observations of the free space. The exploration algorithm terminates when all navigation goals are visited as this corresponds to the full exploration of the environment. A graphical representation of the approach is depicted in Figs. 8.2-8.3

We refer to the process of initializing, resampling, simulating particle expansion, and extracting frontiers as the Stochastic Differential Equation-based Exploration algorithm (SDEE). The SDEE algorithm is run repeatedly as required for the duration of the experiment or until the environment is completely explored, where a new iteration of the algorithm is triggered when the robot successfully visits a navigation goal. Each step of the algorithm is described in detail in the remainder of this section and summarized in Algorithm 2.

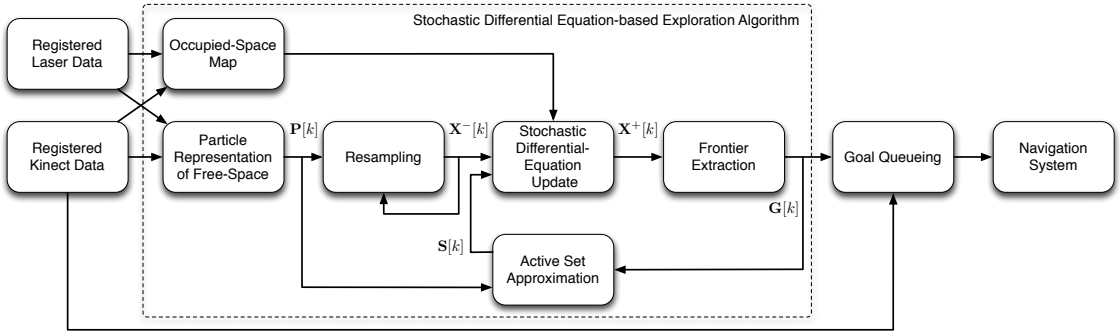
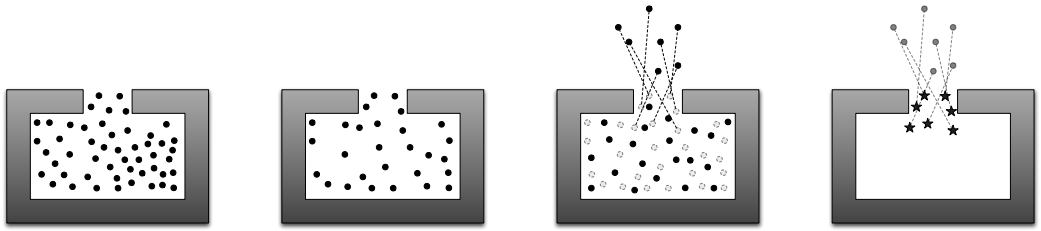


Figure 8.2: The autonomous exploration system design. Registered sensor data is used to populate an occupied voxel cell map and generate a sparse free space representation defined by particles $\mathbf{P}[k]$. Sensor observations in the form of these particles accumulate in the set, $\mathbf{P}[k]$, until the k^{th} iteration of the algorithm is triggered (Sect. 8.3.1). The particle set is resampled (Sect. 8.3.2), yielding a set, $\mathbf{X}^-[k]$, with $N_{\text{min}}[k]$ particles. The particles $\mathbf{X}^-[k]$ undergo a stochastic differential equation simulation (Sect. 8.3.3) to generate the set $\mathbf{X}^+[k]$. Exploration frontiers are extracted based on the transformation of $\mathbf{X}^-[k]$ to $\mathbf{X}^+[k]$ and define the set of goals $\mathbf{G}[k]$ (Sect. 8.3.4). These goals are queued, with the current goal selected based on the current map representation and the state of the robot (Sect. 8.3.5). Goals are also cleared from the queue as they are observed by the Kinect sensor. An approximation to reduce the computational cost of integration of the SDE is defined by considering the most recent or highly weighted particles (termed the *active set*, Sect. 8.3.4), which are defined by the most recent sensor observations, $\mathbf{P}[k]$, and the previous set of frontier goals, $\mathbf{G}[k-1]$.



(a) $\mathbf{X}[k]$ (b) $\mathbf{X}^-[k]$ (c) $\mathbf{X}^+[k]$ (d) $\mathbf{G}[k]$

Figure 8.3: Graphical illustration of the notation at the k^{th} iteration including: the sparse free space representation $\mathbf{X}[k]$ (Fig. 8.3(a)), the resampled particle set $\mathbf{X}^-[k]$ (Fig. 8.3(b)), the particle set after the SDE simulation $\mathbf{X}^+[k]$ (prior $\mathbf{X}^-[k]$ also shown in gray, Fig. 8.3(c)), and the resulting frontier goals $\mathbf{G}[k]$ (shown as black stars with associated particles in $\mathbf{X}^+[k]$ in gray, Fig. 8.3(d)).

8.2.3 Assumptions

We begin by assuming that the MAV is able to autonomously navigate. Hence, this work builds upon systems developed in previous chapters, in which we describe the design of a fully autonomous MAV system able to localize, map, plan, and control onboard the vehicle without requiring external infrastructure or human interaction beyond high-level navigation goals. The output of the SDEE algorithm is a sequence of goal locations in the current map and the input for the navigation system of our MAV as shown in Fig. 8.2.

We assume that the characteristic sensor length, or maximum observable distance of the least capable sensor, is known in advance, and denote this value as D. We set the maximum observable distance of the laser scanner and RGB-D camera sensor as $30\,\text{m}$ and $4\,\text{m}$, respectively. Hence, $D = 4\,\text{m}$ for the simulation and experimental results.

8.3 The SDEE Algorithm

8.3.1 Particle-based Representation of Free Space

Our approach requires a sparsely sampled representation of the free space. As the robot observes and maps the environment, it generates or emits particles at known free space locations according to the sensors. Therefore, as the robot is equipped with a laser scanner and RGB-D camera sensor, we may readily define free space locations by randomly sampling over the space known to be unoccupied – the space between the robot and the observed occupied space.

Define a set of particles, $\mathbf{P}[k]$, where each particle, $p_i[k]$, represents a point mass in free space ($\mathbb{R}^2$ or $\mathbb{R}^3$ for this work) between iterations $k-1$ and k of the SDEE algorithm. As the robot observes the environment, the number of particles increases as the particles accumulate in the set $\mathbf{P}[k]$. During the experiment, we limit the rate of particle emission by only creating new particles after a small translation or rotation. This heuristic is applied to prevent the unnecessary creation of particles, for example, while

the robot hovers in place without gathering any new information. After each iteration of the SDEE algorithm $\mathbf{P}[k]$ is reset to the empty set. We also define a desired initial hover pose directly above the starting position of the vehicle, denoted as q_h. The initial set $\mathbf{P}[0]$ is populated via sensor information gathered as the robot navigates to q_h (see Algorithm 2).

8.3.2 Resampling

As we are continually creating particles to represent the free space, we wish to maintain some bound on the number of particles while ensuring that the particles are evenly distributed over the unoccupied space and accurately reflect the free space in the environment. We wish to keep this number as small as possible. Accordingly, we limit the number of particles to N_{min}, a value that is calculated based on the state of the map and an estimate of the volume of free space.

Consider the k^{th} iteration of the SDEE algorithm with newly-created free space particles, $\mathbf{P}[k]$, and the resampled particles from the previous iteration, $\mathbf{X}^{-}[k-1]$. Denote the union of these two sets as:

$$\mathbf{X}[k] = \mathbf{X}^{-}[k-1] \cup \mathbf{P}[k].$$

Let the number of particles in $\mathbf{X}[k]$ be:

$$N[k] = |\mathbf{X}^{-}[k-1]| + |\mathbf{P}[k]|.$$

Define the mean distance between particles $x_i[k] \in \mathbf{X}[k]$ as:

$$d[k] = \frac{1}{N[k]} \sum_{i=1}^{N[k]} d_i[k]$$

with

$$d_i[k] = \frac{1}{k_i} \sum_{j \in K_i} \|x_j[k] - x_i[k]\| \tag{8.1}$$

and K_i denoting the neighboring set of the nearest k_i particles (i.e. $|K_i| = k_i$).

