Efficient Estimation for Small Multi-Rotor Air Vehicles Operating in Unknown, Indoor Environments

John Charles Macdonald
Brigham Young University - Provo

Efficient Estimation for Autonomous Multi-Rotor Helicopters Operating in Unknown, Indoor Environments

John C. Macdonald Jr.

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

Brigham Young University

November 2012

ABSTRACT

Efficient Estimation for Autonomous Multi-Rotor Helicopters
Operating in Unknown, Indoor Environments

John C. Macdonald Jr.
Department of Electrical and Computer Engineering
Doctor of Philosophy

In this dissertation we present advances in developing an autonomous air vehicle capable of navigating through unknown, indoor environments. The problem imposes stringent limits on the computational power available onboard the vehicle, but the environment necessitates using 3D sensors such as stereo or RGB-D cameras whose data requires significant processing. We address the problem by proposing and developing key elements of a relative navigation scheme that moves as many processing tasks as possible out of the time-critical functions needed to maintain flight.

We present in Chapter 2 analysis and results for an improved multirotor helicopter state estimator. The filter generates more accurate estimates by using an improved dynamic model for the vehicle and by properly accounting for the correlations that exist in the uncertainty during state propagation. As a result, the filter can rely more heavily on frequent and easy to process measurements from gyroscopes and accelerometers, making it more robust to error in the processing intensive information received from the exteroceptive sensors.

In Chapter 3 we present BERT, a novel approach to map optimization. The goal of map optimization is to produce an accurate global map of the environment by refining the relative pose transformation estimates generated by the real-time navigation system. We develop BERT to jointly optimize the global poses and relative transformations. BERT exploits properties of independence and conditional independence to allow new information to efficiently flow through the network of transformations. We show that BERT achieves the same final solution as a leading iterative optimization algorithm. However, BERT delivers noticeably better intermediate results for the relative transformation estimates. The improved intermediate results, along with more readily available covariance estimates, make BERT especially applicable to our problem where computational resources are limited.

We conclude in Chapter 4 with analysis and results that extend BERT beyond the simple example of Chapter 3. We identify important structure in the network of transformations and address challenges arising in more general map optimization problems. We demonstrate results from several variations of the algorithm and conclude the dissertation with a roadmap for future work.

Keywords: simultaneous localization and mapping, SLAM, quadrotor, indoor flight, GPS denied flight, navigation, state estimation, observability, back-end optimization, BERT

ACKNOWLEDGMENTS

Every good thing I have, do, or am should be credited to my Savior, Jesus Christ.

I am also indebted to my wife, Kristine, for her love and support. I owe gratitude to the professors and students at the BYU MAGICC Lab, and to Dr. Randy Beard in particular, for making me a much better student and researcher than I otherwise would be. Brigham Young University is a great place to learn and grow, and I will always be grateful for the formative years I have spent here.

Table of Contents

List of Figures

Chapter 1

Introduction

As context for our research, we seek to enable an autonomous air vehicle to search a completely unknown, indoor environment. This problem is motivated by several potential applications such as reconnaissance in hostile areas or post-disaster assessment, especially when a highly trained remote operator is not available. Even with a remote pilot available, a vehicle equipped with the technologies needed for autonomous exploration could increase the operator's situational awareness and reduce the operator's work load. Despite these potential benefits, successful attempts to design an autonomous indoor flight system have been recent and limited. As recently as the 2009 IEEE International Conference on Robotics and Automation (ICRA), leading researchers in the community were saying, "Several effective systems for indoor and outdoor navigation of ground vehicles are nowadays available. However, we are not aware of a similar system for [indoor] flying robots." [4]

We can further define the context for our research. By "autonomous" we mean that the vehicle is able to operate without any access to information or input produced on a separate system. This precludes the use of a ground station computer, for example. We expect the vehicle must be able to operate independently without even the assistance of other agents such as a cooperative ground vehicle. By a "completely unknown, indoor environment" we are implying that no structure can be assumed in the environment. This certainly precludes using artificial markers that manipulate the environment to provide the vehicle with navigation cues; GPS or other localization beacons are also considered unavailable.

Several additional consequences follow from the problem context. Because the environment is unknown and indoors, an appropriate vehicle should be able to move in possibly cluttered and confined areas. The vehicle should therefore be limited in size with restrictions on the weight and power budget allocated to payload. To move through confined spaces the

Figure 1.1: Example applications for an autonomous indoor flight vehicle. (Left) An aerial image of the Fukushima Daiichi Nuclear Power Plant before the magnitude 9.0 earthquake and subsequent tsunami on 11 March 2011. After the earthquake the walls and floors in several places were broken up and uneven. (Right) An image of a cave in Afghanistan where the floor is strewn with debris and the walls are uneven and slanted. In both cases, the type of autonomous air vehicle we contemplate might be used despite the unknown, unstructured, and confined nature of the environment.

vehicle must also be sufficiently agile. An agile vehicle in turn requires accurate and frequently updated estimates of its dynamic state. Finally, the unknown and unstructured nature of the 3D environment motivates truly 3D sensors (e.g. stereo or RGB-D cameras) as input for online map estimation, collision avoidance, etc.

The confluence of these consequences leads to our primary engineering tradeoff. The limited weight and power for payload lead directly to limits on the vehicle's onboard computing capabilities. But developing timely state and map estimates from voluminous 3D data can be processing intensive. We must meet high computational demands with limited computing resources.

To address the challenges of the above scenario, we have assumed a few design decisions that scope our contribution toward solving the problem. We have adopted small multi-rotor helicopters as our target platform. These vehicles have been identified by leading researchers as "potential game-changers" in robotics [5]. We have also chosen a graph-based paradigm for simultaneous localization and mapping (SLAM). SLAM is the monicker given to any algorithm for autonomous navigation that improves on raw odometry by estimating a map of the environment while at the same time localizing the vehicle within the map. Graph-based SLAM algorithms have benefited from recent theoretical advances and "currently belong to the state-of-the-art [SLAM] techniques with respect to speed and accuracy." [6]

In summary, we aim to further the development of a small multi-rotor helicopter that can autonomously search, map out, and localize itself within an unknown and unstructured indoor environment in spite of its limited ability to process the necessary data. In the remainder of this chapter we will explain why multi-rotor helicopters and graph-based SLAM are especially well suited to our problem. We conclude Chapter 1 with a list of this dissertation's contributions. In Chapter 2 we explain our contributions toward developing an accurate and efficient observer to estimate the dynamic states of the helicopter. In Chapter 3 we describe our initial contributions in map optimization for graph-based SLAM. Our recent extensions to the work in Chapter 3 are presented in Chapter 4. We conclude the dissertation with summary thoughts and a discussion of future work in Chapter 5.

1.1 Multi-Rotor Helicopters

As discussed in [5], multi-rotor helicopters possess several features that make them better platforms than the potential alternatives for indoor flight. Any hovercraft has the advantage over a fixed-wing vehicle in that it can pause or move more gradually in any direction within the constrained indoor air space. Lighter than air vehicles provide hover capability, but they lack sufficient payload and agility for most applications. Hovering vehicles inspired by biological systems (e.g. hummingbirds or insects) may provide sufficient agility, but they are less well understood and significantly more complex than multi-rotor helicopters. Co-axial helicopters present a viable alternative, especially due to their inherent stability. However, the authors of [5] argue that the payload capacity and scalability of multi-rotor vehicles make them more attractive.

Four- and six-rotor vehicles (respectively designated quadrotors and hexacopters) are the most common multi-rotor helicopters. As a developmental platform we use a hexacopter (see Figure 1.1) available from MikroKopter.[1] Our hexacopter weighs about 4 kg as currently configured. It carries a stripped down Microsoft Kinect RGB-D camera weighing about 230

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

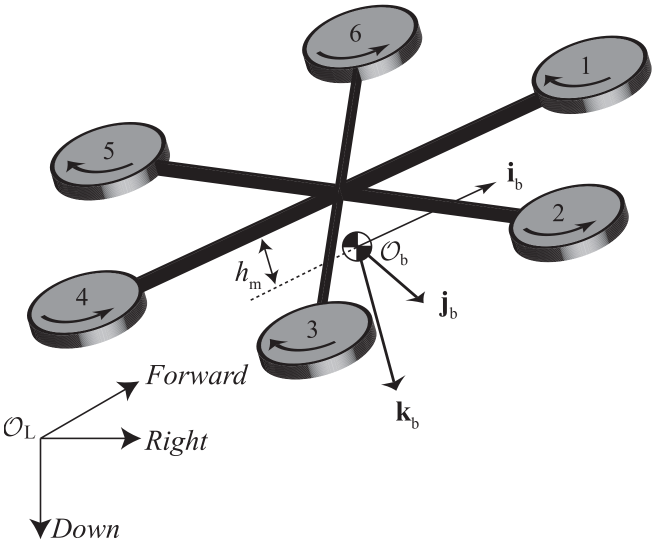

Figure 1.2: A schematic representation of a six rotor helicopter, or hexacopter. The six rotors alternately rotate in opposite directions (indicated by the curved arrows) to allow the vehicle to control its yaw. Roll and pitch, which are controlled by varying the relative motor speeds, produce the lateral accelerations of the vehicle. We use a body-fixed reference frame centered at $\mathcal{O}_b$ at a distance h_m below the vehicle. This right-handed reference frame has an $\mathbf{i}_b$ axis aligned with the vehicle's preferred forward direction and a $\mathbf{k}_b$ axis that aligns with gravity when the vehicle is at a perfect hover. We also use a local reference frame to define the vehicle's position. The right-handed local frame has an arbitrary origin, $\mathcal{O}_\mathrm{L}$, and an arbitrary heading with respect to a world reference frame aligned to north and east directions. However, the *Down* axis of the local frame is always aligned with gravity.

grams and looking in the $\mathbf{i}_\mathrm{b}$ direction. The underside of the vehicle houses a Maxbotic ultrasonic range finder[2] aligned with the $\mathbf{k}_\mathrm{b}$ axis. We use the IMU that comes standard with the MikroKopter control board.[3]

For developmental computing we have mounted to our hexacopter a large, commercially available[4] board built on the Intel 2nd Generation Core i7 processor. The computer runs at 2.1 GHz and has 8 GB of DDR3 RAM. However, we also use a smaller computer more suited for a final implementation. That computer is based on the 1.6 GHz Intel Atom Dual Core D510 with 4 GB of RAM. While the developmental board weighs about 180 grams and consumes around 50 Watts of power, the Atom board weighs in at only 90 grams and runs on about 15 Watts.

[2]`http://www.maxbotix.com/Ultrasonic_Sensors.htm#LV-EZ`
[3]`http://www.mikrokopter.de/ucwiki/en/FlightCtrl_ME_2_1`
[4]`http://www.globalamericaninc.com/global-american-inc-epi-qm67.html`

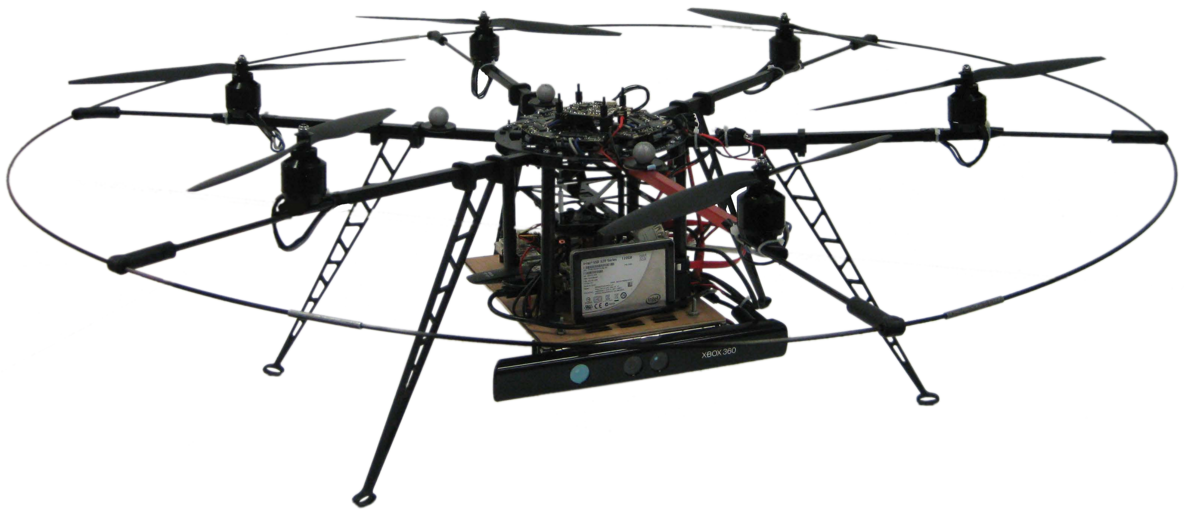

Figure 1.3: BYU MAGICC Lab's developmental indoor flight platform as of Fall 2012. The modified Kinect sensor is placed in the lower front portion of the vehicle. Directly above it is the hard drive with the onboard computer sitting behind it. The motor speed controllers and MikroKopter flight control board containing the IMU sit on top of the vehicle. Reflective dots are placed on the front, back, and back-right arms of the hexacopter to allow the motion capture system to track it. As configured, the hexacopter weighs about 4 kg and measures 13 inches in height and 42 inches in diameter.

While our current vehicle is unrealistically large for many applications, it provides a useful testbed. We have written tutorial documents to help familiarize others with our initial hardware and control architecture [7–9].

The MikroKopter provides several nice features. It is relatively inexpensive compared to the most popular alternatives from Ascending Technologies.[5] The firmware on the control board is open source and provides basic stabilization of roll and pitch angles. There is also a sizable online user community with English language forums at the official MikroKopter web site[6] and one of their US distributers.[7] Perhaps most importantly, there is a fledgling but growing community of researchers adopting the platform [4, 10, 11].

[5]`http://www.asctec.de/?locale=en_GB`
[6]`http://forum.mikrokopter.de/index.html`
[7]`http://www.mikrokopter.us/index.php?action=forum`

For prototyping and ground truth data we use a motion capture system from Motion Analysis Corp.[8] Their system provides an accurate measure of a rigid body's position and orientation by tracking a known pattern of reflective dots attached to the vehicle. The measured pose (i.e. position and orientation) information can be provided at 200 Hz to any computer over an ethernet connection. In our current facility the system can track objects within a 14' long, 10' wide, and 6' high volume.

1.1.1 Multi-Rotor Helicopter Literature

Quadrotors have been used for most, and maybe all, significant indoor flight implementations to date. We will consider only those implementations that allow for long term, repeatable navigation. Systems that rely solely on some form of odometry suffer from unbounded drift in their navigation solution.

Rudimentary efforts (e.g. [12–14]) tend to manipulate the environment in order to simplify estimating the vehicle's dynamic state. In [12], the authors enable a quadrotor to fly autonomously by using specially designed patterns observed by a monocular camera; a similar approach is taken in [13]. More recently, the authors of [14] place visual cues on top of a ground vehicle to allow a quadrotor equipped with a camera to localize with respect to those cues. Given the context of our research, viz. fully autonomous flight in completely unknown environments, these articles and others like them (e.g. [15–18]) bear only lightly on our work. They are primarily mentioned here to provide some perspective on the evolution of indoor flight using quadrotors.

More advanced systems for indoor flight relax the need to manipulate the environment. We cited above an early example of such a system published in the 2009 ICRA [4]. This article comes from the research group of Wolfram Burgard, a highly esteemed researcher in the robotics community. However, much of the article reads like an overview of the indoor flight problem. The authors invite community collaboration on the problem and provide as a first step their implementation using open-source code and commercially available hardware. Their system is built on a MikroKopter quadrotor, with a Hokuyo-URG laser range finder

[8]`http://www.motionanalysis.com/index.html`

for exteroceptive sensing and a XSens MTi-G attitude heading reference system (AHRS) for interoceptive measurements.

We note some details of the implementation in [4] that are relevant to our work. First, they directly use the roll and pitch estimates provided by the AHRS; these are based on the questionable assumption we discuss in Chapter 2 that the onboard accelerometers in the AHRS can measure the gravity vector. They rely on matching laser scans to maintain accurate estimates of the quadrotor's change in position and heading, and they use a graph-based approach to the SLAM problem for developing their map and global pose estimates. The computer onboard their quadrotor is an embedded Gumstix PC. Consequently, all but the most basic computations in their experiments are performed by an offboard laptop computer. They briefly report four separate tests for:

- Autonomously localizing within a known map;
- Autonomously developing a map online while the vehicle is flown manually;
- Autonomously estimating the change in the piecewise constant ground beneath the quadrotor, again while the vehicle is flown manually; and
- Autonomous stabilization of yaw and altitude when the other states of the vehicle are manually controlled.

The fully integrated system is not tested in this article.

The authors of [4] have published some extensions of their work in a very recent article in the IEEE Transactions on Robotics [11]. In describing the more recent article the authors say, "This paper extends our previous work by introducing improved algorithms for simultaneously estimating the altitude of the vehicle and the elevation of the underlying surface. We furthermore provide quantitative results of our SLAM approach and discuss the effect of different modes of the incremental scan-matching on the pose stability of the robot. We also describe our algorithms for path planning, obstacle avoidance and provide additional details and experiments." The most noticeable algorithmic changes appear to be in path planning, obstacle avoidance, and control. Other than modeling multiple floor levels, their approach to localization and mapping seems to be mostly unchanged. The hardware system presented in [11] is also the same. The tests presented in the results section again only test individual components of the system independently from each other, and the system still relies heavily on an offboard computer for processing.[9]

One of the more advanced systems in the literature [1] hails from the GRASP Lab at the University of Pennsylvania. Their lab is more popularly known for its demonstrations of quadrotor control given perfect knowledge of their state,[10] but this paper treats several estimation issues required to enable true autonomy. Their quadrotor system is built on an Ascending Technologies Pelican platform with its standard MEMs IMU and some custom firmware to run on the embedded processor. The vehicle caries two additional sensors: a Hokuyo UTM-30LX scanning laser range finder and a uEye 1220SE camera. The laser is used for most estimation functions; the camera's only function is to identify loop closure using a place recognition algorithm. All algorithm development is done in C++ using the Robot Operating System (ROS) from Willow Garage. The system's onboard computer is based on a 1.6 GHz Intel Atom processor with 1 GB of RAM.

To put their work in context, the authors of [1] cite [4] and some other authors that we will discuss below. They say that, "Relevant to this paper is the work of Bachrach et al., Grzonka et al., and Blosch et al. with results toward online autonomous navigation and exploration with an aerial vehicle. The major points of differentiation between existing results and our work are threefold. First, all the processing is done onboard requiring algorithms that lend themselves to real-time computation on a small processor. Second, we consider multifloor operation with loop closure. Third, we design adaptive controllers to compensate for external aerodynamic effects which would otherwise prohibit operation in constrained environments."

The requirement that processing be done onboard is, to us, the most relevant feature of [1]. Throughout the paper they detail how their design decisions are impacted by the constraint on computational resources. For example, they depend on a laser scan matching algorithm to update the 2D position and heading. Their scan matching is also their only means of correcting velocity estimates through their kinematic relationship to position.

[9]The lead author of [4] and [11] has also made quadrotor navigation one of the primary subjects of his December 2011 Ph.D. dissertation, available at `http://www.slawomir.de/publications.html`

[10]See, for example, the video available at `http://www.youtube.com/watch?v=MvRTALJp8DM`.

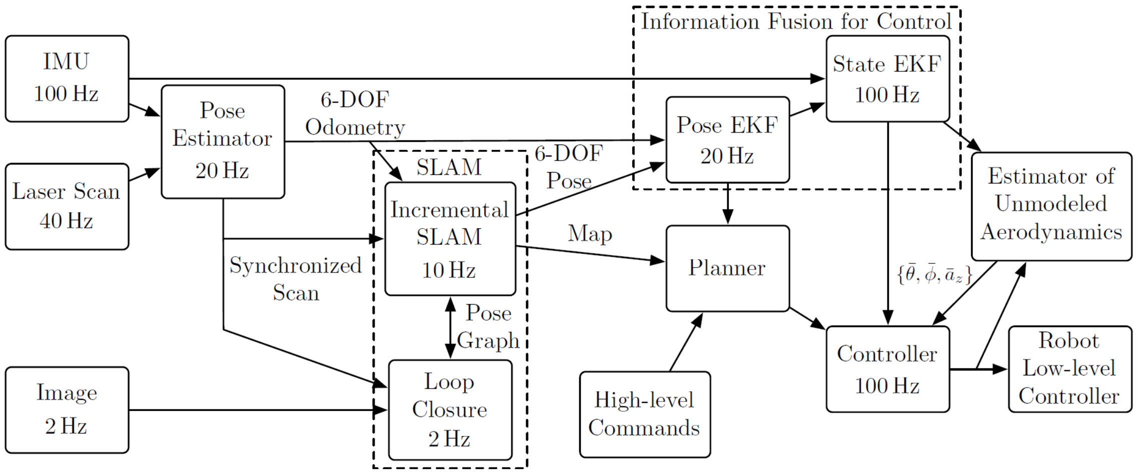

Figure 1.4: Estimation scheme used by [1] (© 2011 IEEE, used with permission). While the scheme is demonstrably effective, it is a little confusing to follow. We believe a more simple scheme can be developed based on principles described below and in Section 1.2.