We want the average distance between particles to be considerably smaller than D, the characteristic sensing length of the robot. Therefore, we define the maximum permissible separation distance between particles as $d_{\max} = \beta D$ with $\beta \ll 1$. Now we can calculate $N_{\min}[k]$, the minimum number of particles required for the k^{th} iteration, so that we ensure a maximum separation distance which is much smaller than the characteristic sensing range. Specifically, we determine the rate at which $N_{\min}[k]$ be changed by the average distance between the particles and the characteristic length scale derived from the sensor field-of-view according to:

$$\frac{N_{\min}[k]}{N_{\min}[k-1]} = \frac{d[k-1]^3}{d_{\max}^3} = \frac{d[k-1]^3}{(\beta D)^3}.$$

As the robot explores the environment, the mean distance between particles will increase, thus requiring $N_{\min}[k]$ to also increase. However, if the mean distance between particles does not increase, corresponding to an instance where the environment representation does not increase in size, the minimum number of required particles will remain constant.

Associate with each particle $x_i[k] \in \mathbf{X}[k]$ the weight $w_i[k]$ using the metric:

$$w_i[k] = \frac{d_i[k]^3}{\sum_{j=1}^{N[k]} d_j[k]^3}.$$

Note that the particle weight is analogous to the density of a mixture medium with uniform mass per particle but with particles in lower density regions having a higher weight, and thus a higher probability of being sampled. The resampling step consists of drawing $N_{\min}[k]$ particles from $\mathbf{X}[k]$ based on the corresponding particle weights. The resampled set is defined as $\mathbf{X}^-[k]$ with $|\mathbf{X}^-[k]| = N_{\min}[k]$.

In the next subsection we will describe the SDE simulation step. To avoid confusion, we will drop the notation for the k^{th} iteration and let $x_i^- \in \mathbf{X}^-$ denote $x_i^-[k] \in \mathbf{X}^-[k]$.

8.3.3 Particle Dynamics

Consider an enclosed three-dimensional environment that contains a fixed number of gas molecules with dynamics that follow the Langevin equation:

$$m\ddot{x}_i(t) = -\nabla U(x_i(t)) - \gamma \dot{x}_i(t) + \sqrt{2\gamma k_b T}\eta(t) \tag{8.2}$$

with m defining the molecule mass, $x_i(t) \in \mathbb{R}^3$ is the position of the i^{th} molecule at time t, $U(x_i(t))$ is the potential due to interactions between molecules, γ is a damping term, k_b is the Boltzmann constant, T is the temperature, and $\eta(t)$ is a δ-correlated stationary zero-mean Gaussian process:

$$< \eta(t) > = 0$$
$$< \eta(t)\ \eta(t') > = \delta(t - t').$$

We want the molecules to emulate an ideal gas. This means that the molecules do not interact ($U(x_i(t)) = 0$), and the collisions are perfectly elastic and satisfy the assumptions for frictionless impact. The evolution of the state of each molecule, $x_i(t)$, is dictated by (8.2), with initial conditions $x_i(0)$, $\dot{x}_i(0)$, and the environment. At steady state, the pressure of the gas, P, in an environment volume, V, is given by the ideal gas law:

$$P = \mu \frac{N}{V} \tag{8.3}$$

where $\mu = k_b T$ can be considered to be a constant.

As the particles disperse through the environment according to the motion equations in (8.2), the volume increases and according to (8.3), the pressure and the density of particles, $\rho = \frac{N}{V}$, decrease. Both of these quantities, P and ρ, can be estimated locally based on the distances between particles. The lowest density regions, which are areas of maximum expansion, correspond to unexplored regions and can be used to guide the identification of frontiers as discussed in Sect. 8.3.4.

Time and Length Scales

From (8.2), the time constant associated with the deterministic, macro-scale Newtonian motion of a particle is given by:

$$\tau = \frac{m}{\gamma}.$$

A choice of the time scale and the initial velocity of a particle leads naturally to a length scale. By solving the differential equation we get an expression of the distance travelled

by the particle:

$$x_i(t) - x_i(0) = \tau\dot{x}(0)(1 - e^{-\frac{t}{\tau}}).$$

Thus the length scale, L, associated with the time scale τ is given by:

$$L = \tau\dot{x}_i(0)(1 - e^{-1}). \tag{8.4}$$

It is natural to choose the length scale based on the sensor field-of-view, $L \sim D$. Thus, we can pick either the initial velocities for the particles in the SDE simulation or the time scale τ and determine the other using the equation above.

SDE Integration and Initialization

Each run of the SDE simulation is carried out for a randomly chosen initial velocity $\dot{x}_i(0)$ for the time duration τ. However, the integration time step, Δt, and initial velocity must be chosen to ensure that we detect any particle collisions with the environment. Therefore, we choose Δt and define $\dot{x}_i(0)$ as a function of Δt and the resolution of the occupied space with voxels of dimension $\Delta M \times \Delta M \times \Delta M$. To ensure that we are able to detect any possible collisions of the particles with the environment, we require $\|\dot{x}_i(0)\| = \frac{\Delta M}{\Delta t}$. The magnitude is consistent across all particles and we draw the direction of the initial velocity vector randomly from a uniform distribution in azimuth and elevation angles.

We denote the particle set after the SDE simulation as $\mathbf{X}^+[k]$. We will now consider the transformation from $\mathbf{X}^-[k]$ to $\mathbf{X}^+[k]$ and extract frontier goals for further exploration.

8.3.4 Frontier Extraction

Based on the previous statement that we are interested in detecting regions of greatest expansion, we can consider volumetric or density changes as a function of local changes in $\mathbf{X}^-[k]$ and $\mathbf{X}^+[k]$. For the sake of brevity, we only discuss detection based on volumetric changes. However, changes in local density are readily computed and may be approached in a similar manner.

Algorithm 2 The SDEE Algorithm

$\mathbf{X}^-[0]$, $\mathbf{G}[0] \leftarrow \emptyset$, $q_i \leftarrow q_h$, $\mathbf{Q}[0] \leftarrow \{q_i\}$, $k \leftarrow 0$

while $\mathbf{Q}[k] \neq \emptyset$ **do**

  $\mathbf{P}[k] \leftarrow \emptyset$

  while $q_i \in \mathbf{Q}[k]$ **and** navigating to q_i **do**

    $\mathbf{P}[k] \leftarrow$ Sample free space particles (Sect. 8.3.1)

    if $q_j \in \mathbf{Q}[k]$ is observed by sensor ($\forall q_j \in \mathbf{Q}[k]$) **then**

      $\mathbf{Q}[k] \leftarrow \mathbf{Q}[k] \setminus q_j$

    end if

  end while

  $k \leftarrow k + 1$

  $\mathbf{X}[k] \leftarrow \mathbf{X}^-[k-1] \cup \mathbf{P}[k-1]$

  $\mathbf{X}^-[k] \leftarrow$ Resample($\mathbf{X}[k]$) (Sect. 8.3.2)

  if $\mathbf{G}[k-1] \neq \emptyset$ **then**

    $\mathbf{S}[k] \leftarrow \mathbf{G}[k-1] \cup \mathbf{P}[k-1]$

    $\mathbf{X}^+[k] \leftarrow$ SDE Integration($\mathbf{S}[k]$) (Sect. 8.3.3)

  else

    $\mathbf{X}^+[k] \leftarrow$ SDE Integration($\mathbf{X}^-[k]$)

  end if

  $\mathbf{G}[k] \leftarrow$ Frontier Extraction($\mathbf{X}^-[k]$, $\mathbf{X}^+[k]$) (Sect. 8.3.4)

  $\mathbf{Q}[k] \leftarrow \mathbf{Q}[k-1] \cup \mathbf{G}[k]$

  $q_i \leftarrow$ Choose Next Goal($\mathbf{Q}[k]$) (Sect. 8.3.5)

end while

We are interested in the particles (only for the k^{th} iteration), $x_i^+ \in \mathbf{X}^+$, that are spatially separated from the particles, $x_i^- \in \mathbf{X}^-$. We draw from the set $\mathbf{X}^-$ the closest neighbor to the particle x_i^+:

$$\hat{d}_i = \underset{x_j^- \in \mathbf{X}^-}{\arg\min} \|x_i^+ - x_j^-\|$$

by constructing a KD-Tree based on $\mathbf{X}^-$ and compute $\hat{d}_i$ for all $x_i^+ \in \mathbf{X}^+$. We choose the largest distances through thresholding:

$$\hat{d}_i > \alpha D$$

where $\alpha \in (0, 1]$ is a scalar. Decreasing α will yield more possible frontier locations but at the cost of increasing the number of false positive solutions – frontier goals that do not lie in the unexplored space.

The actual goal for autonomous navigation should be in the known free space to ensure that the robot is able to continue to localize itself. Therefore, we define the point g_i as the position of the i^{th} particle immediately following its last reflection (assuming an obstacle collision occurs). The orientation of the goal is defined as the orientation of the velocity of the i^{th} particle at g_i: $\frac{\dot{g}_i}{\|\dot{g}_i\|}$. Hence, the exploration goal will always have direct line-of-sight sensing of both explored and unexplored space. These goals define the set $\mathbf{G}[k]$ at the k^{th} iteration with associated position and orientation.

Improving Performance via the Active Set

In general, the number of particles that actively impact the outcome of the frontier extraction are considerably fewer than the number of particles in the set $\mathbf{X}^-[k]$. Define the set of particles, $\mathbf{S}[k]$, as containing those particles with the highest weight, $\mathbf{G}[k-1]$, from the previous SDE simulation and all newly created particles since the last algorithm update, $\mathbf{P}[k]$. We refer to $\mathbf{S}[k]$ as the active set of particles: those particles most influential in the update of the exploration algorithm. At the k^{th} iteration, rather than apply the SDE step to the particles in $\mathbf{X}^-$, we consider only the particles $s_i \in \mathbf{S}$ with $|\mathbf{S}| = N_{\mathbf{S}}$. As we

are guaranteed that $N_{\mathbf{S}} \leq N_{\text{min}}$ (and generally $N_{\mathbf{S}} \ll N_{\text{min}}$), we reduce the computation requirements of the SDE integration by considering fewer particles in the calculation. After integration, for each particle $s_i^+ \in \mathbf{S}^+$ we compute:

$$\hat{d}_i = \underset{x_j^- \in \mathbf{X}^-}{\arg\min} \|s_i^+ - x_j^-\|$$

and apply the same thresholding approach as before. Note that we are computing the change with respect to $\mathbf{X}^-$. During operation, it is possible that this heuristic fails (i.e., no viable goals are computed via thresholding). In this case, we re-run the frontier detection approach based on the full particle set.

8.3.5 Goal Queuing and Algorithm Termination

The final step in the exploration strategy is the queuing of frontiers goals. Define the goal queue:

$$\mathbf{Q}[k] = \mathbf{Q}[k-1] \cup \mathbf{G}[k]$$

starting with $\mathbf{Q}[0] = \emptyset$. We choose from $\mathbf{Q}[k]$ a goal q_i (position and orientation) to send to the autonomous navigation system. The goal is selected to minimize the motion of the vehicle (accounting for motion around obstacles or floor transitions) but could be based on other criteria. We also clear a goal from the queue when the goal is observable by the robot and within a distance D from the robot's state. The event of reaching q_i triggers a new iteration of the SDEE algorithm. The exploration strategy terminates when $\mathbf{Q}[k] = \emptyset$.

8.3.6 Heuristics for Improved Performance

We employ some heuristics to improve upon the performance of the algorithm.

- The exploration rate increases if we bias the sampling of initial velocities of the particles in the SDE simulation to favor x-y motion over z motion. This heuristic

Active Set Heuristic	Particle Resampling	SDE Simulation	Frontier Identification
No	$\boldsymbol{O}(N \log N)$	$\boldsymbol{O}(TN_{\text{min}})$	$\boldsymbol{O}(N_{\text{min}} \log N_{\text{min}})$
Yes	$\boldsymbol{O}(N \log N)$	$\boldsymbol{O}(TN_{\mathbf{S}})$	$\boldsymbol{O}(N_{\mathbf{S}} \log N_{\text{min}})$

Table 8.1: Complexity of the k^{th} iteration of the SDEE algorithm.

is effective because buildings tend to be larger in length and width as compared to height.

- We do not want the robot to fly in directions corresponding to blind spots. Thus the planner finds trajectories whose tangents are limited to a maximum vertical slope of half the field-of-view of the RGB-D sensor with the robot facing the direction of forward motion.

- A goal, $g_i[k]$, is discarded if the line segment connecting $g_i[k]$ and its associated particle in $\mathbf{X}^+[k]$ goes through a small opening in the environment that is not traversable by the actual robot.

8.4 Complexity

As the number of particles varies between iterations, the complexity also varies between iterations. The SDEE algorithm consists of three computational steps: resampling, simulation of the SDE, and selection of the frontiers. The complexity of resampling is $\boldsymbol{O}(N[k] \log N[k])$. The complexity of the SDE simulation is $\boldsymbol{O}(TN_{\text{min}}[k])$, where $T = \tau/\Delta t$ is the total number of update steps. The complexity of frontier identification is $\boldsymbol{O}(N_{\text{min}}[k] \log N_{\text{min}}[k])$. However, if the active set heuristic is applied, the complexity of the SDE simulation and frontier identification is reduced to $\boldsymbol{O}(TN_{\mathbf{S}}[k])$ and $\boldsymbol{O}(N_{\mathbf{S}}[k] \log N_{\text{min}}[k])$, respectively. A summary of the algorithm complexity is presented

in Table 8.1.

Available onboard memory is most greatly impacted by the occupied and free space storage. For a map of size $M \times M \times M$, the occupied-space requires $\boldsymbol{O}(M^3)$ storage space. In practice, we reduce this amount by using our compact environment representation of the occupied space (Sect. 3.3.1) such that the required memory becomes $\boldsymbol{O}(mM^2)$, where m is a small number compared to M. The free space particle representation conservatively requires $\boldsymbol{O}(N[k])$ space to store. Note that for a voxel grid free space representation the map will require $\boldsymbol{O}(M^3)$. In most practical applications $N[k] \ll M^3$. Therefore, the proposed algorithm can run using much less memory than a dense voxel grid environment representation.

8.5 Results

We present both simulation and experimental results to demonstrate the performance of our proposed approach. We also present the comparison of our approach with traditional frontier-based exploration in 2D simulations.

8.5.1 Comparison to Frontier-based Exploration

In this section, we compare the performance of the SDE-based frontier extraction to traditional frontier-based exploration methods [9]. As noted previously (Sect. 8.1) and depicted in Fig. 8.1, the traditional approach performs poorly in three dimensions. For this reason, we consider the comparison in two dimensions. In Fig. 8.4 we show three different maps of different geometric characteristics, the output of the SDE-based frontier extraction, $\mathbf{G}$, and the output of the traditional frontier-based approach using the parameters from Table 8.2. We see that the SDE-based approach yields results similar to the traditional approach.

As previously detailed, algorithm parameters, specifically α, play an important role

α	β	ΔM (m)	Δt (s)	D (m)
0.6	0.3	0.1	0.01	4.0

Table 8.2: Simulation and experimental results parameters.