However, instead of choosing a more accurate scan matching algorithm, their computational limitations force them to settle for a modified Iterative Closest Point (ICP) algorithm. As a consequence they observe that their vehicle requires these updates at a minimum rate of 20 Hz in order to control the vehicle's fast dynamics. They also perform similar trades between accuracy and complexity for their graph-based approach to SLAM which we will discuss further in Section 1.2. Their series of design decisions regarding estimation and control algorithms leads to a rather complicated scheme involving multiple filters running at different rates to meet the different needs of the system (see Figure 1.3).

The other points of differentiation listed by [1] are also noteworthy. Their approach to multiple floor levels appears similar to [11]. But their modeling of aerodynamic effects is important to the work we present in Chapter 2. They use a typical model for the quadrotor dynamics modified by a generic "disturbance vector." The resulting equation is

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} - \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \tag{1.1}$$

where m is the mass of the vehicle, the left-hand side vector represents the vehicle's acceleration in an inertial reference frame, g is gravitational acceleration, R is the rotation matrix from the body-fixed reference frame of the vehicle to the world frame, and F is the total thrust produced by the motors. The disturbance vector $[f_x \ f_y \ f_z]^\top$ captures all other aerodynamic effects and is "assumed to be a slowly varying term." They show that using this model helps to make their system more robust to wind or the vehicle's own propwash.

In a 2012 ICRA paper the authors of [1] also detail a 3D exploration algorithm for use on their indoor quadrotor [19]. In both [1] and [19] the authors present impressive flight test results. It is interesting to note in [19] that this team from the GRASP lab is adopting the Kinect RGB-D camera from Microsoft. They currently use the Kinect in the exploration algorithm to provide 3D depth data at close range (< 4m). The Kinect complements the same scanning laser used in their previous work which measures obstacles at distances up to 30m. We expect they will continue to develop algorithms that leverage the Kinect data in future work. Toward the end of this section we will further discuss the applicability of RGB-D cameras like the Kinect for indoor flight.

A team from the Robust Robotics Group (RRG) at MIT is responsible for another advanced indoor flight system. In 2009 they entered their vehicle in the annual International Aerial Robotics Competition (IARC), a competition sponsored by the Association for Unmanned Vehicle Systems International (AUVSI). The 2009 IARC was the first ever offering of an indoor flight mission in this long running competition. The vehicles were required to autonomously takeoff outside a simulated building, identify and enter through a 1 sq. meter window, and then negotiate the hallways of a 600 sq. meter arena in order to find a target marked by a solid blue LED.

The RRG team completed all of the tasks in the 2009 IARC.[11] They give a comprehensive description of their system in [2], though several of their earlier papers also describe the system [20–23]. Some of their design philosophies and observations about indoor flight are central to our thinking about the problem.

[11]In passing we note that the IARC organizers have since felt the need to make the current indoor flight mission much more challenging. On their website they state, "The new 6th mission is an extension of the 5th mission theme of autonomous indoor flight behavior, however, the 6th mission demands more advanced behaviors than are currently possible in any existing aerial robot." We believe they are currently correct; see their website `http://iarc.angel-strike.com/` for the competition's details.

Like [1], the authors of [2] also use the Pelican quadrotor by Ascending Technologies, a platform that they actually designed for their IARC entry.[12] The onboard microcontroller that comes with the vehicle runs at 1 kHz to stabilize roll and pitch and provides filtered IMU measurements at 100 Hz. For exteroceptive sensing the vehicle uses the Hokuyo UTM-30LX scanning laser range finder that runs at 40 Hz and has a maximum range of 30m. Also like [1], a 1.6 GHz Intel Atom with 1 GB of RAM provides the onboard processing. The authors state that this processor is sufficient to handle all of the "real-time" processing demands of the system, but they still must rely on a more powerful ground station computer to handle their SLAM processing.

We especially hold to the principle in [2] that they call "decoupled system architecture." The idea is simply that time critical processes should be separated from and given priority over any function that is not essential for keeping the vehicle in the air. The authors of [2] use Figure 1.4 to illustrate their application of this concept. The boxes in the middle of the chart, highlighted in yellow, run at the rate of the laser scanner (40 Hz) and constitute the time critical elements for staying airborne. SLAM and the path planning run at a much lower rate, though in [2] they are not entirely independent from the real-time needs of the vehicle. In Section 1.2 we will explain how we expect to apply this principle of decoupling real-time and background tasks, hopefully to the end that all of the processing can be performed onboard with minimal sacrifices in accuracy.

The authors of [2] make another observation about their system that is important to our work. The heart of their navigation system is their laser scan matching algorithm. In fact, among their concluding comments they state, "The fast dynamics of [quadrotors] reduce the amount of computation time allowed for the real-time state estimation algorithms. ... Our work placed special emphasis on the development of a very fast scan matching algorithm that maintained the accuracy and robustness required for use on a [quadrotor]. The laser scan-matcher ... was the key enabling technology that allows our vehicle to fly in GPS-denied environments." Similar to [1], the scan matcher in [2] needs to be fast because it is

[12]Conveniently, one of the students on the RRG team is the brother of one of the founding members of Ascending Technologies.

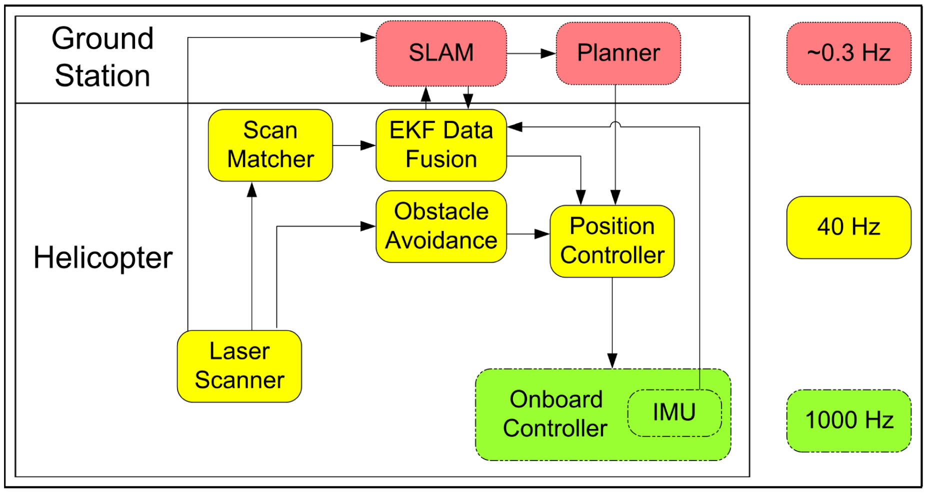

Figure 1.5: Estimation scheme used by [2] (© 2011 IEEE; used with permission). The important element to note is the simple distinction between processes required for real-time operation and those that are less critical.

the only means of correcting the velocity estimates that are fundamental to damping the dynamic behavior of the quadrotor. We discuss this problem further in Chapter 2.

The vehicle dynamic model used in [2] is worth a brief mention here. They rely heavily on the assumption of near hover flight, though that is a reasonable assumption for most autonomous indoor operation. The linear accelerations of the quadrotor are kept in the state vector that is estimated by the EKF Data Fusion filter depicted in Figure 1.4. Keeping the body-frame accelerations in the state vector and assuming roll and pitch estimates are sufficiently accurate leads them to use accelerometer measurements in a direct update of those states. In fact, their estimate of the body-frame z-axis acceleration is simply propagated with a random walk, relying entirely on the measurement update for correction.

They propagate the body x- and y-axis accelerations using the model

$$\ddot{x}^b = k_\theta \theta - k_{\dot{x}} \dot{x}^b + \omega_{\ddot{x}}, \tag{1.2}$$

$$\ddot{y}^b = k_\phi \phi - k_{\dot{y}} \dot{y}^b + \omega_{\ddot{y}}, \tag{1.3}$$

where the terms in Equation (1.2) are defined such that $\ddot{x}^b$ and $\dot{x}^b$ are respectively the acceleration and velocity of the vehicle along its body-fixed x-axis, k_θ and $k_{\dot{x}}$ are constant parameters associated with the hover state of the vehicle, θ is the vehicle's pitch, and $\omega_{\ddot{x}}$

is the zero mean, Gaussian process noise. Analogous terms in Equation (1.3) are similarly defined.

The constant parameters $k_{\dot{x}}$ and $k_{\dot{y}}$ are the aspect of Equations (1.2) and (1.3) that are most relevant to our work. Regarding these parameters the authors state, “While the damping parameters $k_{\dot{x}}$ and $k_{\dot{y}}$ are small enough that they could be ignored when the vehicle is in the hover regime, they are included here to prevent the model from allowing the estimated velocity to grow without bound in the absence of position corrections. The model parameters are learned by a linear least-squares system identification process from flight and control data collected offline in a motion capture studio.” We note that they make no attempt to understand the physical reality associated with these constants. We rejoin this topic in Chapter 2.

Finally, we mention here one more important comment from [2]. They represent the states of their vehicle and elements of the environment in a globally metric reference frame. By this we mean that they choose an arbitrary origin in the environment, usually the initial location of the vehicle, and then represent the position of the vehicle and other features in the environment by measuring them from that origin. This approach is almost universal among those designing indoor flight systems, but [2] cites it as a limiting factor, especially when operating in complex 3D environments. They state that, “Fully handling these types of challenging scenarios will likely require relaxing the need to maintain a [globally] metric representation of the environment and the state of the vehicle.” We base our approach to the indoor flight problem on a locally metric but globally topological representation of the environment and vehicle states. We will describe that further and discuss the ramifications of that representation in Section 1.2

The advanced implementations in [1, 2, 4] rely on a scanning laser range finder to sense the environment. One must still make some strong assumptions about environment structure in order to navigate in 3D with a scanning laser range finder. These assumptions at least include the presence of vertical walls and piecewise constant floors. More subtle is the assumption that potential obstacles can be observed with only the thin slice of the environment sensed by the laser. In discussing their system, the authors of [2] make the following observations about their test flights: “Some areas of the Stata Center at MIT have walls that are slanted. If the slanted walls subtend a large enough portion of the field of view of the laser scanner, the scan-matcher will be unable to disregard as outliers the false matches between scans taken at different heights. Similar failures occur in outdoor environments when the laser scanner predominately observes the leaves and branches of trees (as opposed to vertical tree trunks)." They further state that, "[M]ore work will be required to unlock the full potential of quadrotors to operate in 3D. The incorporation of visual information from camera sensors will be an area of particular interest." We require truly 3D sensors to operate in the unstructured 3D settings described in Figure 1.0.[13]

An initial step in this direction is taken by researchers at MIT in an article from the 2009 ICRA [24]. They equip an Ascending Technologies Hummingbird quadrotor with a forward looking wireless camera and simulated IMU data.[14] All significant processing is performed on a desktop computer.

Though they use only a monocular camera, the navigation algorithm used by [24] processes multiple monocular images to develop a 3D representation of the world. Their approach closely resembles popular feature-based SLAM algorithms [25, 26]. However, these algorithms are known to suffer from problems with computational complexity, as well as drift in the map's scale due to the inherent uncertainty in monocular images [27]. The results presented in [24] are limited. They demonstrate that the vehicle can hover without drifting, and they present some plots of true vs. estimated position for a short 5 meter long flight.

A team from the Autonomous Systems Lab (ASL) at ETH Zurich developed another early quadrotor system built around vision processing [28]. In their 2010 ICRA article they use an Ascending Technologies Hummingbird quadrotor equipped with an IMU and a downward looking monocular camera. They introduce their article by saying that, "To the best of our knowledge, this is the first work describing a micro aerial vehicle able to

[13]We recognize that many indoor environments, including those in Figure 1.0, would not admit the use of passive cameras due to poor lighting. We expect that RGB-D and time-of-flight cameras, along with algorithms to process their data, will continue to mature such that these active sensors could be used for indoor flight in such settings. This does not change the context of our problem; extracting the necessary 3D information from such data should still be processing intensive.

[14]The simulated IMU data is produced by flying the vehicle within a motion capture studio. The motion capture system measures the vehicle's attitude and position at a high rate and uses those measurements to synthesize gyroscope and accelerometer measurements by adding noise to the velocities approximated by numerical derivatives.

navigate through an unexplored environment (independently of any external aid like GPS or artificial beacons), which uses a single camera as only exteroceptive sensor." Like [24], they use a feature-based monocular SLAM algorithm to develop their map [29]. However, the algorithm they use is intended for "small [artificial reality] workspaces" and quickly becomes too computationally expensive for even modest sized maps. This is likely why the system presented in [28] requires a USB cable connection to a ground station computer to process vision data. They use real IMU data and present more significant hardware test results than [24], but the substance of their systems is similar.

ASL researchers[15] continue to work with quadrotors as a platform for indoor and outdoor navigation. Much of that current research is driven by a multi-national European research initiative named sFly. The sFly project is geared toward designing swarms of quadrotors operating outdoors or indoors using only monocular cameras for exteroceptive sensing. Additional publications from the ASL related to sFly can be found at the project's web page.[16]

A separate ETH Zurich team from the school's Computer Vision and Geometry (CVG) Group developed a more capable quadrotor system for indoor flight. In their 2011 ICRA paper they present the system they name PIXHAWK [3]. The system is based on a custom built quadrotor capable of carrying a 400 g payload. They equip the vehicle with an IMU and two pairs of stereo cameras.

The authors spend only a little time in [3] discussing their approach to estimating the vehicle states. They treat the global heading and each axis of the vehicle's global position as four independent subsystems and use a constant velocity motion model for state propagation. They claim the vehicle can go several seconds without an update from the vision algorithms that measure position and heading. However, they also concede that the simplicity of their dynamic model, "is a valid approximation [only] if the filter update frequency is fast enough with respect to the change of speed of the physical object." For our work in Chapter 2 we also note in passing that, like almost all quadrotor users, they assume the accelerometers in the IMU can measure the gravity vector.

[15]In fact, one of the Ph.D. students now at ASL is a member of the MIT RRG team that produced the system discussed above in [2]

[16]`http://www.sfly.org/`

Keeping all computation onboard is a central focus of [3]. Most computation occurs on a custom designed computer based on a 1.86 GHz Intel Core 2 Duo with 2 GB of DDR3 RAM. It is perhaps an insight into their design philosophy that they suggest that, "future upgrade options include Intel i7 CPUs." They note that the weight of the current computer is 230 g, 58% of the vehicle's payload capacity, but they don't mention how much an i7 computer would weigh with the larger heat sink it would require. While it may work for ground vehicles, simply increasing the power of one's computer is a suspect proposition for small multi-rotor helicopters.

However, it is understandable why one would want greater computing resources for a fully autonomous quadrotor using truly 3D sensors. The authors of [3] report that 10% of their CPU capacity is consumed just by identifying the vehicle's position and heading with respect to predefined markers placed on the floor in known locations (see Figure 1.5). They also note that, "Higher-level [tasks] such as stereo obstacle avoidance and pattern recognition . . . [require] significantly higher load in the range of 40 - 60% [of the computer's capacity] if run in parallel." Despite their own report that 70% of CPU capacity is already allocated, they claim that this "leaves enough capacity for future work, including simultaneous localization and mapping." They provide no evidence to support that assertion.

The system presented in [3] is undoubtedly one of the most advanced systems capable of autonomous indoor flight that is presented in the literature, but it is also not a complete solution to the problem presented at the beginning of this Chapter. However, the authors and their associates have been extending the work in [3] since it was published in the 2011 ICRA. In a 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) article [30] the authors replace the artificial markers used in [3] with (1) a forward looking stereo camera performing visual odometry and (2) a downward looking monocular camera paired with a sonar range sensor to compute optical flow. We also note that the pictures of hardware in [30] appear to show that they have switched out some components of their system for components from MikroKopter. They introduce algorithms for autonomous exploration, path planning, and local occupancy grid mapping that are performed onboard the vehicle using the same computer from [3]. These algorithms allow for more sophisticated autonomous behavior than was shown in their prior work, but some important aspects of

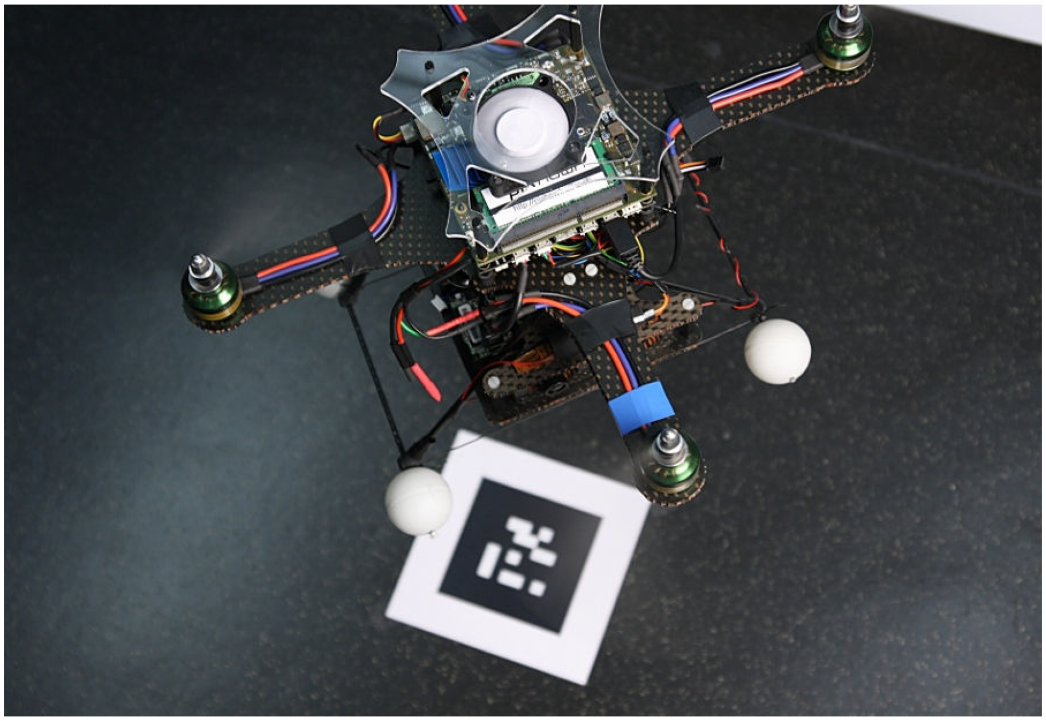

Figure 1.6: The PIXHAWK platform and artificial navigation marker described in [3] (© 2011 IEEE; used with permission). The size, nature, and location of the markers on the floor are known a priori. The vehicle determines its position relative to the marker by estimating the homography that maps the known dimension of the four marker corners into the current 2D image. The vehicle's heading and the absolute location of the marker and the vehicle can be extracted by analyzing the 2D pattern inside the marker.

these new algorithms depend on the same simplifying assumptions used with a 2D scanning laser range finder, namely piecewise constant floors and vertical walls. It is also relevant for us to note that the authors in [30] adopt g^{2}o, an open-source graph-based SLAM tool, to develop a global map of the environment. They run this SLAM algorithm on a separate offboard computer.

As a final example from the literature, we cite a unique article from the 2011 International Symposium on Robotics Research [31]. The paper is a collaborative effort between some of the authors from the MIT team discussed above [2] and researchers at the University of Washington led by Dieter Fox. The article is unique in that it is the first work we are aware of that uses an RGB-D camera on a quadrotor to enable autonomous flight through indoor environments.[17]

[17]It is actually the only such work of which we are aware at the time of this writing.

The system demonstrated in [31] is built around the Ascending Technologies Pelican. They use the computer developed for the PIXHAWK project [3], the 1.86 GHz Core 2 Duo with 4 GB of RAM. For their RGB-D camera they use a stripped down Microsoft Kinect sensor that only weighs 115 grams.

Much of the system design in [31] parallels that of [2]. The system depicted in Figure 1.4 is modified to replace Laser Scanner and Scan Matching with RGB-D Camera and Visual Odometry, respectively. The authors in [31] use a different graph-based SLAM algorithm this time, but it still runs on an offboard computer at a much slower rate than those processes involved in the real-time control loop. The results presented in [31] are limited to (1) a comparison of position estimates vs. motion capture truth during hover, and (2) a qualitative assessment of short flights through small rooms.

The Kinect and other RGB-D cameras like it provide a useful tool for navigating through unstructured indoor environments. The dense depth data is directly useful for obstacle avoidance and near term path planning. It is also easier to extract 3D point features from the Kinect data compared to stereo vision. The Kinect's onboard processor associates depth data with every pixel in the RGB image, relieving the burden of stereo processing on the vehicle's main computer. However, the Kinect itself comes with some downsides. The structured light sensor used to calculate depth is limited to a range of 4 meters. While this works well in many indoor settings, looking down a long hallway or into a large auditorium would make at least most of the depth data unusable. Also, the rolling shutter on the camera imposes limits on how fast the vehicle can move without inducing excessive blur in the image. Again, this is not a problem during many indoor maneuvers, but it is a limitation the system would have to plan around during autonomous operation, e.g. so as to not yaw too quickly.