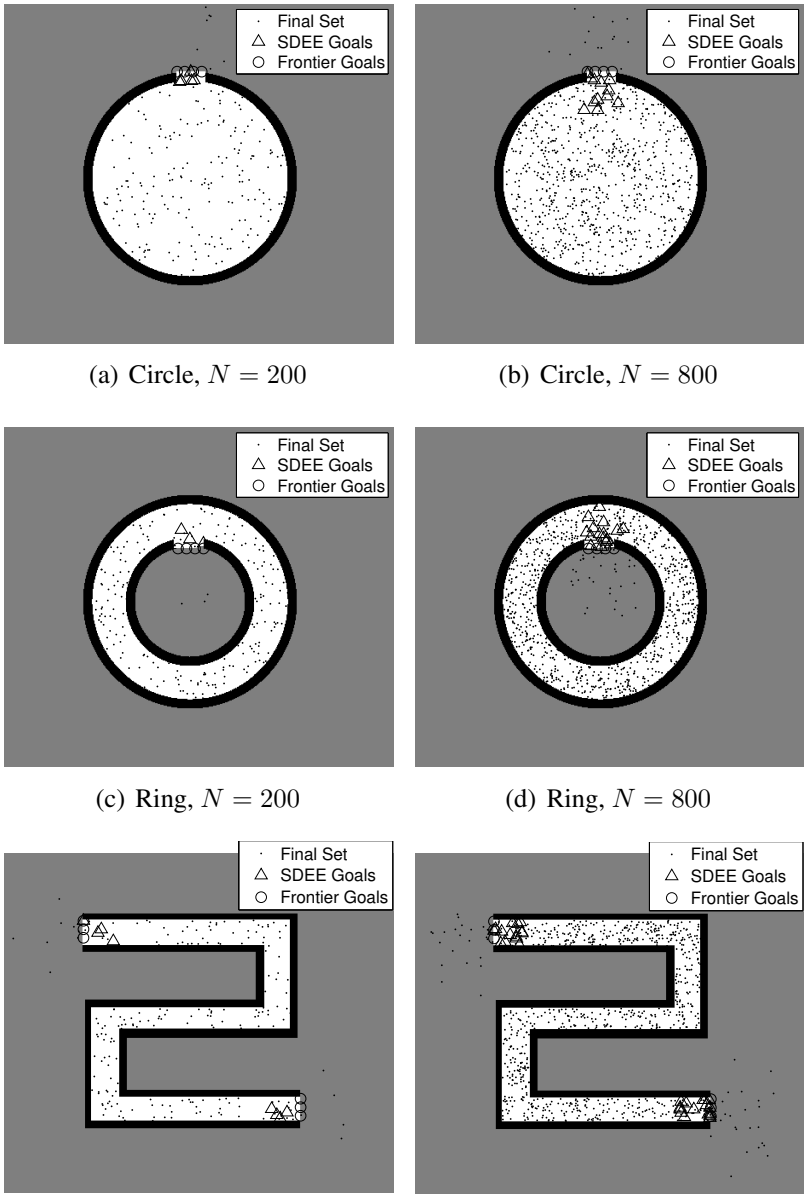
in performance. In Fig. 8.5, we depict the rate of false positive frontiers while varying N and holding $\alpha = 0.6$. The figure depicts the average of 100 simulated trials for each N for each representative map in Fig. 8.4. Note that for $N < 100$, the false positive rate is high, but when $N > 150$, this number drops to zero and remains at this value. This study suggests that the SDEE algorithm compares favorably to frontier-based exploration in effectiveness of identifying regions for further exploration given a relatively sparse representation of the environment free space, supporting the memory efficiency argument made in Sect. 8.4.

In practice, we find that one need only select an appropriate α for a simulated environment of similar scale and the algorithm will work well for a variety of different environments. Hence, α must be reconsidered only when the environment scale and characteristics change considerably.

8.5.2 Simulation Results

We now consider the performance of the SDEE algorithm in three-dimensional environments in simulation. In this section, we are interested in considering the effectiveness of the algorithm in identifying and extracting SDE-based frontiers. We are unable to compare performance to the frontier-based exploration approach leveraged above as it performs poorly in three-dimensional environments as discussed in Sect. 8.1.

We consider two studies to evaluate the performance of the SDEE algorithm. The first study focuses on the ability of the algorithm to identify and extract SDE-based frontiers in a large single floor environment with volume $2650.1\,\mathrm{m}^3$. The second study extends the discussion to a multi-floor environment with volume $765.2\,\mathrm{m}^3$. In both studies, accuracy



(a) Circle, $N = 200$ (b) Circle, $N = 800$

(c) Ring, $N = 200$ (d) Ring, $N = 800$

(e) Tube, $N = 200$ (f) Tube, $N = 800$

Figure 8.4: Numerical simulations of the SDE-based frontier identification approach. The output of the SDEE algorithm is shown for two cases, $N = 200$ and $N = 800$, for three different types of environments. We show the resulting frontier goals (g_i) and results from traditional frontier-based exploration [9].

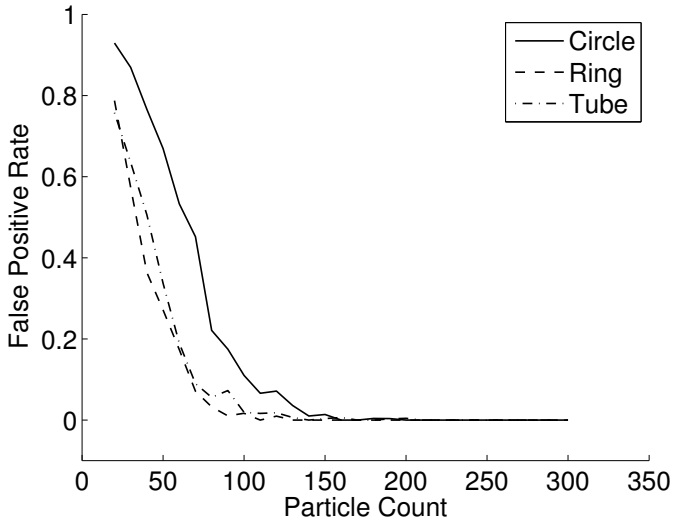


Figure 8.5: The effects of varying the free space particle count while holding $\alpha = 0.6$ for the three environments (Fig. 8.4). The figure depicts the average of 100 simulated trials for each particle count and representative map. The rate of false positives is the percentage of identified particles in $\mathbf{X}^+$ that do not lie in the unexplored space.

is defined as a function of environment coverage and performance is reported based on the total number of required free space particles and the size of the active set.

The simulation software accurately models the vehicle dynamics and sensors, is written in C++, and operates in real-time. The simulated indoor environment is generated from experimental data collected by the aerial vehicle in similar experimental environments to those in Sect. 8.5.3. The simulation of vehicle dynamics matches the experimental vehicle and all onboard estimation and control software are equivalent in the simulation as in the experiments. The simulation of laser and Kinect sensor data is accomplished using ray-tracing methods while IMU sensor data is provided by the dynamic model. The simulated MAV operates with a maximum speed of 1 m/s.

The performance of the single- and multi-floor simulations is shown in Fig. 8.7 with relevant statistics shown in Table. 8.3. The vehicle successfully navigates through the environments with close to complete coverage in comparison to the true environment descriptions (Fig. 8.6). Note that for a few instances in both the single- and multi-floor environments that the active set particle count, $N_{\mathbf{S}}$, and the free space particle count, N,

	Duration (s)	Path Length (m)	Explored (%)
Single floor	2178	1220.2	99.7
Multi-floor	731	463.9	98.6

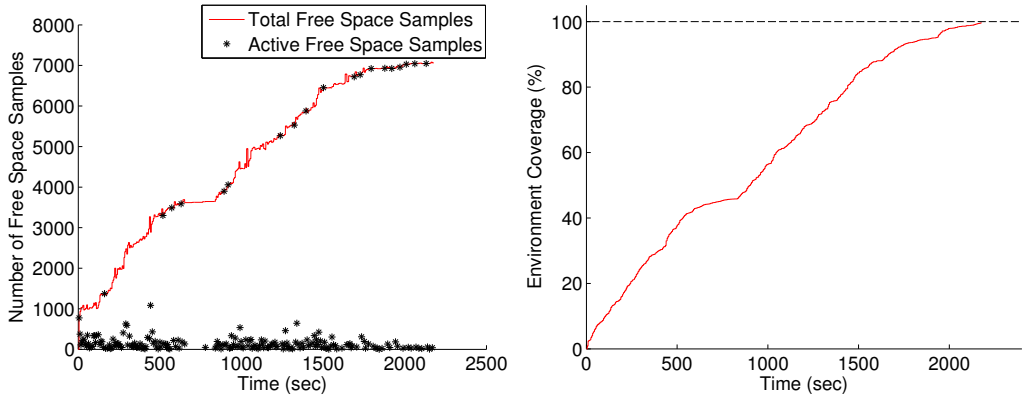
Table 8.3: Simulation performance via duration, path length, and coverage.

are equal, indicating the active set heuristic fails, as discussed in Sect. 8.3.4.

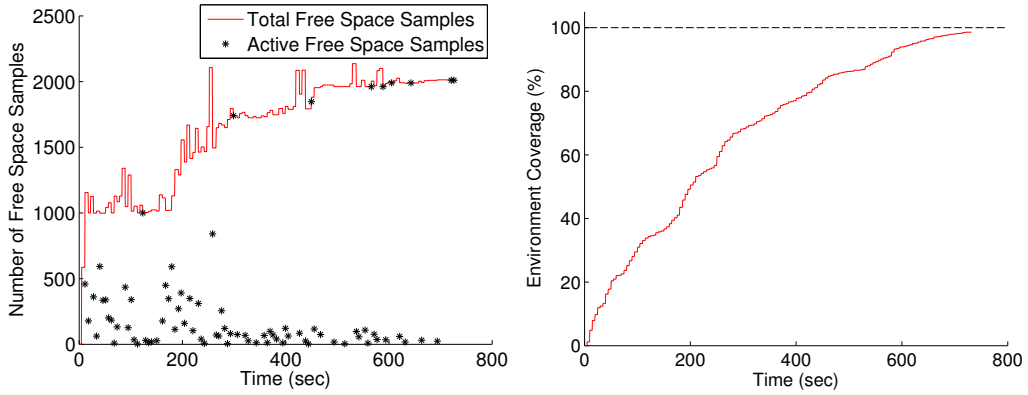
8.5.3 Experimental Results

We present a full 3D exploration in a multi-floor environment to demonstrate the performance of the proposed algorithm. In the experiment, the exploration process is completed when the SDEE algorithm no longer identifies frontiers for further exploration. The configuration of the experimental platform is discussed in Sect. 1.3 and shown in Fig. 1.2(a). Estimation and mapping are powered by approaches in Ch. 3 and Ch. 5. Planning, trajectory generation and control follows the methodology presented in Ch. 7.

The robot operates in an unstructured lobby of a multi-floor building, where there are several vertical spaces for the robot to explore. Figure 8.9 shows the intermediate stages of the exploration process. We can see the goals that lead the robot to first finish the exploration of the first floor, and then try to explore the vertical direction. Despite the fact that the ceiling height exceeds four meters, the proposed algorithm successfully finds exploration goals that guide the robot to sense the high ceiling area, resulting in full coverage of the ceiling and the second floor. The number of free space and active set particles as well as the observed map size are shown in Fig. 8.8.

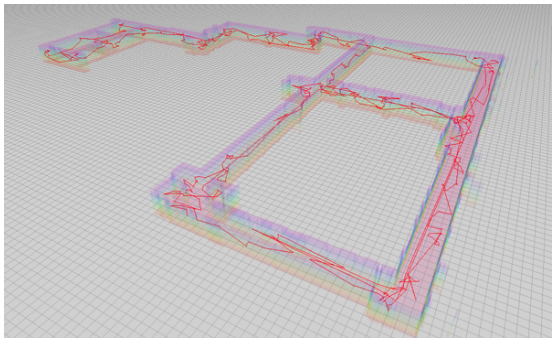


(a) Single floor

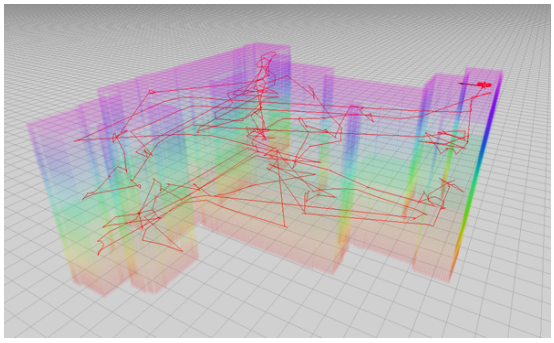


(b) Multi-floor

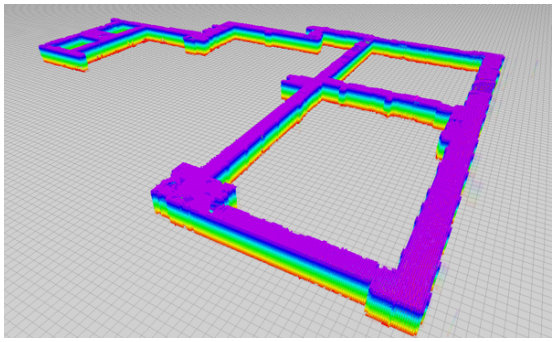
Figure 8.6: Environment coverage and free space particle counts (N and $N_{\mathbf{S}}$) for the exploration simulations in single- and multi-floor environments.



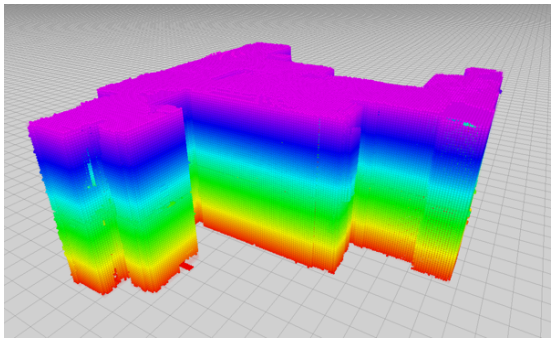
(a) Single floor environment and robot trajectory



(b) Multi-floor environment and robot trajectory



(c) Single floor simulation result



(d) Multi-floor simulation result

Figure 8.7: An aerial robot explores single- and multi-floor environments in simulation (Figs. 8.7(a) and 8.7(b)) with the path shown in red. and the maps resulting from the simulated exploration (Figs. 8.7(c) and 8.7(d)). For both environments, complete coverage is achieved.

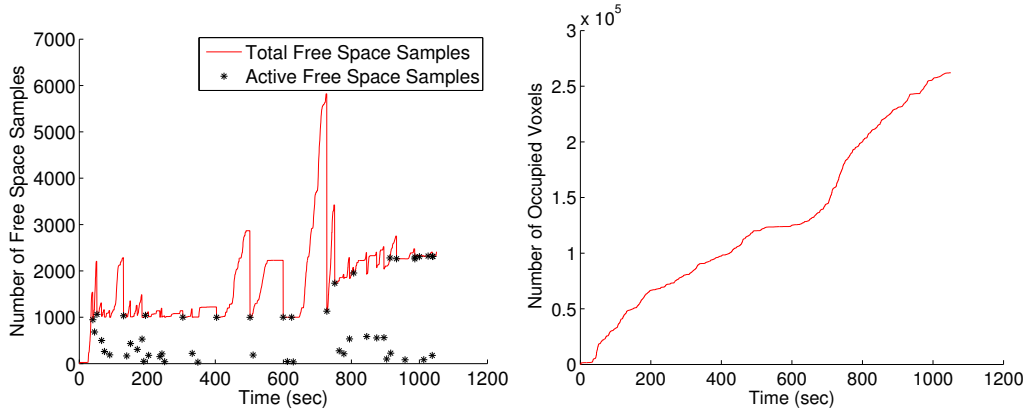


Figure 8.8: Free space particle counts (N and N_S) and total occupied voxel count for the exploration experiment in a multi-floor environment.