To conclude this discussion about multi-rotor helicopters, we observe that they are proven and capable platforms for indoor flight. Several notable institutions (e.g. UPenn, MIT, ETH Zurich) have labs that are actively engaged in using these vehicles to produce increasingly more advanced indoor flight systems. However, we have not seen a system yet that can address the scenario outlined at the beginning of this Chapter. The most important issue to address is the computation required for truly autonomous localization and mapping without use of simplifying assumptions about the environment. Our work in Chapter 2 helps to relieve this problem for localization. We believe this issue can also be effectively addressed by designing an appropriate approach to the SLAM problem that we discuss below.

1.2 Navigation Using SLAM

A vehicle has two options when navigating in an unknown environment without external aids: dead reckoning and simultaneous localization and mapping (SLAM). Both of these approaches are problematic for an indoor air vehicle. All forms of dead reckoning are subject to unbounded drift in the estimated position and heading of the vehicle in a global reference frame. Inertial guidance in particular suffers from the low quality inertial measurement unit (IMU) an indoor air vehicle is able to carry. SLAM addresses this problem by constructing a globally consistent map as the vehicle move about. But SLAM algorithms tend to be computationally expensive and are therefore constrained by the limited processing power available onboard. We begin this section with some general comments on SLAM, followed by some specific observations about the paradigm called graph-based SLAM [6]. We conclude the section and the chapter with a review of the literature on graph-based SLAM

1.2.1 General Comments on SLAM

SLAM algorithms come in many forms, and SLAM has been an active subject of research in and of itself since the foundational papers (e.g. [32], [33]) in the 1990's. One can find excellent tutorial material on SLAM from a number of sources (e.g. [6, 34–38]). The 2008 overview of SLAM presented in [34] provides a useful taxonomy for classifying SLAM algorithms. We summarize here some of the more important terms they introduce to facilitate our further discussion of SLAM:

- **Topological vs. Metric -** These distinctions apply to the type of map used to model the environment. A fully topological map may model the world as a set of abstract places while only tracking the qualitative arrangement of those places. At the other extreme, a fully metric map may track the global coordinates of every mapped aspect of the environment. This characterization of SLAM algorithms is best thought of as a continuum rather than a binary classification. Most sophisticated SLAM algorithms use a blend of both topological and metric elements in their map (e.g. [39], [40], [41]).

- **Volumetric vs. Feature-Based** - Here again we refer to the nature of the map used to represent the environment. A volumetric map makes a dense representation of the world; it describes the vehicle's surroundings with data representative of all or most of the sensor's field of view. A feature-based map extracts and saves a much smaller subset of the data, reducing the raw data to a collection of landmarks that characterize the environment.

- **Online vs. Offline** - These terms are meant to distinguish between causal SLAM algorithms that incorporate data and provide results in real time (Online), and algorithms that operate on the data post facto (Offline). While nothing in the literature explicitly makes the following distinction, we feel online and offline designations correlate respectively with another divide that seems to exist in the SLAM literature: whether the SLAM practitioner is primarily concerned with localizing the vehicle or with generating a map. In the former case, making some sort of map may be treated as a secondary consideration necessary for maintaining a quality navigation solution. In the latter case, localizing the vehicle may be treated as simply a necessary part of generating a quality map. While subtle, we feel this distinction in emphasis between mapping and localizing is important to be aware of when reviewing SLAM literature or selecting an appropriate SLAM approach from those already extent.

Loop closure is another key attribute of all SLAM algorithms; it is the trait that distinguishes SLAM from mere dead reckoning. Loop closure is the process by which the algorithm recognizes when the vehicle has returned to a prior location in the map. This recognition allows the algorithm to incorporate the resultant added information about the vehicle's pose. Closing the loop constrains the drift in the vehicle's pose estimates that would otherwise grow without bound over time.

SLAM algorithms can also be conveniently categorized as belonging to one of three major paradigms [34]. The first to appear in the literature, and still very popular in some circles, applies the extended Kalman filter (EKF) to the SLAM problem. The original online EKF-SLAM algorithm uses a single state vector to jointly estimate a feature-based, metric map and the current pose of the vehicle.

While EKF-SLAM rests on the familiar foundation of Kalman filtering, it is often criticized. Perhaps the most cited, though somewhat outdated, critique addresses computational complexity. In a naive implementation of EKF-SLAM, updating the entire covariance matrix requires computation on the order of N^2 at every time step, where N is the number of landmarks in the map. While this would seem to severely limit the size of the attainable map, excellent work has been presented in [27] and subsequent papers demonstrating how to build large-scale EKF-SLAM maps in amortized linear time and without any approximations beyond the standard EKF linearization. This is done by separating the map into local submaps and exploiting the probabilistic structure of the SLAM problem. We discuss [27] at greater length in Chapter 3.

Another important critique of EKF-SLAM arises from the linearization inherent in the the EKF. In [42] and [43] the authors show naive EKF-SLAM is bound to eventually become inconsistent. By 'inconsistent' we mean the error in the state estimates, $\tilde{x}(k) \overset{\triangle}{=} x_{true}(k) - x_{est}(k)$, fails to satisfy one or both of the following conditions:

$$E\left[\tilde{x}(k)\right] = 0,$$
$$E\left[\tilde{x}(k)\tilde{x}(k)^T\right] = P(k),$$

where $x_{true}(k)$ and $x_{est}(k)$ respectively represent the true and filter-estimated states at time step k; $E[\cdot]$ denotes probabilistic expectation; and $P(k)$ is the filter-calculated covariance [44, pp. 232-233]. However, the authors of [45] and [46] have helped identify the root cause of this phenomenon and have prescribed a few solutions.

Finally, because of the Kalman filter's Gaussian assumption the distribution of uncertainty about the estimates is unimodal by design. EKF-SLAM therefore requires irrevocable decisions at every time step about how to associate information from sensor measurements with existing features in the map. The EKF-SLAM solution can be quite brittle to errors in this data association process.

The laundry list of complaints against EKF-SLAM is partly indicative of the algorithm's maturity. It is well studied, reasonably well understood, and a great place to start to understand SLAM generally. The best introductory material for EKF-SLAM can be found in [35] and [38].

A second paradigm for approaching the SLAM problem is characterized by the algorithm introduced by [47] and dubbed FastSLAM. This method uses the Rao-Blackwellized particle filter to develop online estimates of a feature-based, metric map along with an entire history of vehicle poses. FastSLAM and other particle filtering SLAM algorithms take advantage of the fact that features can be independently estimated on the condition that the vehicle's trajectory is known. Monte Carlo sampling is used to generate several estimates (i.e. particles) of the vehicle's trajectory, and then each particle keeps track of its own N independent EKF's to estimate the positions of the N features.

FastSLAM proponents tout the fact that its computational complexity scales logarithmically in the number of features, as opposed to the quadratic complexity of naive EKF-SLAM. This benefit is more of a straw man argument due to the work of [27] and related algorithms. FastSLAM also offers a natural means for making multiple data association hypotheses in ambiguous cases and for modeling multimodal uncertainty [48]. Unfortunately, particle filters must be regularly resampled to ensure the set number of particles is concentrated on likely estimates of the vehicle trajectory. In [49] the authors show this resampling invariably leads to statistical inconsistency because the current set of particles will eventually all descend from a single, biased particle. An introduction to FastSLAM is available in [34] and [37], though each seems slightly biased by the author's close association with the approach.

Both EKF-SLAM and particle filter SLAM are inherently feature-based. In other words, they reduce the raw sensor data into a set of features. This makes the resulting map less useful for tasks other than localization, such as path planning or rendering a human friendly map that increases an operator's situational awareness.

1.2.2 Graph-Based SLAM

Graph-based SLAM is an increasingly popular alternative to EKF-SLAM and particle filter SLAM. Graph-based SLAM was introduced by [50]. It has "undergone a renaissance" in recent years due to improved insights into the structure of SLAM and new methods for solving sparse linear systems [6]. Of the above referenced indoor flight implementations, [1, 2, 4, 11, 19, 28, 30, 31] all use some type of graph-based SLAM algorithm. All of these articles have been published since 2009, highlighting the recent popularity of the approach.

In graph-based SLAM, a history of estimated vehicle poses are thought of as nodes in a graph. The edges in the graph encode the relative transformation[18] between the nodes. Edges between temporally consecutive pose nodes are typically derived from odometry information. Loop closures are also represented as an edge between temporally distant poses. Therefore, an estimator that generates the edges in the graph is an integral part of graph-based SLAM; this estimator is generally referred to as the graph-based SLAM "front-end" [6].

The map in graph-based SLAM is made by associating exteroceptive sensor data with the vehicle pose nodes. This pairing of data to poses can be done in a number of ways. To develop a feature-based map, features can be made into nodes and connected by an edge to vehicle pose nodes. The edge from a pose to a feature represents the relative position of the feature in the local reference frame of that pose. Alternatively, raw sensor data, such as complete laser scans or stereo camera images, can be saved with the vehicle pose node from which they were collected to produce a volumetric map. In this latter case, the resulting graph is referred to as a pose-only graph, or pose graph.

We note here that the original work on graph-based SLAM [50], as well as many subsequent articles from across the community, employ a pose-only graph. We will tend to focus our discussion on the pose graph case. It may seem impractical to favor raw data over reducing that data into a smaller collection of features. However, a quick calculation shows it is not unrealistic to save raw data. Our current indoor flight system (Figure 1.2) uses a 120 GB solid state hard drive. We could safely allocate about 26 GB for storing Kinect frames that are each about 1.3 MB. That equates to about 20,000 images which, if saved about every 0.5 meters of flight, could represent an ample 10 km of trajectory. Saving the raw sensor data also facilitates higher level planning like exploration and interaction with the environment.

[18]By "relative transformation" here we mean a rigid body transformation defined in the local reference frame of the global poses from which the transformation originates.

Whatever the graph architecture, the collection of edges estimated in the front-end constitutes a globally topological map. Metric estimates are only defined in the local reference frame of their associated poses. A separate estimator is required to refine locally metric estimates into a globally metric map. The estimator that performs this function is referred to as the "back-end optimization" part of graph-based SLAM [6].

It is the back-end estimator that really gives graph-based SLAM its identity. The same outputs from the front-end could be used as measurement updates in a SLAM algorithm based on the EKF or the particle filter. Accordingly, the following literature review focuses on advances in back-end optimization.

Before reviewing the literature, we recall here two of the principles espoused by [2] for developing an effective indoor flight system. Regarding their "decoupled system architecture" the authors write, "Rather than using a single integrated state estimator, we realized that it was critical to develop a local estimator for controlling the position of the vehicle, with special emphasis on estimating and controlling the velocity. The multi-level system hierarchy presented [in Figure 1.4] allows for this decomposition and was a critical development that allowed our system to work. Additionally, once the position of the vehicle is accurately controlled, the task of designing and implementing higher level algorithms [such as mapping and path planning] is greatly simplified." And regarding relative navigation the authors state, "Fully handling these types of challenging scenarios," i.e. navigation in complex, unstructured 3D environments, "will likely require relaxing the need to maintain a [globally] metric representation of the environment and the state of the vehicle." Graph-based SLAM naturally allows for relative navigation and a decoupled system architecture.

Navigation in the graph-based SLAM front-end is fundamentally relative to local reference frames. Raw measurements made in the front-end are inherently relative measurements. For example, the vehicle senses feature A to be 1.5 meters ahead and 0.3 meters to the right of its current position. Similarly, any interaction with the environment (e.g. obstacle avoidance, manipulating a target, etc.) will be accomplished based on current sensor data relative to the vehicle's current state. Front-end decisions about exploring uncharted territory are most intuitively based on the open space estimated relative to the vehicle's current location. Even plans to revisit a distant area of the indoor environment should be made along the chain of relative transformations between saved poses because reaching a distant goal along a path through unexplored indoor space will likely meet with obstructions.

Relative navigation also leads to a natural decoupling of the front-end and back-end components of graph-based SLAM. The purpose of the back-end is to refine the locally referenced transformations using an optimization process that yields an accurate, globally metric map. Performing relative navigation in the front-end means the vehicle does not immediately depend on those globally metric states. Therefore, we can move the computationally challenging optimization of the saved relative transformations completely out of the time critical path of the system.

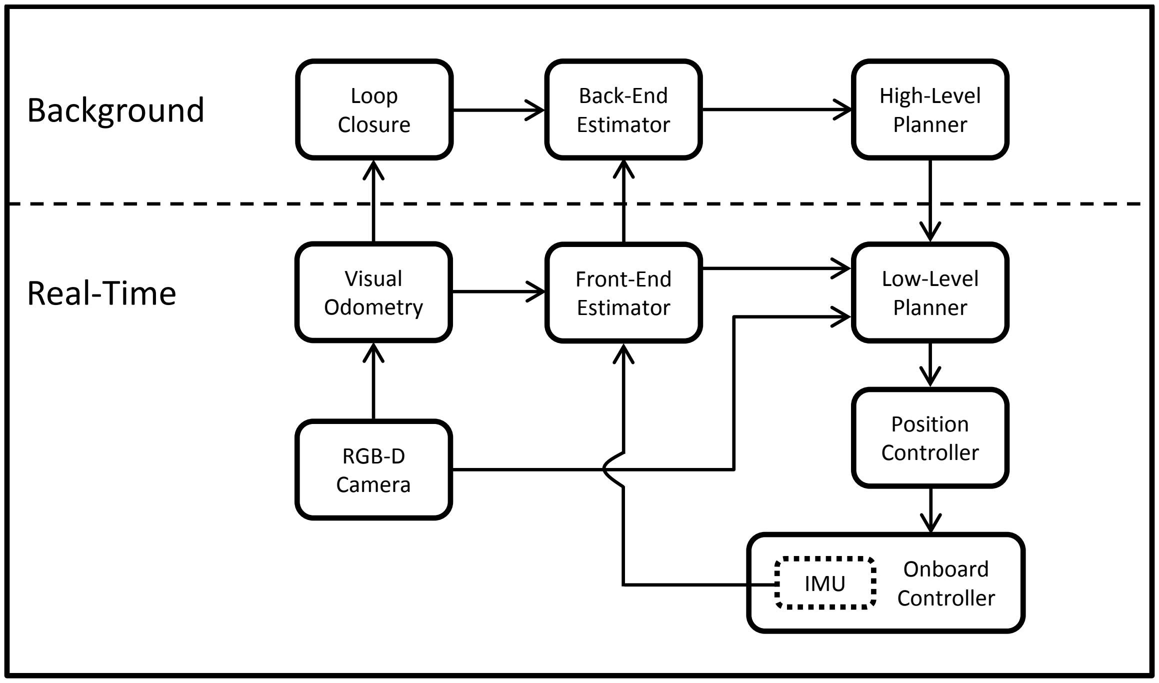

Figure 1.7: Our proposed navigation scheme. The system is based on graph-based SLAM. The time-critical navigation in the front-end is performed relative to reference frames associated with saved images of the environment. Refining the map happens when computational resources permit, and it only feeds directly into the vehicle's high-level decision making.

Our concept of relative navigation using graph-based SLAM is summarized in Figure 1.6. Following is a description of the system components represented by the blocks in this diagram.

- Visual Odometry (VO)
  - The VO takes in RGB-D camera data and estimates the relative transformation to its current pose from the most recently saved reference image. VO then feeds that estimate to the front-end estimator. The VO process also alerts the front-end estimator when a new reference image is selected so that the estimator can reset the appropriate state estimates to be relative to that new reference.
  - VO passes each new reference image to the loop closure process.
- Front-end Estimator
  - The front-end estimator is described further in Chapter 2; it fuses the VO estimate with IMU data to produce the best estimate of the vehicle's relative transformation from the current reference image. These estimates are continuously made available to the low-level planner.
  - The front-end estimator passes its relative transformation estimates to the back-end estimator whenever the VO declares a new reference image. These estimates constitute the odometry-like relative transformations discussed in Chapter 3 because they describe relative transformations between temporally consecutive reference images.
- Loop Closure
  - The loop closure process compares temporally distant reference images to determine if a relative transformation between them can be measured. When it detects and estimates a relative transformation, that estimate is passed to the back-end estimator.
- Back-End Estimator

    - The back-end estimator is an optimization process that takes the loop closure and odometry-like transformations and tries to minimize the error in their estimates. We treat this subject in detail in Chapters 3 and 4.
    - We note a difference here between the scheme of [2] shown in Figure 1.4 and our approach in Figure 1.6. Because we perform all navigation in the front-end relative to local reference frames, the output of the back-end estimator does not feed back into the time-critical state estimation at all. The optimized results from the back-end enter into the vehicle's behavior through decisions in the high-level planner. This allows even greater decoupling between the mapping and localization aspects of navigation.

- High-Level Planner
    - The high-level planner is a subject of future work, but we expect it to take the optimized map of the environment from the back-end estimator and make high level decisions based on the vehicle's mission.
    - As an example, when the vehicle needs to explore new territory the high-level planner might decide where would be best to explore next. It would then identify some reference images nearest to the desired frontier and identify those as the goal for the low-level planner.
    - As another example, a human operator or the high-level planner itself might identify a feature in a previously visited reference image that the vehicle should revisit and interact with. The high-level planner would again direct the low-level planner to reach that reference image. It might also instruct the low-level planner on the relative metric location of the target.
    - The optimized global information from the back-end estimator would be used in the high-level planner to inform its decisions. For example, upon receiving newly optimized information from the back-end, the high-level planner may recognize that a set of reference images previously estimated to be far from each other are actually close together in the globally metric map. The planner might then decide to search for a previously unidentified passage between them.

- Low-Level Planner
  - The low-level planner is also a subject of ongoing work.
  - The low-level planner accepts goals from the high-level planner, current relative pose estimates from the front-end estimator, and raw sensor data. It then determines a path from its current pose through the chain of locally metric spaces to reach the goal.
  - To perform its function, the low-level planner must have access to the map. Recent additions to the map can be passed to it from the front-end estimator. Optimized values for map elements can be passed from the high-level planner.
  - The low-level planner executes functions like reactive obstacle avoidance.
  - The low-level planner is responsible for making smooth transitions in the desired metric states as the vehicle moves between adjacent local reference frames.
- Position Controller
  - This is simply a controller (PID, LQR, etc.) that takes the error between the desired and current pose and outputs the control signals (desired roll angle, pitch angle, yaw rate, and total motor thrust) to the helicopter's Onboard Controller.
- Onboard Controller
  - The onboard controller generally comes included with commercial multi-rotor helicopters. It stabilizes the vehicle about the desired values it receives from the position controller.
  - We currently use the IMU embedded in the onboard controller to provide input to the front-end estimator. A separate IMU could be used instead.

1.2.3 Graph-Based SLAM Literature

In 1997 Lu and Milios [50] proposed optimizing a graph of global poses as a robust solution to the SLAM problem. Their objective was, "to maintain all the local frames of [exteroceptive] data as well as the relative spatial relationships between local frames. ...

Consistency is achieved by using all the spatial relations as constraints to solve for the data frame poses simultaneously." In other words, a front-end system would determine the relative transformations between key poses where saved data was acquired in the globally referenced environment. By "relative" they meant that the transformations between global poses were defined in the reference frame of the pose from which they originated. Once relative transformations were estimated in the front-end, they proposed that those estimates be frozen as constraints to be used in a back-end process to find the most likely arrangement of the global poses. The optimized global poses then aligned their associated data into a consistent map.

To further emphasize the nature of their approach, they stated that, "We treat relations [i.e. relative transformations] as primitives, but treat locations [i.e. global poses] as free variables. ... [W]e do not directly update the existing relations in the network when new observations are made. We simply add new relations to the network. All the relations are used as constraints to solve for the location variables which, in turn, define a set of updated and consistent relations. ... We do not deal with [i.e. optimize] the relations directly."

We reiterate here that, like Lu and Milios, we tend to limit our discussion to graphs containing only relative transformations between global poses. Some of the papers cited below also optimize the global position of features in the environment by accounting for the relative transformations between those features and the several poses from which they were observed. We will mention features in our discussion of the literature where important, but a number of these references also explain how pose-to-feature constraints can be subsumed into pose-to-pose constraints.

Several algorithms building on [50] have since been presented [51–65]. Optimization is achieved by minimizing the error measured by the squared Mahalanobis distance. Let $\boldsymbol{\rho}$ represent a vector of all of the global poses to be optimized, with individual poses designated by $\boldsymbol{\rho}_i$. We define $\boldsymbol{\tau}_i^j$ to be the relative transformation from $\boldsymbol{\rho}_i$ to $\boldsymbol{\rho}_j$. Let the probability distribution $\mathrm{p}(\boldsymbol{\tau}_i^j)$ represent the Gaussian distributed estimate of $\boldsymbol{\tau}_i^j$ with covariance matrix $\boldsymbol{\Sigma}_{i,j}$. Finally, let $\mathbf{h}(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j)$ denote a function that takes $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$ as inputs and returns the relative transformation between them. Finally, let $\mathcal{C}$ be the set of pairs of indices for which a relative transformation has been estimated by the front-end. Using this notation, the error metric to be minimized is written as

$$\epsilon(\boldsymbol{\rho}) = \sum_{(i,j)\in\mathcal{C}} \left(\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right) - \boldsymbol{\tau}_i^j\right)^\top \boldsymbol{\Sigma}_{i,j}^{-1} \left(\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right) - \boldsymbol{\tau}_i^j\right), \tag{1.4}$$

where $\epsilon(\boldsymbol{\rho})$ indicates that the scalar error, ϵ, is a function of all the global poses, $\boldsymbol{\rho}$.