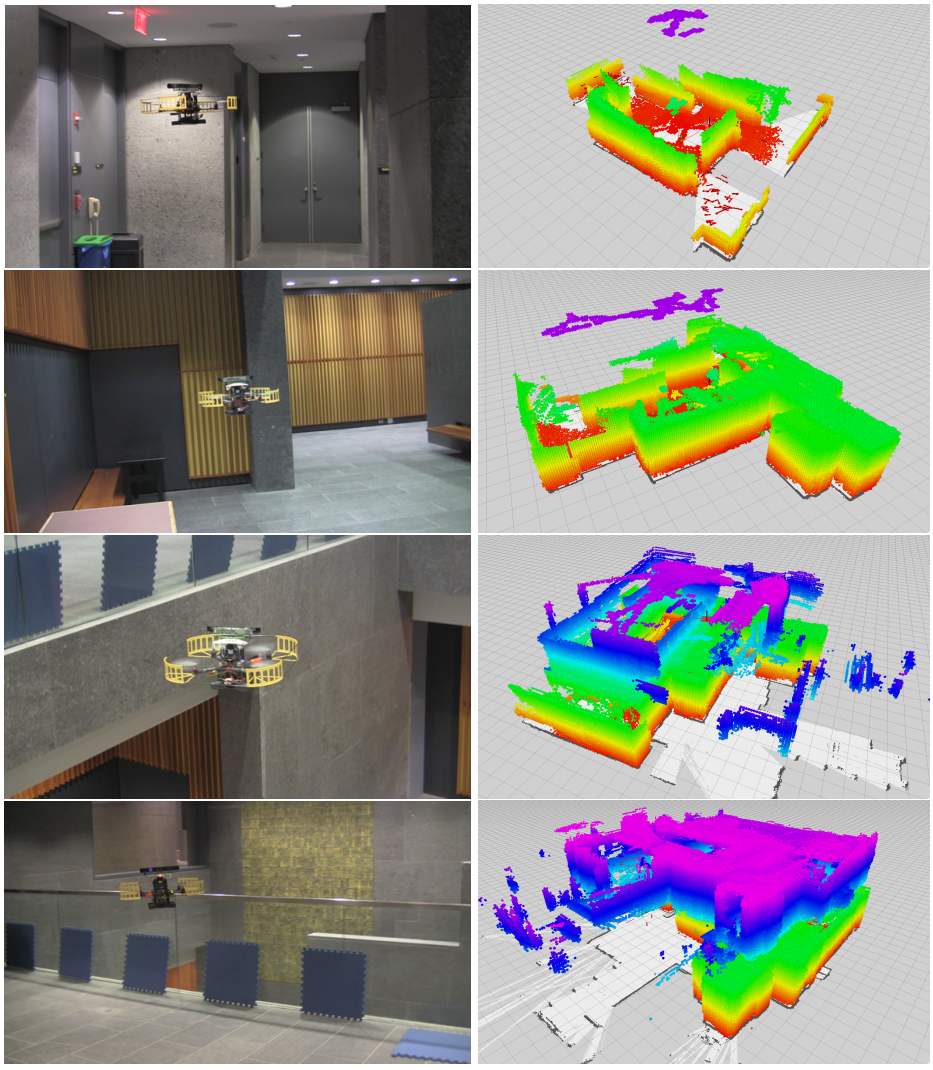


Figure 8.9: Exploration of a multi-floor building. The left column shows views from an external camera to illustrate the structure of the environment. The right column shows the online data visualization.

8.6 Discussion

In this chapter, we propose a stochastic differential equation-based exploration algorithm to enable exploration in three-dimensional indoor environments with a payload constrained MAV. We demonstrate our approach and benchmark our algorithm through numerical simulations and experimental results in single- and multi-floor indoor experiments. This chapter provides the last building block towards fully autonomous MAVs for operations in complex environments without any human interaction. This chapter concludes the development of autonomous MAVs in this thesis.

Chapter 9

Conclusion and Future Work

In this this thesis, we present contributions to the state-of-the-art in autonomous navigation of MAVs in complex indoor and outdoor environments, with a focus on state estimation. In Ch 3 and 4, we develop state estimation and mapping systems that enable off-the-shelf MAVs equipped with laser scanners or cameras to fly autonomous in GPS-denied environments. In Ch 5, we present a modular and extensible methodology that is able to combine information from multiple sensors, including those from previous chapters, in a consistent manner. In Ch 6, we show even with a sensor suite that consists of only one camera and one IMU, it is still possible to recover all metric values of critical navigation states, such as velocity and attitude, without any prior know of the system and the environment. We demonstrate high speed autonomous flight with such minimum sensing capability. We present planning and control approaches that is specifically adapted for autonomous aerial navigation in Ch 7 to form a integrated MAV system. Building on top of all these, in Ch. 8, we present an algorithm that enables autonomous environment coverage without any human interaction.

We note that the although the primary experimental platform used throughout this thesis are quadrotors, our methodologies are not limited to this particular type of platform. In fact, our approaches are sensor-based and work equally well on other vehicles such as fixed and flapping wing MAVs, as well as ground and surface vehicles.

9.1 Summary of Contributions

To summarize, we identify the key contributions of this thesis as follows:

- We develop algorithms and system that enables computation-constrained MAVs to fly autonomously in GPS-denied environments using only lasers or cameras as the primary source of sensing.
- We propose a methodology that improves system reliability via optimally fuse information from heterogeneous sensors.
- We present an algorithm that enables on-the-fly initialization and failure recovery for MAVs equipped with only the monocular visual-inertial sensor suite, and demonstrate high speed autonomous flight with such sensor suite in complex indoor environments.
- We propose a fast online algorithm for generation of informative waypoints that guide towards full coverage of three-dimensional environments.
- We develop fully integrated MAV systems and show through extensive experiments that it is possible to have a MAV equipped with limited onboard sensing and computation to autonomously navigation through complex indoor and outdoor environments. This is also the main goal as well as the expected technical outcome of this thesis.

9.2 Future Work

This thesis opens up several interesting research areas, some are continued development into large-scale systems as the technical capability of MAVs move forward, while others are new directions that may worth pursuing as we evaluate the performance of state-of-the-art approaches.

Intersection of Estimation and Control for High-Speed Navigation

In our current work, planning and control modules are mostly isolated from the estimator. However, as we move towards very high-speed autonomous flight, we may require the estimator and the controller to interact with each other in order to generate and execute efficient, collision-free, and safe flight trajectories.

Dense Vision Systems

Thanks for the rapid development of mobile graphic processors, we believe real-time processing of dense image and pointcloud data is possible. This will lead to significant robustness improvement against motion blur, which is likely to occurred during aggressive maneuvers. Further on this topic, we may investigate the effects of rolling shutters on the performance of dense vision systems, and propose compensation methodologies accordingly.

Novel Sensing Technologies for MAVs

Although conventional sensors such as GPS, cameras, and laser range finders are proven to be useful for MAV navigation, recent advancement in sensing technologies may create new opportunities. For instance, light-field can be an alternative for depth perception, and dynamic vision sensor provides low-latency optical flow-like measurements. There are great potentials in these new sensing technologies for their applications to MAVs.

Sensor versus Observability

It is quite obvious that more sensors lead to better observability of the system, however, how individual sensor affects the observability properties remains unclear. A single camera is able to observe the linear velocities up to a metric scale. With the addition of an IMU and sufficient acceleration excitation, the metric scale becomes observable. However, the definition of sufficient excitation remains unclear. Adding more sensors renders

more quantities observable, but the information that the sensor provides, as well as the quality of the information, result in complex conditions for the observability properties. As such, a generic framework that is able to identify the observability properties in an online fashion will be beneficial for higher level tasks such as motion planning and risk assessment.

Multi-Agent Systems

As the reliability of single MAV systems improves, it becomes feasible to deploy multiple MAVs into more complex missions such collaborative exploration and mapping [57]. It is also interesting to deploy heterogeneous autonomous agents such as ground and aerial vehicles in order to leverage the advantage of different platforms.

Bibliography

[1] Amazon Prime Air. http://www.amazon.com/b?node=8037720011.

[2] DJI Innovations. http://www.dji.com/.

[3] HoneyComb. http://www.honeycombcorp.com/.

[4] A. Bachrach, R. He, and N. Roy. Autonomous flight in unknown indoor environments. *Intl. J. Micro Air Vehicles*, 1(4):217–228, December 2009.

[5] A. Bachrach, S. Prentice, R. He, and N. Roy. RANGE-robust autonomous navigation in gps-denied environments. *J. Field Robotics*, 28(5):644–666, 2011.

[6] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *Proc. of the European Conf. on Computer Vision*, Graz, Austria, May 2006.

[7] C. Bills, J. Chen, and A. Saxena. Autonomous MAV flight in indoor environments using single image perspective cues. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 5776–5783, Shanghai, China, May 2011.