Some early efforts at back-end optimization applied relaxation techniques to the problem. In one example [51] the authors draw an analogy between graph-based SLAM and a mechanical spring-mass system. They assert that global poses correspond to unknown masses. The springs between the masses correspond to the relative transformations. The value of each spring's resting length corresponds with the initial estimate of the relative transformation, and the spring constant of each spring corresponds with the initial estimate of the transformation's uncertainty. They propose a type of gradient descent to sequentially adjust each pose based on the "forces" it experiences from the current best estimate of adjacent poses until the system settles down into a minimum energy configuration.

The approach of [52], dubbed multi-level relaxation (MLR), follows in a similar vein. They primarily differ from [51] in that they propose the network be relaxed at different levels of resolution in order to make computation more efficient. As late as 2006 some authors cited MLR as the current "state-of-the-art" in back-end optimization [54] . See the citations in [52] for additional references to relaxation-based approaches.

More recent back-end optimization algorithms tend to take a different approach due to advances in direct methods for solving sparse linear systems [6]. Equation (1.4) can be approximated by replacing the nonlinear function $h\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)$ with its first-order Taylor expansion. Taking the derivative, and setting the result equal to zero (see [6] for details) yields the normal equations

$$\mathbf{H}\boldsymbol{\rho}_{\boldsymbol{\Delta}} = \mathbf{d}, \tag{1.5}$$

where $\mathbf{H}$ is the information matrix associated with the probability distribution $\mathrm{p}(\boldsymbol{\rho})$, $\boldsymbol{\rho}_{\boldsymbol{\Delta}}$ is an incremental change in $\boldsymbol{\rho}$, and $\mathbf{d}$ is a constant vector.

The authors of [53, 54] use the observation that $\mathbf{H}$ is sparse for graph-based SLAM. Nonzero entries in the information matrix only occur along the block diagonal and in off-diagonal blocks corresponding to poses connected by a measured relative transformation. The authors then employ a variable elimination technique to reduce their initial graph to one containing fewer variables. Specifically, they remove all of the features, incorporating the information from pose-to-feature constraints into appropriate pose-to-pose constraints. The authors then solve the reduced system by matrix inversion or conjugate gradient descent.

The authors of [55–57] take the variable elimination technique further. They present a line of research that seeks to speed up the incremental optimization of $\boldsymbol{\rho}$ by making smart decisions about the order and method for eliminating variables from the graph. Instead of computing the information matrix, they focus attention on the Jacobian of $h\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)$ (and the similar function for landmarks) to improve accuracy and numerical stability [55]. Using QR-factorization they eliminate all of the variables in the graph to form the right triangular matrix $\mathbf{R}$, then solve the resulting structured, sparse linear system. The key to keeping this process efficient is keeping $\mathbf{R}$ as sparse as possible by choosing a good order in which variables are eliminated.

The authors of [55–57] also draw interesting connections between the matrices arising from linearization of (1.4) and probabilistic graphical models (Ch. 8 of [66]). Probabilistic graphical models include Bayesian networks, Markov random fields, and factor graphs. Their emphasis on graphical models culminates in [57] where the authors directly manipulate a tree similar to a Bayesian network [67] instead of the corresponding right triangular matrix $\mathbf{R}$. We also seek to draw insight from probabilistic graphical models and will discuss this further in Chapter 3.

The work presented in [58–63] represents another thread in back-end optimization research. As usual, the methods presented in [58–63] iteratively linearize Equation (1.4) and adjust $\boldsymbol{\rho}$. In [58] the authors introduce a variant of stochastic gradient descent to make the iterative optimization robust to local minima. They also present interesting results that suggest Equation (1.4) as an error metric is "not an adequate measure of graph quality." We will return to this observation in Chapter 3.

In [59] and [60] the authors extend the work in [58], making it converge faster to a more accurate solution by organizing the global poses into a tree structure and distributing rotational error more effectively. The primary contribution of [61] is similar to [55–57] in that they find an efficient solution to the linear system used in iterative optimization by choosing an appropriate ordering for variable elimination. In a contemporary paper [62] the authors modify the typical iterative optimization by calculating state changes on a manifold. They also make use of a hierarchy of graph-based maps with varying degrees of resolution.

Finally, in "A General Framework for Graph Optimization," dubbed g^{2}o, the authors take many of the above innovations and package them in an efficient C++ implementation [63]. The g^{2}o software is designed to be easily configured for any optimization problem that can be cast as a graphical network (e.g. graph-based SLAM and bundle adjustment). Regarding their results they state, "We present evaluations carried out on a large set of real-world and simulated data-sets; in all experiments g^{2}o offered a performance comparable with the state-of-the-art approaches and in several cases even outperformed them."

Of the foregoing papers, all but [60] treat the relative transformations as fixed constraints. In other words, after the front-end passes $\boldsymbol{\tau}_i^j$ to the back-end, that vector and its uncertainty are never altered. Recall that Lu and Milios said they "do not directly update the existing relations in the network when new observations are made. We simply add new relations to the network." This leads to a network that grows over time even if the robot remains within already explored territory. The authors of [60] allow the possibility of revisiting relative transformations, fusing new estimates with old ones. Their update of the transformations is equivalent to a Kalman update step where the measurement function is simply the identity function.

The extended Kalman filter (EKF) SLAM community has also produced a body of work relevant to our research. Several authors (e.g. [64], [65]; see also the references therein) divide the standard EKF SLAM map into a number of statistically independent local maps, or "sub-maps." Several factors motivate this approach to EKF SLAM. Consistency (as defined in Ch. 5 of [44]) is improved since smaller uncertainty in the robot pose relative to the local map leads to smaller linearization error. It is also well known that traditional EKF SLAM becomes computationally intractable as the number of mapped features grows. Sub-maps help to alleviate this problem by bounding the size of the filter state within each sub-map. Such approaches then retain an estimate of the relative transformations between the sub-maps.

Most work in EKF-SLAM using submaps is similar to graph-based SLAM, especially those graph-based SLAM methods that optimize the global position of features. EKF-SLAM arrives at this confluence from its original formulation as a completely feature-based, globally metric map [32]. Submapping approaches then shift the paradigm toward a globally topological and locally metric framework with features contained in the local maps. Graph-based SLAM began by using a globally topological, volumetric map [50]; subsequent works substituted the location of features relative to the local coordinate system defined by each pose. In either case, the globally metric map of feature positions is then obtained using an iterative nonlinear optimization routine.

The work in [27] (and subsequent publications [68–70]) presents an exception to this analogy between EKF SLAM sub-mapping and graph-based SLAM. We discuss [27, 68–70] further in Chapter 3. We note here that [27] introduces a distinction from the rest of the sub-mapping literature. They do not treat the individual sub-maps as statistically independent. Rather, when sub-maps are to be optimized with information gained at loop closure, that information is propagated back through the network of sub-maps using the property of *conditional* independence.

The algorithms in [51–65] all have at their core a paradigm similar to [50]. The relative transformations $\boldsymbol{\tau}_i^j$ from global pose $\boldsymbol{\rho}_i$ to global pose $\boldsymbol{\rho}_j$ are estimated by a front-end system. Once passed to the back-end, the transformations $\boldsymbol{\tau}_i^j$ are almost always treated as fixed constraints. The global poses are the focus of iterative optimization; the objective is to find the arrangement of global poses that best fits the fixed transformation estimates. Innovations over [50] have made this approach more efficient and more accurate, but the gist of the paradigm remains essentially the same.

Our work in Chapters 3 and 4 differs from the foregoing work on graph-based SLAM in that we seek to directly optimize the relative transformations in the back-end. Focusing on relative transformations leads to the same final solution as the preceding algorithms. However, it produces noticeably more accurate intermediate results. Our focus on the relative transformations grows out of our front-end navigation concept presented above.

1.3 Summary of This Dissertation's Contributions

To reiterate, the context of our research puts limits on the computational resources available while demanding timely information from large amounts of data. We seek to address this engineering tradeoff with our system concept, summarized in Figure 1.6. The proposed system offers many areas for further development. The work we present in this dissertation makes the following contributions in two main components of the overall system:

- Front-End Estimator
  - We present a principled approach for tuning a dynamic state estimator that propagates state estimates based on gyroscope measurements.
  - We draw insights from an observability analysis of the front-end estimator to explain the benefit of using an improved dynamic model for a multi-rotor helicopter.
  - We present new results that highlight the front-end estimator's resilience to error in the exteroceptive measurements used to update the filter's state estimates.
  - We also present new results that show how biases in the accelerometer measurements can be accurately estimated as states in the front-end estimator.
- Back-End Estimator
  - We develop the theoretical framework for a new approach to back-end optimization in graph-based SLAM. We dub our resulting algorithm BERT.
  - Using a simple optimization scenario we test a single iteration of BERT against one iteration of g^{2}o, a state-of-the-art implementation of the standard back-end optimization approach. Results show that BERT runs slightly faster while producing superior estimates of the relative transformations between the reference global poses.
  - In the same simple scenario we motivate repeated iterations of BERT and show that BERT and g^{2}o converge to the same final solution while BERT offers better intermediate results.

- We show that multiple iterations of BERT are justified after each iteration by the structure of the joint covariance matrix over all relative transformations and global poses.
- We develop the theoretical framework for extending BERT beyond the initial simple scenario to more general optimization problems.
- We show that extending BERT requires almost no additional computation beyond that required in the original special case.
- We analyze the application of loop closure information in a general loop closure scenario and show that this leads to increased correlation between the originating pose in the loop closure and the destination pose.
- We test the theory of extending BERT and discover that special consideration must be taken in computing the cross-covariance matrices. We propose and implement one solution and identify a likely path to a more sophisticated approach.

Chapter 2

Front-End Dynamic State Estimation For a Multi-Rotor Helicopter

In Chapter 1 we cited several sources (e.g. [1, 30]) to demonstrate the problem indoor flight vehicles have. The primary engineering tradeoff is meeting high computational demands with limited computing resources. The computational demands are driven by the nature of the unknown indoor environment and the dynamics of the vehicle.

The unknown, possibly complex environment necessitates using exteroceptive sensors such as stereo or RGB-D cameras that give a reasonably complete picture of the the vehicle's surroundings. Exteroceptive sensor information is also critical as the only source for directly updating the vehicle's estimate of its position and yaw angle. An IMU provides frequent and simple measurements of angular velocities and linear accelerations, but these only indirectly affect pose (i.e. position and orientation) estimates through their kinematic and dynamic relationships. Simplistic modeling of the vehicle may also degrade the benefit IMU measurements can provide.

The dynamics of multi-rotor helicopters are characterized by the vehicle's light weight, with small moments of inertia and very little aerodynamic drag. These attributes provide the agility desired for moving through confined indoor spaces, but they are agile almost to a fault. The vehicles' dynamics are fast and underdamped, therefore the automatic control must be capable of providing the damping the vehicles naturally lack.

Referring to our navigation concept in Figure 1.6, the front-end estimator has two sources of information for producing its estimates: the position and orientation information derived from exteroceptive data, and the measurements provided by the IMU. The authors of [2] present their quadrotor system stating that "one of the major challenges ... is estimating the position and velocity." The authors of [3] further emphasize that, "the estimated velocity is critical to damp the [quadrotor dynamics].” Estimating velocity is the primary concern for the immediate stabilization and control of multi-rotor helicopters.

As typified by the most advanced systems [1–3], the challenge of estimating the vehicle’s linear velocity is typically met by making position measurements as fast as possible. Frequent position measurements then correct the velocity estimates through their kinematic relationship. However, deriving position information from truly 3D exteroceptive sensors is not easy. To our knowledge, no multi-rotor helicopter using such sensors and operating in an unknown environment has been able to keep all computation onboard.

We have helped to develop an alternative and complementary approach for estimating the vehicle’s velocity. We adopt an improved dynamic model for the helicopter that allows us to get better information from the IMU measurements. This is an important feature given the frequency of the easily processed IMU data; it allows us to reduce the rate at which we require the more computationally expensive exteroceptive sensor processing.

In the remainder of this chapter we describe our contributions to developing a front-end observer that estimates the position, orientation, and linear velocity of the multi-rotor helicopter along with the biases inherent in the accelerometers and gyroscopes in the IMU. We first introduce the filter which is described in greater detail in our previously submitted collaborative work [71]. We then proceed to present our approach to tuning the filter. We conclude this chapter with an observability analysis of the filter and some relevant results.

2.1 Overview of the Front-End Filter

Figure 2.0, repeated from Chapter 1.1 for convenience, shows a schematic representation of the hexacopter and some important notation. The filter we develop for the hexacopter can be readily adapted to other multirotor helicopters. We have given an extended presentation of the vehicle dynamic model in [71] where we discuss several variations on the associated observer designs. We will present here the model’s highlights as well as an additional observer variation that includes estimation of two accelerometer biases.

We define the state vector to be

$$\mathbf{x} \triangleq \begin{bmatrix} f_{\mathrm{L}} & r_{\mathrm{L}} & d_{\mathrm{L}} & \phi & \theta & \psi & u & v & w & \beta_{\mathrm{i_b}} & \beta_{\mathrm{j_b}} & \beta_{\mathrm{k_b}} & \alpha_{\mathrm{i_b}} & \alpha_{\mathrm{j_b}} \end{bmatrix}^\top, \tag{2.1}$$

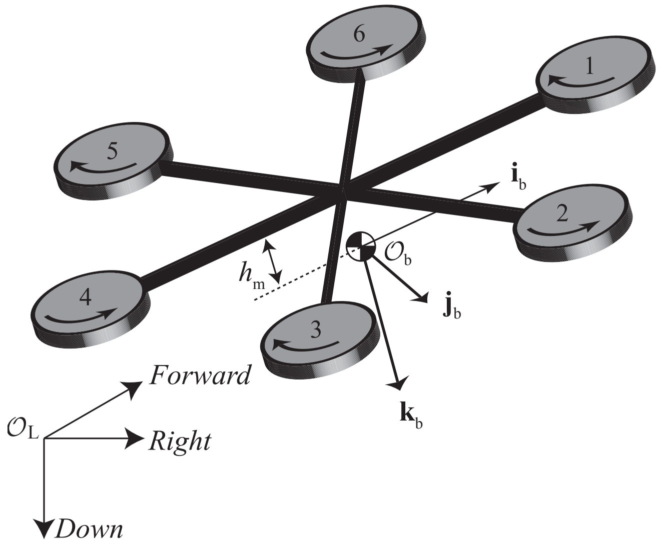

Figure 2.1: A schematic representation of a six rotor helicopter, or hexacopter. The six rotors alternately rotate in opposite directions (indicated by the curved arrows) to allow the vehicle to control its yaw. Roll and pitch, which are controlled by varying the relative motor speeds, produce the lateral accelerations of the vehicle. We use a body-fixed reference frame centered at $\mathcal{O}_b$ at a distance h_m below the vehicle. This right-handed reference frame has an $\mathbf{i}_b$ axis aligned with the vehicle's preferred forward direction and a $\mathbf{k}_b$ axis that aligns with gravity when the vehicle is at a perfect hover. We also use a local reference frame to define the vehicle's position. The right-handed local frame has an arbitrary origin, $\mathcal{O}_\mathrm{L}$, and an arbitrary heading with respect to a world reference frame aligned to north and east directions. However, the *Down* axis of the local frame is always aligned with gravity.

where f_L, r_L, and d_L represent the vehicle's forward, right, and down displacement from a local reference frame; ϕ, θ, and ψ are the roll, pitch, and yaw angles relating the local reference frame to the body-fixed reference frame; u, v, and w represent the hexacopter's linear velocity resolved in the body $\mathbf{i}_\mathrm{b}$, $\mathbf{j}_\mathrm{b}$, and $\mathbf{k}_\mathrm{b}$ axes; $\beta_{\mathrm{i}_\mathrm{b}}$, $\beta_{\mathrm{j}_\mathrm{b}}$, and $\beta_{\mathrm{k}_\mathrm{b}}$ are the biases in gyroscopes measuring angular velocity about the body $\mathbf{i}_\mathrm{b}$, $\mathbf{j}_\mathrm{b}$, and $\mathbf{k}_\mathrm{b}$ axes; and $\alpha_{\mathrm{i}_\mathrm{b}}$ and $\alpha_{\mathrm{j}_\mathrm{b}}$ are biases in the accelerometers aligned with the $\mathbf{i}_\mathrm{b}$ and $\mathbf{j}_\mathrm{b}$ axes. The rotation matrix from the body-fixed to the local reference frame is

$$\mathbf{R}_{\mathrm{b}}^{\mathrm{L}} \triangleq \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.2}$$

$$\triangleq \begin{bmatrix} \mathrm{c}_\theta \mathrm{c}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{c}_\psi - \mathrm{c}_\phi \mathrm{s}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi \\ \mathrm{c}_\theta \mathrm{s}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{s}_\psi + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{s}_\phi \mathrm{c}_\theta & \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix},$$

where $\mathrm{c}_\theta \triangleq \cos(\theta)$, $\mathrm{s}_\theta \triangleq \sin(\theta)$, etc.

The state propagation and measurement equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{2.3}$$

$$y_n = h_n(\mathbf{x}), \tag{2.4}$$

are defined below. The vector $\mathbf{u}$ represents the inputs to the model. In the observer we use gyroscope measurements to drive the prediction of state estimates. Therefore,

$$\mathbf{u} \triangleq \begin{bmatrix} \gamma_{\mathrm{i_b}} \\ \gamma_{\mathrm{j_b}} \\ \gamma_{\mathrm{k_b}} \end{bmatrix} = \begin{bmatrix} p + \beta_{\mathrm{i_b}} \\ q + \beta_{\mathrm{j_b}} \\ r + \beta_{\mathrm{k_b}} \end{bmatrix}, \tag{2.5}$$

where $\gamma_{\mathrm{i_b}}$, $\gamma_{\mathrm{j_b}}$, and $\gamma_{\mathrm{k_b}}$ are the gyroscope measurements about the subscripted axes, and p, q, and r are the actual rotation rates. We will show in the subsequent section of this chapter that using sensors to drive the estimation model allows for easy tuning of the process noise. This sensor driven model also allows us to avoid deriving some complicated elements of the vehicle model such as the relationship between individual motors speeds, moments of inertia, and angular accelerations.

We can decompose Eq. (2.3) into subsystems. We model propagation of bias states as a random walk and define the rest of Eq. (2.3) as

$$\begin{bmatrix} \dot{f}_\mathrm{L} \\ \dot{r}_\mathrm{L} \\ \dot{d}_\mathrm{L} \end{bmatrix} = \mathbf{R}_\mathrm{b}^\mathrm{L} \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{2.6}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \tag{2.7}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -g\sin\theta + (vr - wq) - \frac{F_\mathrm{di}}{m} \\ g\sin\phi\cos\theta + (wp - ur) - \frac{F_\mathrm{dj}}{m} \\ g\cos\phi\cos\theta + (uq - vp) - \frac{T}{m} \end{bmatrix}, \tag{2.8}$$

where g is the acceleration due to gravity, m is the vehicle's mass, and T is the total thrust generated collectively by the vehicle's motors. The variables F_di and F_dj are forces along the body-fixed $\mathbf{i}_\mathrm{b}$ and $\mathbf{j}_\mathrm{b}$ axes arising from drag experienced by the rotors. These drag terms, first described in detail by [72], represent the improvement in the dynamic model that enables better use of IMU data. While seemingly a minor change, correctly characterizing these drag forces significantly influences our interpretation of accelerometer measurements.

The $\mathbf{i}_\mathrm{b}$ and $\mathbf{j}_\mathrm{b}$ axis accelerometers measure the specific forces (i.e. total force minus the effect of gravity) associated with $\dot{u}$ and $\dot{v}$. The specific forces modeled in (2.8) include Coriolis terms and F_di and F_dj. The Coriolis terms are relatively small for indoor flight, therefore we model the $\mathbf{i}_\mathrm{b}$ and $\mathbf{j}_\mathrm{b}$ axis accelerometer measurements respectively as

$$h_1(\mathbf{x}) \triangleq -\frac{F_\mathrm{di}}{m} = -\frac{\mu}{m}u, \tag{2.9}$$

$$h_2(\mathbf{x}) \triangleq -\frac{F_\mathrm{dj}}{m} = -\frac{\mu}{m}v, \tag{2.10}$$

where μ is an aerodynamic term that translates linear velocity into a drag force as described in [71] and [72]. The important point here is that these accelerometers offer a direct, scaled measurement of the corresponding velocity components.

We implement the observer as an extended Kalman filter using the preceding definitions for the components of (2.3) and (2.4) along with additional measurement models for the position and orientation measurements derived from the exteroceptive sensor. Reference [2] includes terms similar to $-\frac{\mu}{m}u$ and $-\frac{\mu}{m}v$ in the dynamic model used in their filter. They give no physical reason for the presence of these terms and explain them as an expediency to prevent unbounded growth in the predicted velocity. They do not appear to model accelerometers as measuring these drag terms. Reference [1] lumps these drag forces into a generic term in their filter described in Chapter 1.1.1, Equation (1.1); the term is used to model all aerodynamic disturbances which they assume are slowly varying.