[8] A. Bry, A. Bachrach, and N. Roy. State estimation for aggressive flight in gps-denied environments using onboard sensing. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 1–8, Saint Paul, MN, May 2012.

[9] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Trans. Robot.*, 21(3):376–386, June 2005.

[10] D. Burschka and E. Mair. Direct pose estimation with a monocular camera. In *RobVis*, pages 440–453, 2008.

[11] J. Carlson. *Mapping Large Urban Environments with GPS-Aided SLAM*. PhD thesis, CMU, Pittsburgh, PA, July 2010.

[12] D. Chetverikov. Fast neighborhood search in planar point sets. *Pattern Recognition Letters*, 12(7):409–412, July 1991.

[13] A. Cowley, C. J. Taylor, and B. Southall. Rapid multi-robot exploration with topometric maps. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 1044–1049, Shanghai, China, May 2011.

[14] M. Cummins and P. Newman. Probabilistic appearance based navigation and loop closing. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 2042–2048, Rome, Italy, April 2007.

[15] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *Intl. J. Robot. Research*, 30(9):1100–1123, August 2011.

[16] F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. Robot. Research*, 25(12): 1181–1203, December 2006.

[17] T. Dong-Si and A. I. Mourikis. Estimator initialization in vision-aided inertial navigation with unknown camera-IMU calibration. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 1064–1071, Vilamoura, Algarve, Portugal, October 2012.

[18] C. Dornhege and A. Kleiner. A frontier-void-based approach for autonomous exploration in 3D. In *Proc. of IEEE Intl. Sym. on Safety, Security, and Rescue Robotics*, Kyoto, Japan, November 2011.

[19] I. Dryanovski, W. Morris, and Jizhong Xiao. Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 1553–1559, Taipei, Taiwan, October 2010.

[20] I. Dryanovski, W. Morris, and J. Xiao. An open-source pose estimation system for micro-air vehicles. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 4449–4454, Shanghai, China, May 2011.

[21] C. Estrada, J. Neira, and J. D. Tardos. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Trans. Robot.*, 21(4):588–596, August 2005.

[22] J. A. P. Fentanes, R. F. Alonso, E. Zalama, and J. G. García-Bermejo. A new method for efficient three-dimensional reconstruction of outdoor environments using mobile robots. *J. Field Robot.*, 28:832–853, November 2011.

[23] F.Moreno-Noguer, V.Lepetit, and P.Fua. Accurate non-iterative O(n) solution to the PnP problem. In *Proc. of the IEEE Intl. Conf. on Computer Vision*, Rio de Janeiro, Brazil, October 2007.

[24] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Hong Kong, China, May 2014.

[25] J. Fournier, B. Ricard, and D. Laurendeau. Mapping and exploration of complex environments using persistent 3D model. In *Canadian Conf. on Computer and Robot Vision*, pages 403–410, Montreal, Canada, May 2007.

[26] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schultz, and B. Stewart. Distributed multirobot exploration and mapping. *Proc. of the IEEE*, 94(7):1325–1339, July 2006.

[27] F. Fraundorfer, L. Heng, D. Honegger, G. H Lee, L. Meier, P. Tanskanen, , and M. Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor MAV. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 4557–4564, Vilamoura, Algarve, Portugal, October 2012.

[28] L. Freda, G. Oriolo, and F. Vecchioli. Sensor-based exploration for general robotic systems. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 2157–2164, Nice, France, September 2008.

[29] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 2432–2437, Barcelona, Spain, April 2005.

[30] S. Grzonka, G. Grisetti, and W. Burgard. A fully autonomous indoor quadrotor. *IEEE Trans. Robot.*, PP(99):1–11, 2011.

[31] D. Holz, N. Basilico, F. Amigoni, and S. Behnke. Evaluating the efficiency of frontier-based exploration strategies. In *Proc. of the Intl. Sym. on Robot.*, pages 36–43, Munich, Germany, June 2010.

[32] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Proc. of the Intl. Sym. of Robot. Research*, Flagstaff, AZ, August 2011.

[33] P.J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(2):73–101, 1964.

[34] V. Indelman, S. Williams, M. Kaess, and F. Dellaert. *Robot. and Autom. Syst.*

[35] V. Indelman, A. Melim, and F. Dellaert. Incremental light bundle adjustment for robotics navigation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 1952–1959, Tokyo Japan, November 2013.

[36] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard. Autonomous exploration for 3D map learning. In *Fachgesprach Autonome Mobile Systeme*, Kaiserslautern, Germany, October 2007.

[37] E. S. Jones and S. Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *Intl. J. Robot. Research*, 30(4):407–430, April 2011.

[38] S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In I. Kadar, editor, *Proc. of SPIE*, volume 3068, pages 182–193, July 1997.

[39] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robot.*, 24(6):1365–1378, December 2008.

[40] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Proc. of Robot.: Sci. and Syst.*, Zaragoza, Spain, June 2010.

[41] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *Intl. J. Robot. Research*, 30(1):56–79, January 2011.

[42] F. Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *J. Field Robotics*, 29(2):315–378, 2012.

[43] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 3748–3754, Karlsruhe, Germany, May 2011.

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

[45] L. Kneip, S. Weiss, and R. Siegwart. Deterministic initialization of metric state estimation filters for loosely-coupled monocular vision-inertial systems. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 2235–2241, San Francisco, CA, September 2011.

[46] D. G. Kottas, J. A. Hesch, S. L. Bowman, and S. I. Roumeliotis. On the consistency of vision-aided inertial navigation. In *Proc. of the Intl. Sym. on Exp. Robot.*, Quebec, Canada, June 2012.

[47] D.G. Kottas, K. Wu, and S.I. Roumeliotis. Detecting and dealing with hovering maneuvers in vision-aided inertial navigation systems. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 3172–3179, Tokyo, Japan, November 2013.

[48] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimizations. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 3607–3613, Shanghai, China, May 2011.

[49] A. Kushleyev, B. MacAllister, and M. Likhachev. Planning for landing site selection in the aerial supply delivery. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 1146–1153, San Francisco, CA, September 2011.

[50] S. Lange, N. Sunderhauf, and P. Protzel. Incremental smoothing vs. filtering for sensor fusion on an indoor UAV. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 1773–1778, Karlsruhe, Germany, May 2013.

[51] T. Lee, M. Leoky, and N.H. McClamroch. Geometric tracking control of a quadrotor uav on SE(3). In *Proc. of the Intl. Conf. on Decision and Control*, pages 5420–5425, Atlanta, GA, December 2010.

[52] T. Lefebvre, H. Bruyninckx, and J. De Schuller. Comment on "a new method for the nonlinear transformation of means and covariances in filters and estimators". 47(8):1406–1409, 2002.

[53] J.J Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 1442–1447, Osaka, Japan, November 1991.

[54] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. Keyframe-based visual-inertial SLAM using nonlinear optimization. In *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, June 2013.

[55] M. Li and A.I. Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *Intl. J. Robot. Research*, 32(6):690–711, May 2013.

[56] V. Lippiello and R. Mebarki. Closed-form solution for absolute scale velocity estimation using visual and inertial data with a sliding least-squares estimation. In *Proc. of Mediterranean Conf. on Control and Automation*, pages 1261–1266, Platanias-Chania, Crete, Greece, June 2013.

[57] S. Liu, K. Mohta, S. Shen, and V. Kumar. Towards collaborative mapping and exploration using multiple micro aerial robots. In *Proc. of the Intl. Sym. on Exp. Robot.*, Marrakech, Morocco, 2014. Submitted.

[58] G. Loianno, G. Cross, C. Qu, Y. Mulgaonkar, J.A. Hesch, and V. Kumar. Smart phones power flying robots. *IEEE Robot. Autom. Mag.*, 2014. Submitted.

[59] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, pages 24–28, Vancouver, Canada, August 1981.

[60] T. Lupton. *Inertial SLAM with Delayed Initialisation*. PhD thesis, Univ. of Sydney, Sydney, Australia, March 2010.

[61] T. Lupton and S. Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robot.*, 28 (1):61–76, February 2012.

[62] 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 *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Tokyo, Japan, November 2013.

[63] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte. An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 534–539, EPFL, Switzerland, September 2002.

[64] A. Marjovi, J. G. Nunes, L. Marques, and A. Almeida. Multi-robot exploration and fire searching. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 1929–1934, St. Louis, MO, October 2009.

[65] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Proc. of the IEEE Intl. Conf. on Pattern Recognition*, pages 1–8, Minneapolis, MN, 2007.

[66] A. Martinelli. Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. 28(1):44–60, February 2012.

[67] A. Martinelli. Closed-form solution of visual-inertial structure from motion. *IEEE Trans. Robot.*, 0(0):0, August 2013.

[68] Y. Mei, Y. Lu, C. S. G. Lee, and Y. C. Hu. Energy-efficient mobile robot exploration. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 505–511, Orlando, FL, May 2006.

[69] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 2520–2525, Shanghai, China, May 2011.

[70] R. V. D. Merwe, E. A. Wan, and S. I. Julier. Sigma-point kalman filters for nonlinear estimation: Applications to integrated navigation. In *Proc. of AIAA Guidance, Navigation, and Controls Conf.*, Providence, RI, August 2004.

[71] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The GRASP multiple micro UAV testbed. *IEEE Robot. Autom. Mag.*, 17(3):56–65, September 2010.

[72] N. Michael, S. Shen, K. Mohta, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. In *Proc. of the Intl. Conf. on Field and Service Robot.*, Miyagi, Japan, 2012.

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

[74] A. Mobarhani, S. Nazari, A. H. Tamjidi, and H. D. Taghirad. Histogram based frontier exploration. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 1128–1133, San Francisco, CA, September 2011.

[75] D. C. Moore, A. S. Huang, M. Walter, and E. Olson. Simultaneous local and global state estimation for robotic navigation. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 3794–3799, Kobe, Japan, May 2009.

[76] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 3565–3572, Roma, Italy, April 2007.

[77] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. of the IEEE Intl. Conf. Comput. Vis.*, pages 2320–2327, Barcelona, Spain, November 2011.

[78] D. Nister. An efficient solution to the five-point relative pose problem. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 195–202, Madison, WI, June 2003.

[79] A. Nuchter, H. Surmann, and J. Hertzberg. Planning robot motion for 3D digitalization of indoor environments. pages 222–227, Coimbra, Portugal, June 2003.

[80] E. B. Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, MIT, Cambridge, MA, June 2008.

[81] E.B. Olson. Real-time correlative scan matching. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 4387–4393, Kobe , Japan, May 2009.

[82] C. Pravitra, G. Chowdhary, and E. Johnson. A compact exploration strategy for indoor flight vehicles. pages 3572–3577, Orlando, FL, December 2011.

[83] C. Richter, A. Bry, and N. Roy. Polynomial trajectory planning for quadrotor flight. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Karlsruhe, Germany, May 2013. To appear.

[84] S. I. Roumeliotis and J. W. Burdick. Stochastic cloning: A generalized framework for processing relative state measurements. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 1788–1795, Washington, DC, May 2002.

[85] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. of Intl. Conf. on 3-D Digital Imaging and Modeling*, pages 145–152, Quebec City, Quebec, May 2001.

[86] D. Scaramuzza, A. Martinelli, and R R. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Proc. of IEEE Intl. Conf. of Vision Systems*, New York, NY, January 2006.

[87] D. Scaramuzza, M.C. Achtelik, L. Doitsidis, F. Fraundorfer, E.B. Kosmatopoulos, A. Martinelli, M.W. Achtelik, M. Chli, S.A. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G.H. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J.C. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss. Vision-controlled micro

flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robot. Autom. Mag.*, 21(3), 2014.

[88] David Schleicher, Luis M. Bergasa, Manuel Ocaa, Rafael Barea, and Elena Lpez. Real-time hierarchical GPS aided visual SLAM on urban environments. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 4381–4386, Kobe, Japan, May 2009.

[89] K Schmid, T. Tomic, F. Ruess, H. Hirschmuller, and M. Suppa. Stereo vision based indoor/outdoor navigation for flying robots. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 3955–3962, Tokyo, Japan, November 2013.

[90] R. Shade and P. Newman. Discovering and mapping complete surfaces with stereo. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 3910–3915, Anchorage, AK, May 2010.

[91] R. Shade and P. Newman. Choosing where to go: Complete 3D exploration with stereo. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 2806–2811, Shanghai, China, May 2011.

[92] S. Shen and N. Michael. State estimation for indoor and outdoor operation with a micro-aerial vehicle. In *Proc. of the Intl. Sym. on Exp. Robot.*, Qubec, Canada, 2012.

[93] S. Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 20–25, Shanghai, China, May 2011.

[94] S. Shen, N. Michael, and V. Kumar. Stochastic differential equation-based exploration algorithm for autonomous indoor 3D exploration with a micro-aerial vehicle. *Intl. J. Robot. Research*, 31(12):1431–1444, 2012.

[95] S. Shen, N. Michael, and V. Kumar. Autonomous indoor 3D exploration with a micro-aerial vehicle. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 9–15, Saint Paul, MN, May 2012.

[96] S. Shen, N. Michael, and V. Kumar. Obtaining liftoff indoors: Autonomous navigation in confined indoor environments. *IEEE Robot. Autom. Mag.*, 20(4):40–48, 2013.

[97] S. Shen, N. Michael, and V. Kumar. Vision-based autonomous navigation in complex environments with a quadrotor. In *Technical Report, University of Pennsylvania*, March 2013.

[98] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-based state estimation for autonomous rotorcraft MAVs in complex environments. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 1758–1764, Karlsruhe, Germany, May 2013.

[99] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, 2013.

[100] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Hong Kong, China, May 2014. To Appear.

[101] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Initialization-free monocular visual-inertial estimation with application to autonomous MAVs. In *Proc. of the Intl. Sym. on Exp. Robot.*, Marrakech, Morocco, 2014. Submitted.

[102] J. Shi and C. Tomasi. Good features to track. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, WA, June 1994.

[103] G. Sibley, L. Matthies, and G. Sukhatme. Sliding window filter with application to planetary landing. *J. Field Robot.*, 27(5):587–608, September 2010.

[104] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, volume 4, page 850, Rayleigh, NC, March 1987.

[105] C. Stachniss, O. M. Mozos, and W. Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Ann. Math. Artif. Intell.*, 52 (2-4):205–227, April 2008.

[106] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale drift-aware large scale monocular SLAM. In *Proc. of Robot.: Sci. and Syst.*, Zaragoza, Spain, June 2010.

[107] I. A. Sucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robot. Autom. Mag.*, 19(4):72–82, December 2012. `http://ompl.kavrakilab.org`.

[108] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 2531–2538, Nice, France, September 2008.

[109] G. D. Tipaldi and K. O. Arras. FLIRT - interest regions for 2D range data. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 3616–3622, Anchorage, AK, May 2010.

[110] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka. Autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE Robot. Autom. Mag.*, 19(3): 46–56, 2012.

[111] L. Torabi and K. Gupta. An autonomous six-DOF eye-in-hand system for in situ 3D object modeling. *Intl. J. Robot. Research*, 31(1):82–100, January 2012.

[112] R. Vincent, D. Fox, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schultz, and B. Stewart. Distributed multirobot exploration, mapping, and task allocation. *Ann. Math. Artif. Intell.*, 52(2-4):229–255, April 2008.

[113] S. Weiss, D. Scaramuzza, and R. Siegwart. Monocular-slam-based navigation for autonomous micro helicopters in GPS-denied environments. *J. Field Robot.*, 28 (6), 2011.

[114] 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 *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 957–964, Saint Paul, MN, May 2012.

[115] S. Weiss, M. Achtelik, S. Lynen, M. Achtelik, L. Kneip, M. Chli, and R. Siegwart. Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *J. Field Robot.*, 30(5):803–831, 2013.

[116] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. of IEEE Sym. on Comput. Intelli. in Robot. and Autom.*, pages 146–151, Monterey, CA, July 1997.