We can treat μ as a constant for nominal indoor flight conditions. We mention in [71] that μ can be included in the observer state to estimate its value. An accurate value for μ can also be estimated directly given knowledge of the velocity components u and v, such as from a motion capture system. Measured values of u and v can be used to find a least-squares fit of (2.9) and (2.10) to the actual accelerometer measurements. Using this approach to estimate μ, we demonstrate in Figure 2.1 the agreement between actual accelerometer measurements and the measurement models defined in (2.9) and (2.10). We take the quality of the fit do be sufficient validation of the model and the assumptions made in its derivation.

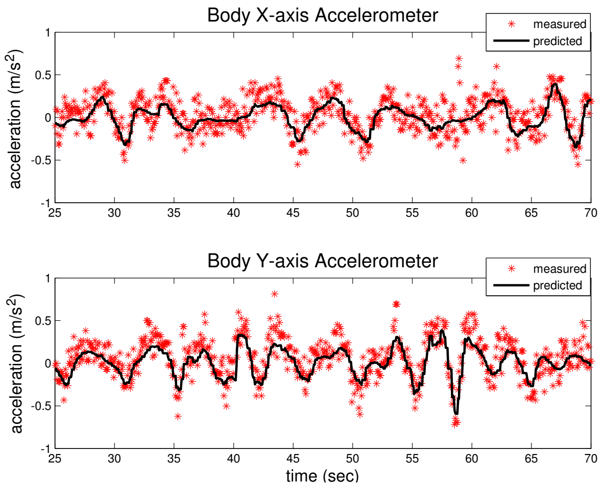

Figure 2.2: Actual accelerometer measurements for a nominal indoor flight plotted against those predicted by (2.9) and (2.10).

Using this model for accelerometer measurements significantly improves estimates of the velocities u and v. Roll and pitch estimates also especially benefit because of their causal influence on the hexacopter's lateral acceleration. We leave the presentation of most of these results to [71] because they are primarily the contribution of that article's lead author. In Section 2.4 we do present some results not included in [71].

The improved estimates are facilitated by our approach to tuning the filter, and they can be explained analytically by examining the observability of (2.3) and (2.4). We proceed with these two topics next in that order. We then present some results and concluding thoughts for the chapter.

2.2 Tuning The Sensor Driven Filter

As mentioned in the preceding section, the filter is designed to propagate its state estimates forward in time based on measured angular velocities. This is opposed to using the actual control inputs sent to the vehicle. In the latter case one must model the vehicle from the reception of those inputs on through to the effect they have on the states. Consider our navigation concept in Figure 1.6. The position controller sends control inputs to the hexacopter's Onboard Controller. From those desired values the Onboard controller determines desired motor speeds for each of the six motors. The cumulative effect of the motors then creates angular accelerations that depend on the vehicle's physical characteristics (e.g. mass, moments of inertia, etc.). This leads to an expanded state vector that includes angular rates, and the gyroscope measurements would be used in an update step to the filter.

Using the gyroscope measurements to instead drive propagation of the other states simplifies the vehicle model in the observer. It also leads to an accurate approach to model the uncertainty in that propagation process. We will first offer just a few thoughts on the proper approach to filter tuning. We will then derive the method used to tune the process uncertainty.

2.2.1 Tuning Philosophy

As in any design process, guess and check should be a means of last resort. If a principled approach can be taken without too much effort, then it should be preferred over a heuristic. We also observe that uncertainty, or noise, is most accurately and easily modeled where it actually occurs. We can often model the error in a raw measurement, for example, with a simple (e.g. mutually independent) joint Gaussian distribution.

We will use a qualitative description of visual odometry to illustrate what we mean.[1] Visual odometry begins with comparing point features detected in two images of the same scene taken from different poses. Identifying the point features in an image is the place where error and uncertainty originally enter the problem. The image is quantized into pixels and suffers from effects like blurring or the lens distortion. The feature detector itself might also be imprecise. These effects combine to make the feature detector identify a point on the image plane that does not agree with the location an ideal pinhole camera model would suggest.

If raw feature locations in the image plane were the measurement of interest, modeling the uncertainty would be pretty straightforward. If the camera is reasonably well calibrated we could probably assume the error in the features are mutually independent. We could also probably safely model the error for each feature as an independent joint Gaussian distribution over the x and y pixel locations.

However, the filter that will use the visual odometry output expects to receive a measurement of the camera's change in position and orientation. To develop that measurement from raw feature locations we must[2] use the camera's projective geometry to estimate the detected features' 3D positions. We then need to find the best rigid body transformation that explains the change in perspective. The simple uncertainty in individual feature locations must now be mapped through a complex process to provide the uncertainty in the resulting measurement of the camera's change in pose.

At this point, one might be justified in throwing in the towel and adopting a heuristic. For example, one might approximate the visual odometry covariance by hand tuning a 6x6 diagonal matrix and dividing it by the number of corresponding features used to develop the rigid body transformation estimate. However, the result could certainly be more accurate if we developed a principled approach to mapping the underlying uncertainty from the raw

[1]This example is motivated by the ongoing work of Robert Leishman

[2]We assume that incorrect feature associations from one image to the next are eliminated by an outlier rejection step, e.g. RANSAC [73].

measurements into the final visual odometry output. In the heuristic just offered, the 6 DOF elements of the visual odometry measurement are not really independent as a diagonal covariance matrix implies.

We map uncertainty from one space into another space using the Jacobian of the function that transforms one random variable into another. Intuitively, the Jacobian describes how perturbing the input variables leads to perturbations of the output. A covariance matrix can be thought of as a belief about how much the estimate is perturbed from the truth. Therefore, we should be able to use the Jacobian to map that belief. We now proceed to derive how that works for the filter presented in this chapter.

2.2.2 Tuning Approach

We will assume the additive error in each gyroscope measurement is zero mean and Gaussian distributed. We can test this by collecting gyroscope measurements when the rotation rate is known (e.g. zero); this information is also available in the gyroscope's manufacturing specifications. We will assume for this derivation that there are no biases in the gyroscopes or accelerometers. This reduces the filter state to

$$\mathbf{x} = \left[\begin{array}{ccccccccc} f_{\mathrm{L}} & r_{\mathrm{L}} & d_{\mathrm{L}} & \phi & \theta & \psi & u & v & w \end{array} \right]^{\top}. \tag{2.11}$$

At the end of the derivation we will discuss how this assumption is relaxed.

Error in the gyroscope measurements is the primary source of uncertainty in propagating the state estimates. The states evolve according to Equations (2.6) - (2.8). In particular, there are no approximations or modeling errors at all in Equations (2.6) and (2.7). We have also shown the quality of the dynamic model in Equation (2.8) by demonstrating the model's agreement with measured accelerometer data. Therefore, in the following presentation we will assume that *all* uncertainty in propagating the state estimates can be accounted for in the gyroscope measurement error. We will relax this assumption at the end of the derivation.

We will adapt some of the notation from the previous section to make the following presentation easier to follow and self contained. We define the estimated states to be the true states plus some error; i.e.

$$\hat{\mathbf{x}} \triangleq \mathbf{x} + \tilde{\mathbf{x}}.$$

Similarly, we define the input from the gyroscope measurements to be the true rotation rates plus some error; i.e.

$$\hat{\mathbf{u}} \triangleq \mathbf{u} + \tilde{\mathbf{u}},$$

where $\mathbf{u} = [p\ q\ r]^\top$ and $\tilde{\mathbf{u}} \sim \mathcal{N}(0, \mathbf{G})$. The diagonal covariance matrix $\mathbf{G}$ represents the independent error parameters for each gyroscope.

The estimated states evolve according to

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= f\left(\hat{\mathbf{x}}, \hat{\mathbf{u}}\right) \\ &\approx \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\hat{\mathbf{u}}, \end{aligned}$$

where

$$\begin{aligned} \mathbf{A} &\triangleq \left.\frac{\partial f\left(\mathbf{x}, \mathbf{u}\right)}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}, \hat{\mathbf{u}}}, \\ \mathbf{B} &\triangleq \left.\frac{\partial f\left(\mathbf{x}, \mathbf{u}\right)}{\partial \mathbf{u}}\right|_{\hat{\mathbf{x}}, \hat{\mathbf{u}}}. \end{aligned}$$

Analogous equations can be written for the time evolution of the true states. Therefore, the evolution of the estimation error can be written

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \dot{\hat{\mathbf{x}}} - \dot{\mathbf{x}} \\ &= \left(\mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\hat{\mathbf{u}}\right) - \left(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}\right) \\ &= \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\tilde{\mathbf{u}}. \end{aligned}$$

The solution to this differential equation is given by

$$\tilde{\mathbf{x}} = \mathrm{e}^{\mathbf{A}t}\tilde{\mathbf{x}}_0 + \int_0^t \mathrm{e}^{\mathbf{A}(t-\tau)}\mathbf{B}\tilde{\mathbf{u}}(\tau)d\tau. \tag{2.12}$$

The uncertainty in the state estimates is defined to be

$$\mathbf{P} \overset{\triangle}{=} \mathbb{E}\left\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top\right\},$$

where $\mathbb{E}\{\cdot\}$ denotes the expectation of a random variable or vector. The state estimate uncertainty evolves through time according to

$$\begin{aligned}
\dot{\mathbf{P}} &= \frac{d}{dt}\mathbb{E}\left\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top\right\} \\
&= \mathbb{E}\left\{\dot{\tilde{\mathbf{x}}}\tilde{\mathbf{x}}^\top + \tilde{\mathbf{x}}\dot{\tilde{\mathbf{x}}}^\top\right\} \\
&= \mathbb{E}\left\{\left(\mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\tilde{\mathbf{u}}\right)\tilde{\mathbf{x}}^\top + \tilde{\mathbf{x}}\left(\mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\tilde{\mathbf{u}}\right)^\top\right\} \\
&= \mathbb{E}\left\{\mathbf{A}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top + \mathbf{B}\tilde{\mathbf{u}}\tilde{\mathbf{x}}^\top + \tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top\mathbf{A}^\top + \tilde{\mathbf{x}}\tilde{\mathbf{u}}^\top\mathbf{B}^\top\right\}
\end{aligned}$$

which, when we distribute the expectation operator, gives

$$\dot{\mathbf{P}} = \mathbf{A}\mathbf{P} + \mathbb{E}\left\{\mathbf{B}\tilde{\mathbf{u}}\tilde{\mathbf{x}}^\top\right\} + \mathbf{P}\mathbf{A}^\top + \mathbb{E}\left\{\tilde{\mathbf{x}}\tilde{\mathbf{u}}^\top\mathbf{B}^\top\right\}.$$

We need an expression for $\mathbb{E}\left\{\tilde{\mathbf{x}}\tilde{\mathbf{u}}^\top\mathbf{B}^\top\right\}$ to complete the derivation. Substituting Equation (2.12) into $\mathbb{E}\left\{\tilde{\mathbf{x}}\tilde{\mathbf{u}}^\top\mathbf{B}^\top\right\}$ gives

$$\begin{aligned}
\mathbb{E}\left\{\tilde{\mathbf{x}}\tilde{\mathbf{u}}^\top\mathbf{B}^\top\right\} &= \mathbb{E}\left\{\left(\mathrm{e}^{\mathbf{A}t}\tilde{\mathbf{x}}_0 + \int_0^t \mathrm{e}^{\mathbf{A}(t-\tau)}\mathbf{B}\tilde{\mathbf{u}}(\tau)d\tau\right)\tilde{\mathbf{u}}^\top\mathbf{B}^\top\right\} \\
&= \mathbb{E}\left\{\mathrm{e}^{\mathbf{A}t}\tilde{\mathbf{x}}_0\tilde{\mathbf{u}}^\top\mathbf{B}^\top + \int_0^t \mathrm{e}^{\mathbf{A}(t-\tau)}\mathbf{B}\tilde{\mathbf{u}}(\tau)\tilde{\mathbf{u}}(\tau)^\top\mathbf{B}^\top d\tau\right\} \\
&= \int_0^t \mathrm{e}^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{G}\mathbf{B}^\top\delta(t-\tau)d\tau \\
&= \frac{1}{2}\mathbf{B}\mathbf{G}\mathbf{B}^\top,
\end{aligned}$$

where the $\frac{1}{2}$ comes from integrating over only half the volume of the delta function. The product $\frac{1}{2}\mathbf{B}\mathbf{G}\mathbf{B}^\top$ is symmetric, therefore $\mathbb{E}\left\{\mathbf{B}\tilde{\mathbf{u}}\tilde{\mathbf{x}}^\top\right\} = \mathbb{E}\left\{\left(\tilde{\mathbf{x}}\tilde{\mathbf{u}}^\top\mathbf{B}^\top\right)^\top\right\} = \frac{1}{2}\mathbf{B}\mathbf{G}\mathbf{B}^\top$. We can now write the time propagation of the estimation uncertainty as

$$\dot{\mathbf{P}} = \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^\top + \mathbf{B}\mathbf{G}\mathbf{B}^\top.$$

The $\mathbf{B}\mathbf{G}\mathbf{B}^\top$ term maps the simple, diagonal covariance $\mathbf{G}$ into a more complex uncertainty that acknowledges the correlation between various state elements. This was especially useful in our development of the filter presented in this chapter. In earlier attempts to leverage the improved dynamic model we designed three independent filters[3] [74]. The state vectors for each filter consisted of

$$\mathbf{x}_{\text{long}} = \begin{bmatrix} u \\ \theta \end{bmatrix}, \quad \mathbf{x}_{\text{lat}} = \begin{bmatrix} v \\ \phi \end{bmatrix}, \quad \mathbf{x}_{\text{3dof}} = \begin{bmatrix} n \\ e \\ \psi \end{bmatrix}, \tag{2.13}$$

where n and e respectively designate the north and east position of the vehicle with respect to a global reference frame. The propagation equations for these states were essentially the same as given above in Equations (2.6) - (2.8).[4] The decision to use the three separate filters was motivated by a desire to make it easier to hand tune the filters.

However, estimating the states in 3 separate filters is an approximation because these states are not actually independent. In order to extend and improve on [74] using the filer defined in this chapter we need to tune a 15 x 15 process uncertainty matrix. To be accurate we should account for the intuitive correlations between several of the state estimates. Using $\mathbf{B}\mathbf{G}\mathbf{B}^\top$ in the state propagation step allows the single, large filter to provide accurate results with almost no heuristic tuning.

We mentioned a few assumptions in the beginning of the above derivation that we now address. First, we assumed that the IMU sensors were unbiased. Relaxing this assumption would obviously change the above derivation by changing $\tilde{\mathbf{u}} \sim \mathcal{N}(0, \mathbf{G})$. However, we show

[3]Most of the work in [74] was done by Robert Leishman.

[4]The only differences were: (1) using a global instead of a local reference frame for position states, and (2) using a term for μ that varied slightly based on the current motor speeds.

below that the gyroscope and accelerometer biases are observable. We can calibrate and remove gyroscope biases just before flight, and because they remain well estimated thereafter we can continue to subtract out their effect.

We also made the mild assumption that all process uncertainty stems from error in the gyroscope measurements. We address this assumption by adding a hand tuned component, $\mathbf{Q}$, to the process uncertainty such that

$$\dot{\mathbf{P}} = \mathbf{AP} + \mathbf{PA}^\top + \mathbf{BGB}^\top + \mathbf{Q}. \tag{2.14}$$

The matrix $\mathbf{Q}$ is easy to tune because most of the process uncertainty is in fact correctly represented in $\mathbf{BGB}^\top$. We set $\mathbf{Q}$ equal to the zero matrix and only adjust a few elements along its diagonal. The most important function of $\mathbf{Q}$ is to model the random walk evolution of the IMU biases.

2.3 Analyzing the Filter

We now proceed with the observability analysis to inform our understanding of the filter's performance. Since the system is nonlinear the observability analysis follows the approach and notation of [75, Chapter 7]. We introduce that notation and theory in Appendix A and include a small example there.

For the observability analysis it is convenient to rewrite (2.3) as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \sum_{j=1}^{3} u_j \mathbf{g}_j(\mathbf{x}), \tag{2.15}$$

with $\mathbf{f},\ \mathbf{g}_j \in V(X)$. In addition to the accelerometer measurements given in (2.9) and (2.10), we assume an appropriate algorithm (e.g. visual odometry) provides the measurements

$$h_3 \overset{\triangle}{=} f_\mathrm{L}, \tag{2.16}$$

$$h_4 \overset{\triangle}{=} r_\mathrm{L}, \tag{2.17}$$

$$h_5 \overset{\triangle}{=} d_\mathrm{L}, \tag{2.18}$$

$$h_6 \overset{\triangle}{=} \psi, \tag{2.19}$$

where $h_n \in S(X)$. We have chosen here to omit the roll and pitch measurements that could be provided by visual odometry in order to highlight the information afforded by the improved model.

It is easy to see that the six row vectors of the form $\mathbf{d}h_n|_{\mathbf{x}_0}$ are all linearly independent. We next examine the six additional vectors of the form $\mathbf{d}L_{\mathbf{f}}h_n|_{\mathbf{x}_0}$. To keep expressions compact we represent a vector of zeros of length m as $\mathbf{0}_m$ to make the expressions more compact. We also use the element-wise definition of $\mathbf{R}_{\mathrm{b}}^{\mathrm{L}}$ from (2.2). We can then write

$$\mathbf{d}L_{\mathbf{f}}h_1 = \begin{bmatrix} \mathbf{0}_3 \\ 0 \\ \frac{g\mu}{m}\mathrm{c}_\theta \\ 0 \\ \frac{\mu^2}{m^2} \\ \frac{\mu}{m}\beta_{\mathrm{k_b}} \\ -\frac{\mu}{m}\beta_{\mathrm{j_b}} \\ 0 \\ -\frac{\mu}{m}w \\ \frac{\mu}{m}v \\ \mathbf{0}_2 \end{bmatrix}^\top, \qquad \mathbf{d}L_{\mathbf{f}}h_2 = \begin{bmatrix} \mathbf{0}_3 \\ -\frac{g\mu}{m}\mathrm{c}_\phi\mathrm{c}_\theta \\ \frac{g\mu}{m}\mathrm{s}_\phi\mathrm{s}_\theta \\ 0 \\ -\frac{\mu}{m}\beta_{\mathrm{k_b}} \\ \frac{\mu^2}{m^2} \\ \frac{\mu}{m}\beta_{\mathrm{i_b}} \\ \frac{\mu}{m}w \\ 0 \\ -\frac{\mu}{m}u \\ \mathbf{0}_2 \end{bmatrix}^\top,$$

$$\mathbf{d}L_{\mathbf{f}}h_3 = \begin{bmatrix} \mathbf{0}_3 \\ v\frac{\partial r_{12}}{\partial \phi} + w\frac{\partial r_{13}}{\partial \phi} \\ u\frac{\partial r_{11}}{\partial \theta} + v\frac{\partial r_{12}}{\partial \theta} + w\frac{\partial r_{13}}{\partial \theta} \\ u\frac{\partial r_{11}}{\partial \psi} + v\frac{\partial r_{12}}{\partial \psi} + w\frac{\partial r_{13}}{\partial \psi} \\ r_{11} \\ r_{12} \\ r_{13} \\ \mathbf{0}_5 \end{bmatrix}^\top, \qquad \mathbf{d}L_{\mathbf{f}}h_4 = \begin{bmatrix} \mathbf{0}_3 \\ v\frac{\partial r_{22}}{\partial \phi} + w\frac{\partial r_{23}}{\partial \phi} \\ u\frac{\partial r_{21}}{\partial \theta} + v\frac{\partial r_{22}}{\partial \theta} + w\frac{\partial r_{23}}{\partial \theta} \\ u\frac{\partial r_{21}}{\partial \psi} + v\frac{\partial r_{22}}{\partial \psi} + w\frac{\partial r_{23}}{\partial \psi} \\ r_{21} \\ r_{22} \\ r_{23} \\ \mathbf{0}_5 \end{bmatrix}^\top,$$

$$\mathbf{d}L_{\mathbf{f}}h_5 = \begin{bmatrix} \mathbf{0}_3 \\ v\frac{\partial r_{32}}{\partial \phi} + w\frac{\partial r_{33}}{\partial \phi} \\ u\frac{\partial r_{31}}{\partial \theta} + v\frac{\partial r_{32}}{\partial \theta} + w\frac{\partial r_{33}}{\partial \theta} \\ 0 \\ r_{31} \\ r_{32} \\ r_{33} \\ \mathbf{0}_5 \end{bmatrix}^{\top}, \qquad \mathbf{d}L_{\mathbf{f}}h_6 = \begin{bmatrix} \mathbf{0}_3 \\ -\frac{\mathrm{c}_\phi}{\mathrm{c}_\theta}\beta_{\mathrm{j_b}} + \frac{\mathrm{s}_\phi}{\mathrm{c}_\theta}\beta_{\mathrm{k_b}} \\ -\frac{\mathrm{s}_\phi \mathrm{s}_\theta}{(\mathrm{c}_\theta)^2}\beta_{\mathrm{j_b}} - \frac{\mathrm{c}_\phi \mathrm{s}_\theta}{(\mathrm{c}_\theta)^2}\beta_{\mathrm{k_b}} \\ \mathbf{0}_5 \\ -\frac{\mathrm{s}_\phi}{\mathrm{c}_\theta} \\ -\frac{\mathrm{c}_\phi}{\mathrm{c}_\theta} \\ \mathbf{0}_2 \end{bmatrix}^{\top}.$$

We combine the vectors obtained so far into a single observability matrix, $\mathbf{O}_{\mathrm{M}}$, and pick a point in the state space to evaluate the matrix elements. For our vehicle, $\frac{\mu}{m} \approx 0.28$ and gyroscope biases are on the order of 10^{-3} at the beginning of a flight. We assume ϕ and θ are small such that $\cos(\phi) \approx 1$, $\sin(\phi) \approx \phi$, etc. We also assume u, v, and w are at most 0.25 meters per second. These assumptions satisfy the needs of the autonomous hexacopter. High velocities and large angles lead to large accelerations, degrading the exteroceptive sensor measurements the vehicle uses to interact with the environment.

For the values and assumptions given above we find that

$$\mathbf{O}_{\mathrm{M}} \triangleq \dots$$

$$\dots \begin{bmatrix} \mathbf{d}h_1 \\ \mathbf{d}h_2 \\ \mathbf{d}L_{\mathbf{f}}h_1 \\ \mathbf{d}L_{\mathbf{f}}h_2 \\ \mathbf{d}h_3 \\ \mathbf{d}h_4 \\ \mathbf{d}h_5 \\ \mathbf{d}h_6 \\ \mathbf{d}L_{\mathbf{f}}h_3 \\ \mathbf{d}L_{\mathbf{f}}h_4 \\ \mathbf{d}L_{\mathbf{f}}h_5 \\ \mathbf{d}L_{\mathbf{f}}h_6 \end{bmatrix} \approx \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\mu}{m} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\mu}{m} & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{g\mu}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{g\mu}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w\mathrm{s}_\psi & w\mathrm{c}_\psi & o_1 & \mathrm{c}_\psi & -\mathrm{s}_\psi & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -w\mathrm{c}_\psi & w\mathrm{s}_\psi & o_2 & \mathrm{s}_\psi & \mathrm{c}_\psi & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & v & -u & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}, \quad (2.20)$$

where $o_1 \triangleq -u\mathrm{s}_\psi - v\mathrm{c}_\psi$ and $o_2 \triangleq u\mathrm{c}_\psi - v\mathrm{s}_\psi$. The most significant entries in $\mathbf{O}_\mathrm{M}$ will be between approximately 0.3 and 3.0. We have grouped rows in $\mathbf{O}_\mathrm{M}$ according to their corresponding sensor type. The first four rows derive from accelerometer measurements, and the remaining rows depend on the exteroceptive sensor algorithm. In the sequel we will indicate a single element of the matrix with the notation $\mathbf{O}_\mathrm{M}(\mathrm{row}, \mathrm{column})$

We can see that the improved model for accelerometer measurements leads to significant entries at $\mathbf{O}_\mathrm{M}(1,7)$ and $\mathbf{O}_\mathrm{M}(2,8)$. These columns correspond to u and v, the two components of velocity in the plane of the rotors. Without these entries arising from the improved model, observing u and v would totally rely upon the position measurements h_3 and h_4, as indicated by the nonzero elements at $\mathbf{O}_\mathrm{M}(9,7)$, $\mathbf{O}_\mathrm{M}(9,8)$, $\mathbf{O}_\mathrm{M}(10,7)$, and $\mathbf{O}_\mathrm{M}(10,8)$.

We also note that the improved model for accelerometer measurements leads to significant values at $\mathbf{O}_\mathrm{M}(3,5)$ and $\mathbf{O}_\mathrm{M}(4,4)$. These columns correspond to ϕ and θ. Having significant entries at $\mathbf{O}_\mathrm{M}(3,5)$ and $\mathbf{O}_\mathrm{M}(4,4)$ supports our assertion in Section 2.1, i.e. that the improved dynamic model ought to improve attitude estimates because deflections from hover cause the accelerations and therefore the velocities that the accelerometers measure.

The 10th and 11th columns in (2.20) lack significant entries; these columns correspond to the gyroscope bias states $\beta_{\mathrm{i_b}}$ and $\beta_{\mathrm{j_b}}$. This deficiency leads us to consider other candidate vectors for $\mathbf{O}_\mathrm{M}$ that would have significant entries in these positions. All vectors of the form $\mathbf{d}L_{\mathbf{g}_j}h_n$ have only zero in these columns. We therefore consider second order derivative vectors of the form $\mathbf{d}L_{\mathbf{g}_j}L_\mathbf{f}h_n$ or $\mathbf{d}L_\mathbf{f}L_\mathbf{f}h_n$.

The only likely candidates are $\mathbf{d}L_\mathbf{f}L_\mathbf{f}h_1$ and $\mathbf{d}L_\mathbf{f}L_\mathbf{f}h_2$. Evaluating at the same point in the state space as before gives

$$\begin{aligned}
\mathbf{d}L_\mathbf{f}L_\mathbf{f}h_1 &\approx \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{g\mu}{m} & 0 & 0 & 0 \end{bmatrix}, \\
\mathbf{d}L_\mathbf{f}L_\mathbf{f}h_2 &\approx \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{g\mu}{m} & 0 & 0 & 0 & 0 \end{bmatrix}.
\end{aligned} \tag{2.21}$$

The term $\frac{g\mu}{m} \approx 2.8$ provides the necessary entries in the 10th and 11th columns. When we augment $\mathbf{O}_\mathrm{M}$ with these two rows we find a full rank, well conditioned matrix.

In the preceding development we have highlighted the important structure of $\mathbf{O}_\mathrm{M}$ by evaluating at a particular point in the state space. To verify observability for nominal indoor flight conditions we evaluate $\mathbf{O}_{\mathrm{M}}$ over time for an entire flight. We use a motion capture system to measure the vehicle's position and orientation during the flight. We also use this high rate ($\approx$ 200 Hz) information to estimate the vehicle's velocity in the body-fixed reference frame. We make only one approximation, that gyro biases are zero, becasue these biases are calibrated immediately before flight. In Figure 2.2 we plot the condition number of $\mathbf{O}_{\mathrm{M}}$ over a typical flight and note that $\mathbf{O}_{\mathrm{M}}$ is full rank for the duration.

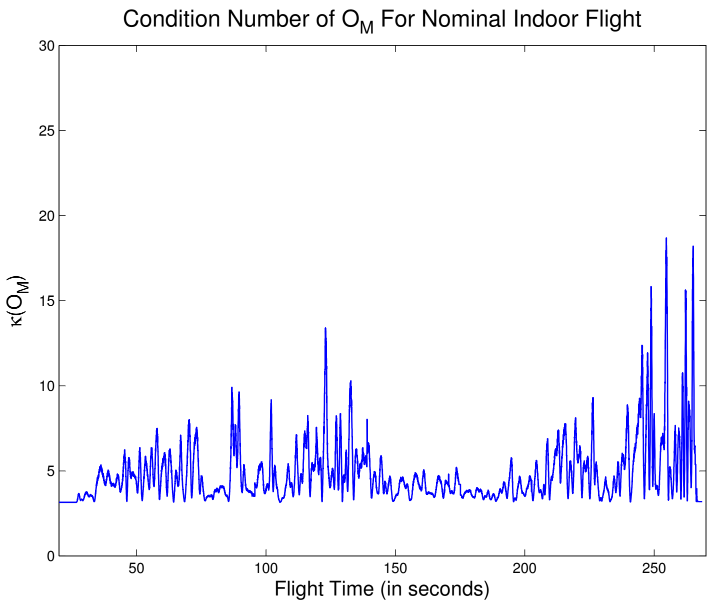

Figure 2.3: This figure shows the condition number of the observability matrix as a function of time for a nominal indoor flight. The condition number is defined as the ratio of the largest to the smallest singular value of $\mathbf{O}_{\mathrm{M}}$. This figure represents the condition number for $\mathbf{O}_{\mathrm{M}}$ without approximations other than neglecting terms multiplied by gyroscope biases.

2.4 Results

As mentioned earlier, we use the model described above in a continuous-discrete extended Kalman filter [76, Ch. 8, Algorithm 2]. IMU data arrive at approximately 40 Hz. Upon receiving IMU data the state estimates are propagated forward with measurements from the three-axis MEMs gyroscopes using numerical integration of (2.6) – (2.8). After propagation the accelerometer measurements are applied using (2.9) and (2.10) in a typical Kalman update step.

The filter covariance $\mathbf{P}$ is propagated using numerical integration of

$$\dot{\mathbf{P}} = \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^\top + \mathbf{B}\mathbf{G}\mathbf{B}^\top + \mathbf{Q}, \tag{2.22}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the Jacobians of (2.3) with respect to the states and the inputs, respectively; $\mathbf{G}$ is a diagonal covariance matrix representing the uncertainty in the gyroscope inputs; and $\mathbf{Q}$ is a hand-tuned, diagonal matrix primarily used to model the random walk for the bias states (i.e. most elements are set to 0). Since gyroscope measurements drive state propagation, we can measure the noise characteristics that define $\mathbf{G}$. Mapping that uncertainty through the Jacobian $\mathbf{B}$ helps make the filter accurate and easy to tune.

To be consistent with the foregoing observability analysis, the filter receives exteroceptive measurements (2.16) – (2.19) giving the relative position and heading of the hexacopter. These measurements would be provided from a vision-based algorithm such as described in [77] or [78]. However, in these results we intend to compare the characteristics of our state estimation approach relative to approaches common in the literature. We therefore synthesize relative state measurements to allow easy adjustment of the update rate and noise characteristics. This is done using information from a motion capture system.

For the results presented below, we sample motion capture data at 5 Hz and transform it into a local reference frame. We randomly delay the measurements with a mean delay of 250 ms to simulate the time needed to process vision data. We also add various levels of independent Gaussian noise to each of the relative pose measurements. While there are certainly artifacts of real vision processing not captured by this approach, we assume it is sufficient for a comparative analysis.

For that comparison, we have implemented an observer that seems common among researches using multirotor helicopters. Referring to Figure 1.6, a low-level observer is generally implemented in the Onboard Controller sold with the commercial hexacopter. This low-level observer estimates roll and pitch angles, ϕ_{I} and θ_{I}, based only on the Onboard Controller's IMU data. These attitude estimates are then passed along with gyroscope and accelerometer data as inputs into another observer implemented on the user's separate processor. The second observer propagates velocity states according to

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \mathbf{a}_{\mathrm{m}}^{\mathrm{b}} + \mathbf{R}_{\mathrm{L}}^{\mathrm{b}}(\phi_{\mathrm{I}}, \theta_{\mathrm{I}})\mathbf{g}, \tag{2.23}$$

where $\mathbf{a}_{\mathrm{m}}^{\mathrm{b}}$ is the three axis accelerometer measurements in the body frame, $\mathbf{R}_{\mathrm{L}}^{\mathrm{b}}(\phi_{\mathrm{I}}, \theta_{\mathrm{I}})$ is the rotation matrix based on ϕ_{I} and θ_{I} from the local to the body frame, and $\mathbf{g}$ is the gravity vector expressed in the local reference frame. Exteroceptive measurements are used in an update step that corrects position and heading estimates.

Equation (2.23) is a valid method for propagating velocity estimates *only if* $\mathbf{R}_{\mathrm{L}}^{\mathrm{b}}$ is correct. We discuss at length in prior work [71] why ϕ_{I} and θ_{I} from commercial multirotor helicopters tend to be consistently inaccurate. To be brief here, the low-level observers on these platforms assume accelerometers can measure the direction of the gravity vector in situations where that is not the case. While we do not use a commercial multirotor autopilot to generate ϕ_{I} and θ_{I} for these results, we have taken care to make estimates of ϕ_{I} and θ_{I} qualitatively match those from a popular commercial multirotor platform[5]. This is done by using the observation discussed in [71] that typical estimates of ϕ_{I} and θ_{I} are actually a low-pass filtered version of truth[6].

To keep the presentation concise, we do not present here the estimation results for d, w, or ψ since the improved dynamic model does not offer any direct benefit in these states over the traditional approach. We will also omit discussing the estimation results for gyroscope biases. These biases can be easily recalibrated just before flight and do not evolve significantly over short flights like the one presented below. However, including gyroscope biases in the state is especially significant in scenarios where the vehicle flies autonomously for longer periods, such as when it can recharge autonomously [79], for example.

[5]http://www.asctec.de/home-en/

[6]We used to have a working Ascending Technologies platform, but it has since been rendered inoperable. We compared the output of a low-pass filter approximation of ϕ_{I} and θ_{I} to data previously recorded from the Ascending Technologies vehicle until we felt the filters were tuned appropriately.

The yaw angle ψ is kept close to zero for the majority of this nominal flight. When $\psi = 0$, then f_{L}, θ, and u are decoupled from r_{L}, ϕ, and v (i.e. changes in v caused by ϕ are solely responsible for changes in r_{L}). We will omit plots of f_{L}, θ, and u since they are qualitatively similar to the plots of r_{L}, ϕ, and v.

Figures 2.3 – 2.5 show results for the first minute of a manually controlled, nominal flight. We note in Figure 2.3 that errors in ϕ_{I} used in (2.23) do not appear exceptionally large. The most noticeable discrepancies are on the order of 10 degrees. The traditional approach also leads to errors for translational displacement in Figure 2.4 that seem somewhat tolerable. Although the displacement errors shown here can be as much as 0.5 meters, they are brief and at their worst only after a rapid transition.

However, the significance of errors in ϕ_{I} becomes more apparent when estimating the corresponding component of velocity. This is illustrated in Figure 2.5. The relatively infrequent rate (5 Hz) of position updates in these results cannot adequately compensate for the errors introduced in (2.23) by the incorrect rotation matrix. While the traditional estimate of v trends correctly, it can be off by as much as 1.5 m/s. This is despite relatively accurate (5 cm standard deviation) exteroceptive updates. The hexacopter depends on accurate state estimates, especially of velocity, to control its fast dynamics. Feeding back poor velocity estimates into a controller would have a deleterious effect on flight performance.

The more accurate estimates afforded by the improved model are also robust to decreased accuracy in the exteroceptive measurements. This is illustrated in Figure 2.6 where we plot the RMS error over the entire four minute flight for the roll, velocity, and position displacement. We calculate the RMS errors for three different levels of error in the exteroceptive sensor algorithm's measurement of position.

We note in Figure 2.6 that the estimates of ϕ_{I} are unaffected by changes in exteroceptive errors. This is because ϕ_{I} is estimated in a separate observer. However, it is worth noting that estimates of ϕ generated by using the improved model are also essentially unchanged by the quality of exteroceptive updates. The first four rows of $\mathbf{O}_{\mathrm{M}}$ in (2.20) suggest that the roll and pitch angles (as well as the velocities u and v) are observable based only on accelerometer measurements. We show in [71] how the improved dynamic model can also be used in the independent, low-level observer based only on IMU measurements to make its attitude estimates ϕ_{I} and θ_{I} more accurate. Doing this would consequently improve the integration of accelerometer measurements in (2.23), making the traditional approach more accurate.

As expected, Figure 2.6 also demonstrates a more graceful degradation in position and velocity estimates when using the improved model. Position and velocity states suffer more under the traditional approach relying on (2.23) because there is no direct correction of velocity. These results highlight the complementary nature of the improved dynamic model and corrections from exteroceptive sensor processing. The improved model allows velocity estimates to be corrected directly by frequent accelerometer data instead of relying only on indirect corrections from measurements of position. Better velocity estimates in turn improve position estimates through their kinematic relationship making all the estimates more robust to degraded information from the exteroceptive sensor.

Finally, Figure 2.7 and Figure 2.8 show estimation results for $\alpha_{\mathrm{j_b}}$, the bias in the body y-axis accelerometer; results for $\alpha_{\mathrm{i_b}}$ are similar. We note that for the estimation results shown above for the traditional approach we used carefully calibrated values for the accelerometer bias. However, this was not the case for the filter presented in this chapter. For those results the accelerometer bias estimates were initialized to zero as though they were not calibrated at all. Despite this handicap, the filter using the improved model was still able to achieve good performance.

Figure 2.7 shows estimates of $\alpha_{\mathrm{j_b}}$ based on three different initial conditions that represent a careful calibration, a poor calibration, and no calibration at all. Figure 2.7 shows that the estimate of accelerometer bias quickly converges to the correct value even without a calibrated initial value. Figure 2.8 further illustrates the behavior in the case of no prior calibration. The filter begins with a large initial error and marginal uncertainty for $\alpha_{\mathrm{j_b}}$. As the flight continues, the uncertainty and error decrease until reaching steady state behavior at around $t = 90$ seconds (i.e. about 60 seconds into the flight). Thereafter, the accelerometer updates in the EKF keep the random walk propagation of $\alpha_{\mathrm{j_b}}$ tightly bounded.

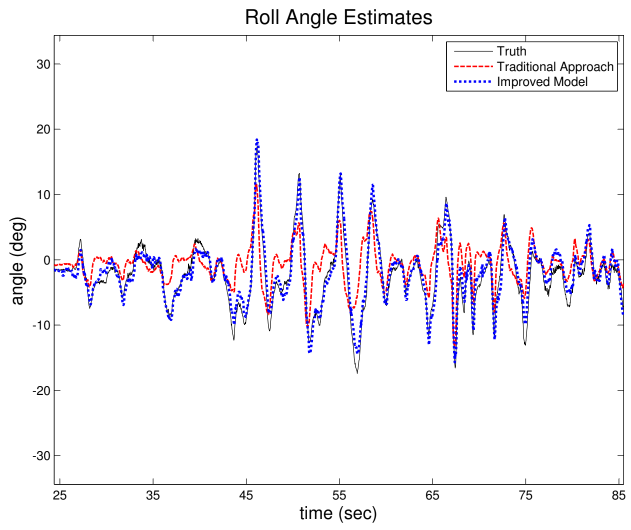

Figure 2.4: Estimates of roll angle, ϕ, during the first minute of a manually controlled flight. This results were generated using exteroceptive position updates arriving at 5 Hz, delayed on average by 250 ms, and with 5 cm standard deviation of error. Flight begins at about $t = 25$ seconds.

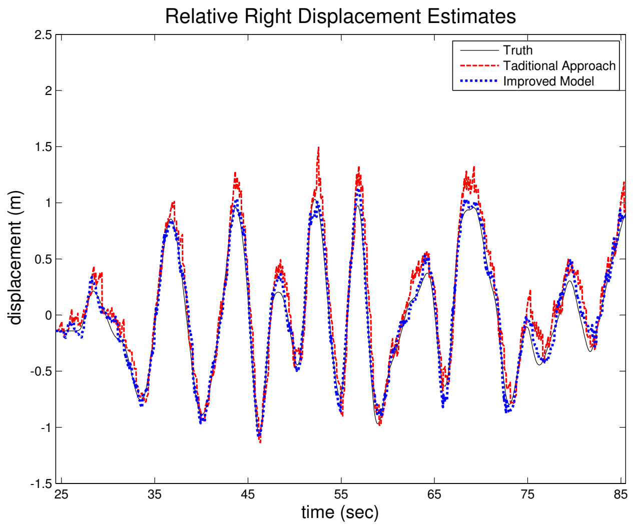

Figure 2.5: Estimates of right displacement from the local reference frame, r_{L}. See also the caption on Figure 2.3

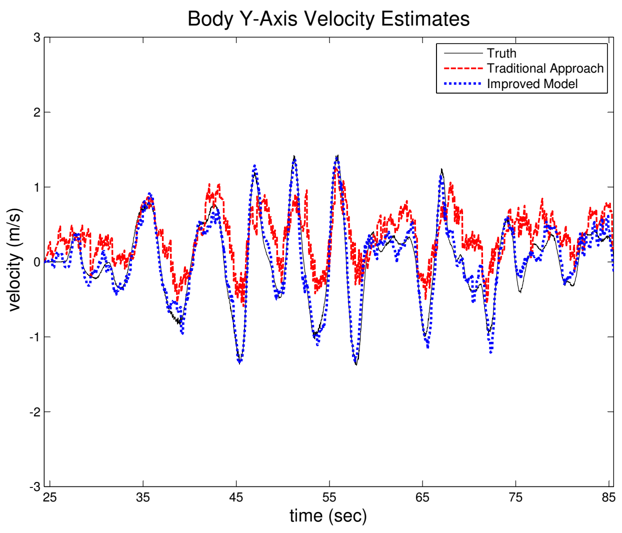

Figure 2.6: Estimates of the body y-axis velocity, v. See also the caption on Figure 2.3

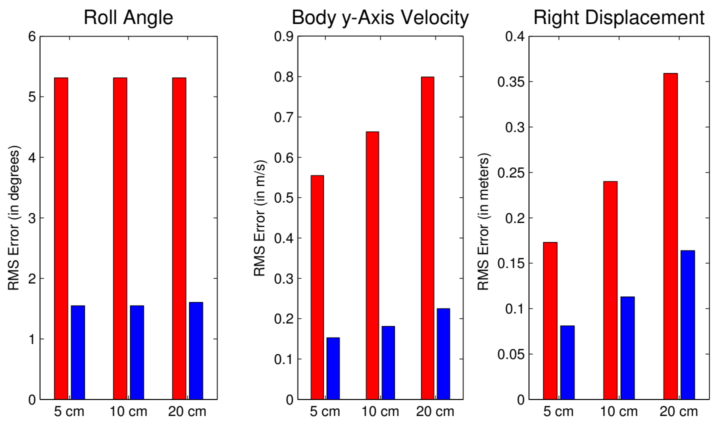

Figure 2.7: Average error in roll angles ϕ and ϕ_{I} (left), body y-axis velocity v (center), and relative right displacement r_{L} (right) over a four minute flight. Error of the traditional approach using (2.23) is graphed in red (light gray); error of estimates based on the improved model are graphed in blue (dark gray). Error is calculated for three scenarios differing in the level noise in the exteroceptive position update.

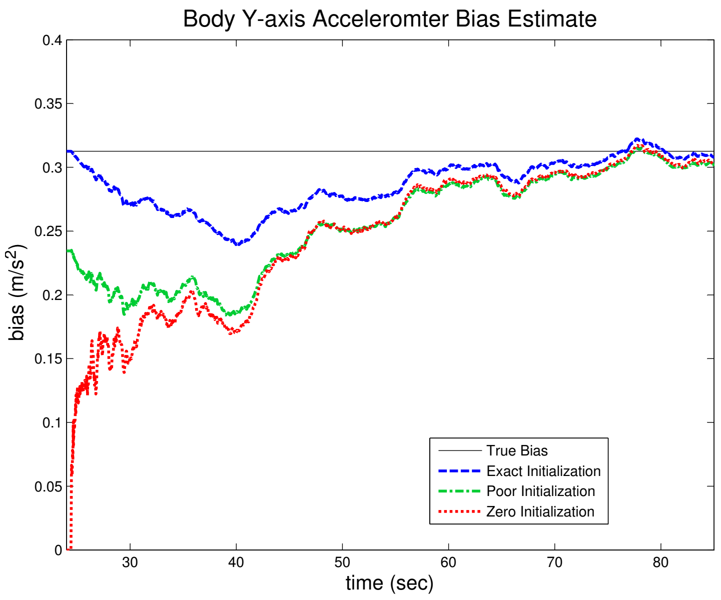

Figure 2.8: Estimates of α_{j_b} over the first minute of flight. The true bias is marked by the solid black line at about 0.31 m/s^2. The three estimates plotted correspond to accurate (blue dashed), poor (green dash-dot), and zero (red dotted) initial calibration. After the estimates converge they remain close to the true bias for the duration of the flight.

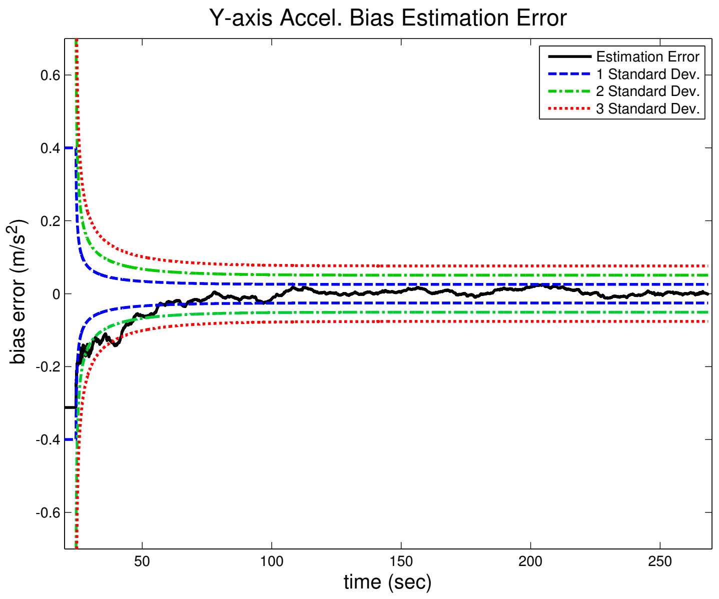

Figure 2.9: Error of the α_{j_b} estimate over the entire flight when initialized to zero (i.e. uncalibrated). The error is plotted as the solid black line. The remaining curves designate multiples of the marginal standard deviation of the error as calculated by the filter.

2.5 Chapter 2 Conclusion

We have presented and analyzed an observer based on an improved dynamic model for multirotor helicopters. The filter produces more accurate attitude and velocity estimates compared to traditional models based on (2.23) because it correctly accounts for the relationship between accelerometer measurements and velocity in the plane of the rotors. We have explained the improved estimation by analyzing the observability properties of the improved estimator. The increased accuracy provided by this observer complements and relaxes constraints on developing an appropriate vision-based position update. Using the model presented here, both accelerometer and exteroceptive measurements can be used to more effectively update the velocity estimates so critical to autonomous control of these vehicles.

The results presented here could be further improved by more frequent access to gyroscope and accelerometer measurements. Using IMU data at 40 Hz is modest compared to rates commonly reported in the literature (e.g. [80]). We used this rate only because of the current hardware limitations of the prototype testbed. However, improvements due to faster access to the IMU may not be worth the increased computation given the already accurate estimates.

We also note that attitude and the accelerometer bias estimates are almost completely unaffected by the accuracy of exteroceptive position measurements. This is consistent with the observability analysis of Section 2.3 which suggests that these state estimates are strongly a function of the accelerometer measurements.

Future work[7] developing this front-end filter centers on integrating it with a real-time vision-based position estimation algorithm similar to [78].

[7]Currently underway, and conducted by Robert Leishman.

Chapter 3

Back-End Optimization of Relative Transformations

In the preceding chapter we treated important elements of implementing and analyzing the Front-End Estimator (Figure 1.6). Because of the relative navigation scheme used in the front-end we have so far been able to discuss localization without much concern for mapping. However, long-term, repeatable navigation through unknown environments requires that an autonomous vehicle develop a map while it navigates. For the remainder of this dissertation we will focus on the process of optimizing that map in the Back-End Estimator.

The objective of the back-end optimization should be related to the real-time navigation needs of the vehicle. For example, if the vehicle makes time critical decisions based on global pose estimates, the back-end should be able to quickly compute their mean and covariance. If, on the other hand, the vehicle navigates relative to saved images of the environment, the relative transformations between those images may be the most important output from the back-end. Traditional back-end optimization in graph-based SLAM is focused on improving global pose estimates. The object of the optimization is to find the most likely configuration of global poses given fixed relative transformation constraints produced in the front-end.

In this chapter we show how making relative transformations the focus of the back-end optimization leads to a novel algorithm that iteratively improves the joint estimate of global poses and relative transformations. We test the algorithm on front-end data from a 225 meter closed-loop trajectory. Results show that the new algorithm provides the same global pose estimates in slightly less computational time when compared to g^{2}o, a state-of-the-art back-end optimization tool. However, g^{2}o requires several iterations in this test before the optimized relative transformations it produces become more accurate than their initial values. The algorithm presented in this chapter improves both global pose and relative transformation estimates at each iteration, making the result of the optimization suitable to a wider variety of applications.

3.1 Algorithm Development

In this section we present the initial development leading to our alternative approach to back-end optimization. We first discuss some relevant background material related to Bayesian networks and manipulating Gaussian distributions. With that context we then present the details of our approach. We conclude this section by discussing similarities and differences between our work and that of [27, 68–70].

3.1.1 Bayesian Network Concepts

An excellent introduction to Bayesian networks can be found in [66]. We have also found [81] to be a useful source for additional insights. We only mention a few concepts in this section and refer the reader to these and other sources for a more thorough treatment.

A Bayesian network (also called a Bayes net or belief network) is a directed acyclic graph that represents conditional dependencies among a collection of random variables. The random variables make up the nodes[1] in the graph depicted by a labeled circle. A probabilistic dependence is indicated by a directed edge such that the node at the arrow's head is dependent on the node at the arrow's tail. To use the jargon, the variable at the head is referred to as the child node. The variable at the tail is called the parent node. The Asia Network in Figure 3.0 offers a canonical example of a Bayesian network and illustrates the concepts we introduce here.

The authors of [81] recommend that Bayesian networks be constructed so that the direction of an edge represent a causal relationship. This is not technically necessary; a simple Bayes net with two nodes and a single edge would represent the joint distribution over two variables, e.g. $\mathbf{y}$ and $\mathbf{z}$. From basic rules of probability we know that the joint

[1]Because they are both graphical representations, graph-based SLAM and Bayesian networks use similar terminology. We caution the reader to carefully examine the context to identify what is being described. We'll also do our best not to conflate the concepts.

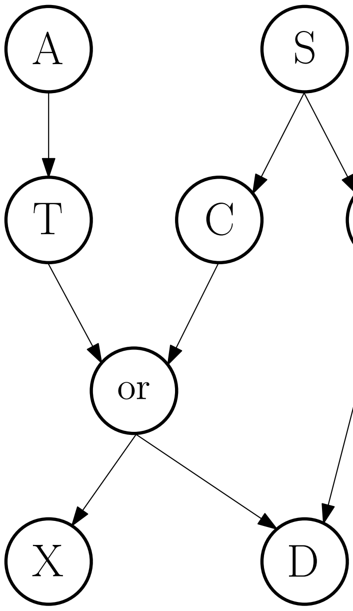

Figure 3.1: The so-called Asia Network, a canonical example of a Bayesian network. The network models a physician's belief about the variables: A - the patient has recently been to Asia; S - the patient is a smoker; T - the patient has tuberculosis; C - the patient has cancer; B - the patient has bronchitis; X - the results of a patient's X-ray are abnormal; D - the patient exhibits dyspnoea (i.e. shortness of breath). The variable labeled 'or' is a mediating variable that captures the belief that the patient has either tuberculosis or cancer.

distribution can be written such that

$$\mathrm{p}(\mathbf{y}, \mathbf{z}) = \mathrm{p}(\mathbf{y}|\mathbf{z})\mathrm{p}(\mathbf{z}) = \mathrm{p}(\mathbf{z}|\mathbf{y})\mathrm{p}(\mathbf{y}).$$

As a consequence, we can draw that edge in either direction between the two nodes to represent the probabilistic dependence between **y** and **z**. However, the authors of [81] argue that using edges to represent causal relationships makes inference more intuitive. Causal relationships are manifest in Figure 3.0. Smoking, for example, has a causal influence on getting cancer or bronchitis.

It is also useful to mention here the concept of a Markov blanket. Formally defined, the Markov blanket for a given variable, x, is the set of its parents, children, and co-parents. More intuitively, we can think of a Markov blanket as the minimal set of nodes that completely isolates our belief about x from the rest of the network [66]. The distribution of x is conditionally independent of all other variables *given* the variables in its Markov blanket. In

Figure 3.0, the Markov blanket for node 'C' consists of nodes 'S' (parent), 'or' (child), and 'T' (co-parent).

Finally, we introduce here the concept of a mediating variable. The authors of [81] describe mediating variables as, "variables for which posterior probabilities are not of immediate interest, but which play important roles for achieving correct conditional independence and dependence properties and/or efficient inference." In Figure 3.0 the 'or' variable is a mediating variable.

3.1.2 Updating Joint Gaussian Distributions

In this subsection we discuss some details of the interplay between joint, conditional, and marginal distributions. The joint distribution is the ultimate description of a collection of random variables. Conditional and marginal distributions allow one to decompose the joint distribution, and marginal distributions are especially relevant when we receive outside information (i.e. a measurement) about a subset of the random variables in a joint distribution. We will only briefly present some derivations here that relate these types of distributions for jointly Gaussian random variables. For more detail, see Appendix B.

Let a D-dimensional random vector $\mathbf{x}$ of jointly Gaussian variables be partitioned into two, disjoint sub-vectors such that $\mathbf{x} = \left[\mathbf{x}_1^\top, \mathbf{x}_2^\top\right]^\top$, where $\mathbf{x}_1$ is dimension D_1 and $\mathbf{x}_2$ is dimension D_2. Then the joint distribution $\mathrm{p}(\mathbf{x})$, with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, is partitioned such that

$$\begin{aligned} \boldsymbol{\mu} &= \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \\ \boldsymbol{\Sigma} &= \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}. \end{aligned}$$

Typical textbook derivations (e.g. [66, Chapter 2.3.1]) define the conditional distribution $\mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)$, with mean $\boldsymbol{\mu}_{1|2}$ and covariance $\boldsymbol{\Sigma}_{1|2}$, such that

$$\boldsymbol{\mu}_{1|2} \triangleq \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \tag{3.1}$$

$$\boldsymbol{\Sigma}_{1|2} \triangleq \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}. \tag{3.2}$$

To simplify notation in the sequel we define

$$\mathbf{K} \triangleq \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}. \tag{3.3}$$

There are a few undesirable aspects of expressing the conditional distribution $\mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)$ using (3.1) and (3.2). First, the mean vector $\boldsymbol{\mu}_{1|2}$ has dimension D_1. The marginal distribution $\mathrm{p}(\mathbf{x}_2)$ has a mean $\boldsymbol{\mu}_2$ of dimension D_2. To recover the joint distribution $\mathrm{p}(\mathbf{x}) = \mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)\,\mathrm{p}(\mathbf{x}_2)$ would require that we sum exponents with different dimensions. It is also unattractive to leave the conditional distribution's functional dependence on $\mathbf{x}_2$ buried in the conditional mean.

We can rewrite the conditional distribution $\mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)$ (see Appendix B) such that

$$\log(\mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)) \propto (\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{A} (\mathbf{x} - \boldsymbol{\mu}), \tag{3.4}$$

where

$$\mathbf{A} \triangleq \begin{bmatrix} \boldsymbol{\Sigma}_{1|2}^{-1} & -\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{K} \\ -\mathbf{K}^\top\boldsymbol{\Sigma}_{1|2}^{-1} & \mathbf{K}^\top\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{K} \end{bmatrix}. \tag{3.5}$$

Note that the dimension and mean of the conditional exponent now correspond to the original joint distribution, and the conditional distribution's functional dependence on $\mathbf{x}_2$ is clear. We can similarly rewrite $\mathrm{p}(\mathbf{x}_2)$ such that

$$\log(\mathrm{p}(\mathbf{x}_2)) \propto (\mathbf{x} - \mathbf{b})^\top \mathbf{B} (\mathbf{x} - \mathbf{b}), \tag{3.6}$$

where the D-dimensional vector $\mathbf{b}$ is defined as

$$\mathbf{b} \triangleq \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mu}_2 \end{bmatrix},$$

and

$$\mathbf{B} \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{22}^{-1} \end{bmatrix}.$$

In Appendix B we offer some additional observations about the relationship between the information matrices $\mathbf{A}$, $\mathbf{B}$, and $\boldsymbol{\Sigma}^{-1}$.

As mentioned above, the marginal distribution is important when incorporating new information from a measurement that involves a subset of variables from the full joint distribution. Let the measurement be a function of $\mathbf{x}_2$, and let $\mathrm{p}(\check{\mathbf{x}}_2)$, with mean $\check{\boldsymbol{\mu}}_2$ and information matrix $\check{\boldsymbol{\Sigma}}_{22}^{-1}$, represent our belief about the states updated by the measurement. We also define

$$\check{\mathbf{b}} \triangleq \begin{bmatrix} \mathbf{0} \\ \check{\boldsymbol{\mu}}_2 \end{bmatrix},$$
$$\check{\mathbf{B}} \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \check{\boldsymbol{\Sigma}}_{22}^{-1} \end{bmatrix}.$$

To propagate the new information contained in $\check{\mathbf{x}}_2$ into the remaining elements of $\mathbf{x}$ we recover the joint distribution from the conditional and the updated marginal distributions: $\mathrm{p}(\mathbf{x}) = \mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)\mathrm{p}(\check{\mathbf{x}}_2)$. This product gives

$$\begin{aligned} \log\left(\mathrm{p}\left(\mathbf{x}\right)\right) &\propto (\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{A}(\mathbf{x}-\boldsymbol{\mu}) + (\mathbf{x}-\check{\mathbf{b}})^\top \check{\mathbf{B}}(\mathbf{x}-\check{\mathbf{b}}) \\ &\propto (\mathbf{x}-\overset{\star}{\boldsymbol{\mu}})^\top \overset{\star}{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\overset{\star}{\boldsymbol{\mu}}), \end{aligned}$$

where the optimized joint covariance and mean are

$$\begin{aligned}
\overset{\star}{\boldsymbol{\Sigma}} &\triangleq \left(\mathbf{A} + \check{\mathbf{B}}\right)^{-1} \\
&= \begin{bmatrix} \boldsymbol{\Sigma}_{1|2}^{-1} & -\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{K} \\ -\mathbf{K}^\top\boldsymbol{\Sigma}_{1|2}^{-1} & \check{\boldsymbol{\Sigma}}_{22}^{-1} + \mathbf{K}^\top\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{K} \end{bmatrix}^{-1}, \qquad (3.7) \\
\overset{\star}{\boldsymbol{\mu}} &\triangleq \overset{\star}{\boldsymbol{\Sigma}} \left(\mathbf{A}\boldsymbol{\mu} + \check{\mathbf{B}}\check{\mathbf{b}}\right). \qquad (3.8)
\end{aligned}$$

The optimized covariance $\overset{\star}{\boldsymbol{\Sigma}}$ of the joint distribution can be recovered without inverting the information matrix as suggested by (3.7). Instead, following the derivation in Appendix B, we find the optimized covariance and mean to be

$$\begin{aligned}
\overset{\star}{\boldsymbol{\Sigma}} &= \begin{bmatrix} \overset{\star}{\boldsymbol{\Sigma}}_{11} & \overset{\star}{\boldsymbol{\Sigma}}_{12} \\ \overset{\star}{\boldsymbol{\Sigma}}_{21} & \overset{\star}{\boldsymbol{\Sigma}}_{22} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{\Sigma}_{11} - \mathbf{K}\left(\boldsymbol{\Sigma}_{22} - \check{\boldsymbol{\Sigma}}_{22}\right)\mathbf{K}^\top & \mathbf{K}\check{\boldsymbol{\Sigma}}_{22} \\ \check{\boldsymbol{\Sigma}}_{22}\mathbf{K}^\top & \check{\boldsymbol{\Sigma}}_{22} \end{bmatrix}, \qquad (3.9) \\
\overset{\star}{\boldsymbol{\mu}} &= \begin{bmatrix} \boldsymbol{\mu}_1 - \mathbf{K}\left(\boldsymbol{\mu}_2 - \check{\boldsymbol{\mu}}_2\right) \\ \check{\boldsymbol{\mu}}_2 \end{bmatrix}. \qquad (3.10)
\end{aligned}$$

3.1.3 Optimizing Relative Transformations

Most back-end optimization routines seek to optimize the global poses by minimizing the squared Mahalanobis distance metric we defined previously in Chapter 1.2.3:

$$\epsilon(\boldsymbol{\rho}) = \sum_{(i,j)\in\mathcal{C}} \left(\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right) - \boldsymbol{\tau}_i^j\right)^\top \boldsymbol{\Sigma}_{i,j}^{-1} \left(\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right) - \boldsymbol{\tau}_i^j\right). \qquad (3.11)$$

The function $\epsilon(\boldsymbol{\rho})$ is parameterized by saved relative transformations considered to be "measured" in the front end and fixed in the back-end. Equation (3.11) compares the saved transformations to the relative transformations derived from the optimized global poses. The global poses are the focus of the optimization.

However, those saved relative transformations are themselves only estimates. As described in [71] and Chapter 1.2.2, the saved estimates of the relative transformations are an amalgamation of several IMU, altimeter, and visual odometry measurements. There is no general guarantee that one will arrive at a globally accurate map by making global pose estimates agree with error prone relative transformation estimates. This may be the reason why the authors of [58] argue that Equation (3.11) is "not an adequate measure of graph quality."

We also note that the set of all relative transformations fully characterizes the globally metric map just as much as the set of all the global poses. By this we mean that if all the relative transformations are known without error then the map is perfectly defined. Since the relative transformations are the only thing we can actually observe, why not seek to further refine the transformation estimates in the back-end. Using a relative navigation scheme in the front-end makes this subtle shift in emphasis seem especially relevant. A focus on optimizing relative transformations is the underlying philosophy driving our approach to back-end optimization.

Direct Approach

For the remainder of this chapter we will consider the simple case of a vehicle traveling around a long, rectangular hallway before returning to the origin to detect a single loop closure. This scenario allows us to examine some of the fundamental aspects of our approach. In Chapter 4 we will investigate extending the algorithm to more complex map topologies.

For this simple scenario, a direct approach to optimizing relative transformations might be modeled by the Bayesian network in Figure 3.1. The relative transformations represented in the top row exist between N + 1 temporally consecutive poses corresponding to images saved during exploration. We will refer to these transformations between temporally consecutive poses as odometry-like transformations; they are of the form $\boldsymbol{\tau}_i^{i+1}$. The transformation at the bottom of Figure 3.1 corresponds to the loop closure transformation from the origin to the pose of the last image in the trajectory.

When a new image of the environment is first saved, the vehicle begins navigation with respect to that image. The front end can estimate its change in pose with respect to

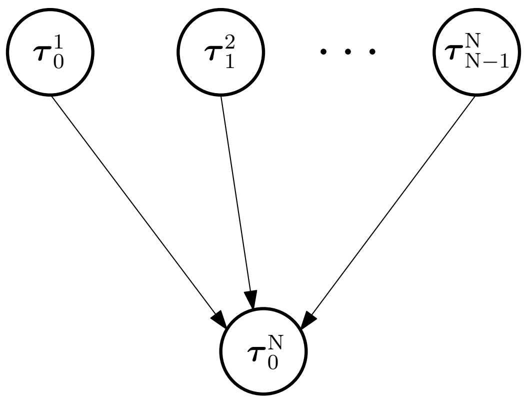
$\boldsymbol{\tau}_0^1$
$\boldsymbol{\tau}_1^2$
$\cdots$
$\boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}$
$\boldsymbol{\tau}_0^{\mathrm{N}}$

Figure 3.2: A Bayesian network for a direct approach to optimizing the relative transformations. The nodes at the top represent odometry-like transformations between temporally consecutive saved images. The node at the bottom represents the single loop closure considered in this discussion. When evidence about $\boldsymbol{\tau}_0^{\mathrm{N}}$ becomes available through a loop closure measurement, all of the odometry-like measurements become correlated by the "explaining away" phenomenon [66].

that image *independent* of any other estimates of global poses or relative transformations. This is because the vehicle was certainly at the spot where the image was captured, no matter where that image is in the world or how the vehicle got there. This independence is reflected by the fact that no arrows enter the variables in the top row of Figure 3.1.

As the vehicle continues to explore, our prior belief about the odometry-like transformations $\boldsymbol{\tau}_i^{i+1}$ provides the only source of information about the relative transformation $\boldsymbol{\tau}_0^{\mathrm{N}}$. This is reflected in the Bayesian network of Figure 3.1 by the arrows pointing from all of the $\boldsymbol{\tau}_i^{i+1}$ into $\boldsymbol{\tau}_0^{\mathrm{N}}$. However, at loop closure the vehicle measures its pose relative to the origin. This "evidence," to use the Bayesian network parlance, changes our belief about $\boldsymbol{\tau}_0^{\mathrm{N}}$. For a Bayesian network like that of Figure 3.1, evidence on $\boldsymbol{\tau}_0^{\mathrm{N}}$ makes all of the $\boldsymbol{\tau}_i^{i+1}$ transformations correlated [66].

The reason for the correlation can also be understood by considering the covariance matrix for the joint distribution $\mathrm{p}(\boldsymbol{\tau})$, where

$$\boldsymbol{\tau} \triangleq \begin{bmatrix} \boldsymbol{\tau}_0^1 \\ \boldsymbol{\tau}_1^2 \\ \vdots \\ \boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}} \\ \boldsymbol{\tau}_0^{\mathrm{N}} \end{bmatrix}.$$

Before measuring a loop closure the covariance matrix has an upper left submatrix that is block diagonal because of the independence of the relative transformations $\boldsymbol{\tau}_i^{i+1}$. However, the last block row and column are dense because our prior belief about $\boldsymbol{\tau}_0^{\mathrm{N}}$ is a function of all the transformations preceding it. When a measurement of $\boldsymbol{\tau}_0^{\mathrm{N}}$ is applied to the joint distribution $\mathrm{p}(\boldsymbol{\tau})$ via a Kalman update step, the cross-covariance elements in the last block row and column cause the remainder of the matrix to become dense also.

The correlations induced by the first loop closure measurement make it intractable to directly estimate the joint distribution over all relative transformations. To apply any future loop closure would require manipulating a large, dense covariance matrix. In a sense, this is analogous to naive EKF SLAM; using a single covariance matrix to keep track of correlations between all state elements is too computationally expensive.

Decomposing the Joint Distribution

Consider the relationship between a particular relative transformation $\boldsymbol{\tau}_i^j$ and the rest of the relative transformations if the global poses $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$ are *given.* Knowing these global poses ensures our belief about $\boldsymbol{\tau}_i^j$ is always independent from the remaining relative transformations because $\boldsymbol{\tau}_i^j$ is only defined by $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$. The global poses $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$ constitute a Markov blanket for $\boldsymbol{\tau}_i^j$.

This motivates us to reconsider our approach to back-end optimization by using the joint distribution $\mathrm{p}(\boldsymbol{\tau}, \boldsymbol{\rho})$. We construct the Bayesian network in Figure 3.2 following the guideline of assigning directed edges based on causal relationships. Each relative transforma-

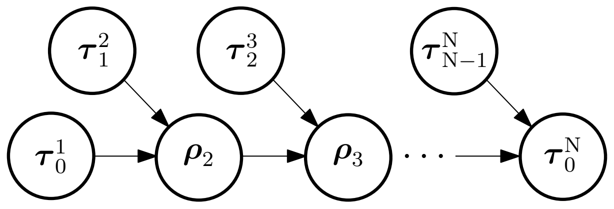

Figure 3.3: A Bayesian network based on the idea of using global poses as mediating variables, thus leading to the joint distribution $\mathrm{p}(\boldsymbol{\tau}, \boldsymbol{\rho})$. Directed edges in the network are drawn to represent the physical reality that a relative transformation moves the vehicle from one global pose to another.

tion points into a global pose because a transformation estimate represents our belief about how the vehicle arrived there from the previous pose. We have made $\boldsymbol{\rho}_0$ the arbitrary and certain global origin, therefore $\boldsymbol{\rho}_1 = \boldsymbol{\tau}_0^1$.

The remaining global poses can be considered mediating variables that help explain the "correct conditional independence ... properties" [81] between the relative transformations. For our simple scenario in Figure 3.2, if $\boldsymbol{\rho}_2$ is given then our belief about $\boldsymbol{\tau}_0^1$ and $\boldsymbol{\tau}_1^2$ cannot be affected by any other relative transformation in the network. It is also evident from this Bayesian network that the Markov blanket of $\boldsymbol{\tau}_2^3$, for example, consists of $\boldsymbol{\rho}_2$ (co-parent) and $\boldsymbol{\rho}_3$ (child).

The Bayesian network in Figure 3.2 suggests the following decomposition of the joint distribution $\mathrm{p}(\boldsymbol{\tau}, \boldsymbol{\rho})$. To keep expressions compact we introduce the notation $\mathbf{x}_\mathrm{r}$ to indicate the remaining variables in a random vector $\mathbf{x}$ that have not already been broken out in the preceding conditional distributions. Then the joint distribution can be written as

$$\begin{aligned}
\mathrm{p}(\boldsymbol{\tau}, \boldsymbol{\rho}) &= \mathrm{p}\left(\boldsymbol{\tau}_0^1, \boldsymbol{\tau}_1^2 | \boldsymbol{\tau}_\mathrm{r}, \boldsymbol{\rho}_\mathrm{r}\right) \mathrm{p}(\boldsymbol{\tau}_\mathrm{r}, \boldsymbol{\rho}_\mathrm{r}) \\
&= \mathrm{p}\left(\boldsymbol{\tau}_0^1, \boldsymbol{\tau}_1^2 | \boldsymbol{\rho}_2\right) \mathrm{p}(\boldsymbol{\tau}_\mathrm{r}, \boldsymbol{\rho}_\mathrm{r}) \\
&= \mathrm{p}\left(\boldsymbol{\tau}_0^1, \boldsymbol{\tau}_1^2 | \boldsymbol{\rho}_2\right) \mathrm{p}\left(\boldsymbol{\tau}_2^3, \boldsymbol{\rho}_2 | \boldsymbol{\tau}_\mathrm{r}, \boldsymbol{\rho}_\mathrm{r}\right) \mathrm{p}(\boldsymbol{\tau}_\mathrm{r}, \boldsymbol{\rho}_\mathrm{r}) \\
&= \mathrm{p}\left(\boldsymbol{\tau}_0^1, \boldsymbol{\tau}_1^2 | \boldsymbol{\rho}_2\right) \mathrm{p}\left(\boldsymbol{\tau}_2^3, \boldsymbol{\rho}_2 | \boldsymbol{\rho}_3\right) \mathrm{p}(\boldsymbol{\tau}_\mathrm{r}, \boldsymbol{\rho}_\mathrm{r}) \\
&\vdots \\
&= \mathrm{p}\left(\boldsymbol{\tau}_0^1, \boldsymbol{\tau}_1^2 | \boldsymbol{\rho}_2\right) \mathrm{p}\left(\boldsymbol{\tau}_2^3, \boldsymbol{\rho}_2 | \boldsymbol{\rho}_3\right) \cdots \\
&\qquad \cdots \mathrm{p}\left(\boldsymbol{\tau}_{N-1}^N, \boldsymbol{\rho}_{N-1} | \boldsymbol{\tau}_0^N\right) \mathrm{p}\left(\boldsymbol{\tau}_0^N\right). \qquad (3.12)
\end{aligned}$$

After completely decomposing $\mathrm{p}\left(\boldsymbol{\tau}, \boldsymbol{\rho}\right)$ we have the marginal distribution $\mathrm{p}\left(\boldsymbol{\tau}_0^{\mathrm{N}}\right)$ at the end of (3.12). The distribution $\mathrm{p}\left(\boldsymbol{\tau}_0^{\mathrm{N}}\right)$ is our marginal belief about the loop closure transformation. The loop closure measurement will be applied to this marginal distribution so that $\mathrm{p}\left(\check{\boldsymbol{\tau}}_0^{\mathrm{N}}\right)$ represents our updated belief about this relative transformation.

We can now use Equations (3.9) and (3.10) to find the optimized joint distribution $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{\mathrm{N}-1}, \overset{\star}{\boldsymbol{\tau}}_{\mathrm{N}-1}^{\mathrm{N}}, \overset{\star}{\boldsymbol{\tau}}_0^{\mathrm{N}}\right)$. In those calculations, the states $\boldsymbol{\rho}_{\mathrm{N}-1}$ and $\boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}$ play the part of $\mathbf{x}_1$, and $\boldsymbol{\tau}_0^{\mathrm{N}}$ plays the part of $\mathbf{x}_2$. We note that this is a simple operation because the matrices involved are small. Once we have $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{\mathrm{N}-1}, \overset{\star}{\boldsymbol{\tau}}_{\mathrm{N}-1}^{\mathrm{N}}, \overset{\star}{\boldsymbol{\tau}}_0^{\mathrm{N}}\right)$ we can trivially extract the marginal distribution $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{\mathrm{N}-1}\right)$ and again apply Equations (3.9) and (3.10) to find $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{\mathrm{N}-2}, \overset{\star}{\boldsymbol{\tau}}_{\mathrm{N}-2}^{\mathrm{N}-1}, \overset{\star}{\boldsymbol{\rho}}_{\mathrm{N}-1}\right)$. This pattern repeats back through the network until all variables have been updated with the loop closure information.

Algorithm 1 summarizes the back-end optimization process. Relative transformation estimates are originally produced in the front end. Those relative transformations can be composed to find our prior belief about the joint distributions over a relative transformation and the global poses it connects (Algorithm 1, lines 1 and 2). After a loop closure measurement is applied (line 3), the new information can be propagated back through the network of small joint distributions by repeated use of Equations (3.9) and (3.10) (lines 4 - 6). Since our goal is <u>B</u>ack-<u>E</u>nd optimization of <u>R</u>elative <u>T</u>ransformations, we will dub this approach BERT.

Small joint distributions of the form $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$ are the important entity in BERT. We can use a collection of such distributions as a modular representation of the entire joint distribution, $\mathrm{p}(\boldsymbol{\tau}, \boldsymbol{\rho})$. For example, in the simple scenario under consideration, our collection of small joint distributions is

$$\begin{gathered}
\mathrm{p}\left(\boldsymbol{\rho}_1, \boldsymbol{\tau}_1^2, \boldsymbol{\rho}_2\right) \\
\mathrm{p}\left(\boldsymbol{\rho}_2, \boldsymbol{\tau}_2^3, \boldsymbol{\rho}_3\right) \\
\mathrm{p}\left(\boldsymbol{\rho}_3, \boldsymbol{\tau}_3^4, \boldsymbol{\rho}_4\right) \\
\vdots \\
\mathrm{p}\left(\boldsymbol{\rho}_{\mathrm{N}-1}, \boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}, \boldsymbol{\rho}_{\mathrm{N}}\right).
\end{gathered}$$

Algorithm 1: BERT using a modular representation of $\mathrm{p}(\boldsymbol{\tau}, \boldsymbol{\rho})$ for the single loop scenario.

```
  for ( $i = 1$; $i < \mathrm{N}$; $i$++) do
1     Compose $\boldsymbol{\rho}_i$ with $\boldsymbol{\tau}_i^{i+1}$ to find $\boldsymbol{\rho}_{i+1}$.
2     Calculate the joint covariance for $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1}\right)$.
  end
3 Apply the loop closure measurement in a Kalman update of $\mathrm{p}(\check{\boldsymbol{\tau}}_0^{\mathrm{N}})$.
4 Use Equations (3.9) and (3.10) to find $\mathrm{p}(\overset{\star}{\boldsymbol{\rho}}_{\mathrm{N}-1}, \overset{\star}{\boldsymbol{\tau}}_{\mathrm{N}-1}^{\mathrm{N}}, \overset{\star}{\boldsymbol{\tau}}_0^{\mathrm{N}})$ where $\mathbf{x}_1$ corresponds to
  $\boldsymbol{\rho}_{\mathrm{N}-1}$ and $\boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}$, and $\mathbf{x}_2$ corresponds to $\boldsymbol{\tau}_0^{\mathrm{N}}$.
  for ( $i = \mathrm{N}-1$; $i > 0$; $i$--) do
5     Extract the marginal distribution $\mathrm{p}(\overset{\star}{\boldsymbol{\rho}}_i)$ from $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_i, \overset{\star}{\boldsymbol{\tau}}_i^{i+1}, \overset{\star}{\boldsymbol{\rho}}_{i+1}\right)$
6     Find $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{i-1}, \overset{\star}{\boldsymbol{\tau}}_{i-1}^{i}, \overset{\star}{\boldsymbol{\rho}}_i\right)$ using Equations (3.9) and (3.10) where $\mathbf{x}_1$ corresponds to
      $\boldsymbol{\rho}_{i-1}$ and $\boldsymbol{\tau}_{i-1}^{i}$, and $\mathbf{x}_2$ corresponds to $\boldsymbol{\rho}_i$
  end
```

We can update any of the individual variables within any $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$ and then propagate that information through the remaining small joint distributions based on Equations (3.9) and (3.10).

Notice that a given global pose variable can occur in multiple distributions. The prior belief about the global pose variables is identical in each small joint distribution by construction. This redundancy highlights the role of global poses as mediating variables. They serve to isolate our belief about each relative transformation, and they act as containers where information can be stored before propagating it through the rest of the network.

3.1.4 Comparison to CI Submap EKF-SLAM

We find some parallels and differences between BERT and the algorithms described in [27] and subsequent articles [68–70].

Similar to BERT, [27] divides the entire joint distribution into conditionally independent modules. In their case these modules are EKF SLAM sub-maps, primarily consisting of point features in the environment. They also show how duplicating some elements in multiple sub-maps allows the sub-maps to remain conditionally independent and efficiently share information gained at loop closure. During exploration the duplicated variables used in [27] consist of point feature estimates and vehicle pose estimates shared by temporally consecutive sub-maps.

We believe some important differences between [27] and our work arise from the way we represent the environment. In our concept of graph-based SLAM, we save key images of the environment that coincide with past vehicle poses. We do not reduce the raw images into a particular set of individual features. This removes feature position estimates from the back-end optimization, making it more efficient.

More importantly, because the vehicle was certainly at the pose represented by the saved image, reobserving the image is equivalent to reobserving the vehicle's actual pose at that previous point in time. In [69] the authors observe, "The price paid to maintain the conditional independence between submaps is some overhead in the size of the maps. We call overhead to *all those elements of a submap that cannot be observed from it, i.e., robot positions corresponding to the transitions between submaps and [additional] features included in the current submap* [only because they are needed to transmit new measurement information along the chain of intermediate submaps]." (emphasis added)

In their initial work [27], the authors use a simple, single loop closure scenario to illustrate their method just as we do here. Even in this simple case some feature estimates must be added as overhead to each of the sub-maps around the loop. Such is not the case with the approach we have presented.

We acknowledge that some non-trivial work remains on our part to establish whether we will need in the general case something equivalent to the overhead described in [27] and [69]. In [69] the authors are describing the extension of their algorithm to complex map topologies. Such an extension remains an item of ongoing work for us that we discuss in Chapter 4. However, observing past poses in the form of saved images is a fundamental departure from feature-based maps.

3.2 Results and Further Development

3.2.1 Test Scenario

We have demonstrated in prior work [71, 82] that the roll and pitch of a multi-rotor vehicle can be estimated with high accuracy using an improved dynamic model in the front-end observer. It is also possible to use vision measurements in conjunction with frequent IMU and laser altimeter measurements to maintain accurate estimates of the vehicle's relative down position. Accordingly, we will confine our presentation of results to 2D optimization. Future work includes applying BERT to all six degrees of freedom.

Let $\boldsymbol{\rho}_i \triangleq [n_i,\ e_i,\ \psi_i]^\top$, where the first two elements represent the global north and east position, and the last element is the clockwise rotation of the vehicle's forward direction measured from the global north axis. Let $\boldsymbol{\tau}_i^j \triangleq [\Delta f,\ \Delta r,\ \Delta\psi]^\top$, where the elements respectively represent the change in position along the local forward and right axes and the vehicle's change of heading. We define the composition function for 2D pose to be

$$\begin{aligned} \mathbf{g}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j\right) &\triangleq \begin{bmatrix} n_i + \Delta f \cos\psi_i - \Delta r \sin\psi_i \\ e_i + \Delta f \sin\psi_i + \Delta r \cos\psi_i \\ \psi_i + \Delta\psi \end{bmatrix} \\ &= \begin{bmatrix} n_j \\ e_j \\ \psi_j \end{bmatrix}. \end{aligned} \tag{3.13}$$

Similarly, we define the measurement equation used to evaluate Equation (3.11) to be

$$\begin{aligned} \mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right) &\triangleq \begin{bmatrix} \left(n_j - n_i\right)\cos\psi_i + \left(e_j - e_i\right)\sin\psi_i \\ -\left(n_j - n_i\right)\sin\psi_i + \left(e_j - e_i\right)\cos\psi_i \\ \psi_j - \psi_i \end{bmatrix} \\ &= \begin{bmatrix} \Delta f \\ \Delta r \\ \Delta\psi \end{bmatrix}. \end{aligned} \tag{3.14}$$

For this test we generate front-end data using a Simulink simulation based on the dynamic model and state observers described in [71]. The vehicle flies around a rectangular hallway with a total trajectory length of about 225 meters. During exploration the vehicle designates new saved images about every 0.3 meters change in position or 10 degrees change in heading. This results in 735 odometry-like relative transformations. In this experiment there is only a single loop closure transformation between the last pose and the origin.

The front-end saves the relative transformation estimates and their uncertainty for use in the back-end. The average error in the position change per transformation is 2.1 cm, or about 7% of the length of the transformation. The average error in the heading change of a transformation is 0.0115 radians (0.7 degrees). We note that the covariance matrix is not simply diagonal.

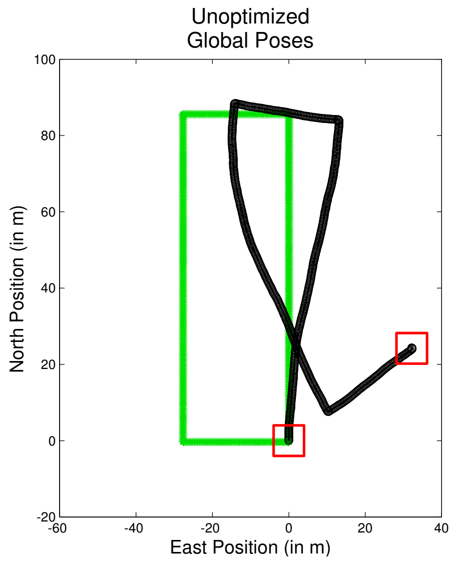

Figure 3.4: True global poses in green (gray) verses the unoptimized estimate (black) based on composed odometry-like transformations. Red boxed regions are shown close up in Figures 3.4 & 3.5.

While these errors may not seem exceptionally high, the length of the trajectory allows for significant error to accumulate in the global pose estimates. Figure 3.3 shows an overhead view of the true global poses compared to the composition of unoptimized odometry-like transformations. Figures 3.4 and 3.5 show close up portions of Figure 3.3 to emphasize the error in the global pose estimate that has accumulated by the end of the trajectory.

We implement BERT using the Eigen C++ linear algebra library. We compare our results with those obtained using g^{2}o [63], which also relies on the Eigen library. The g^{2}o

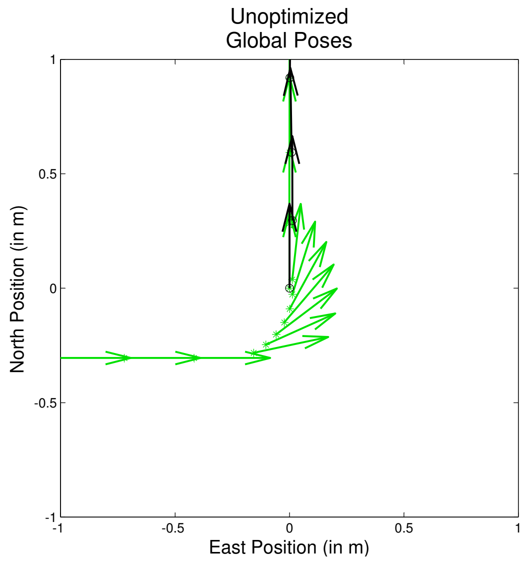

Figure 3.5: A close up view of Figure 3.3 around the true beginning and end of the trajectory, where loop closure occurs. Arrows indicate the true (green/gray) and estimated (black) heading; the corresponding positions are marked (green/gray stars; black circles) at the base of each arrow.

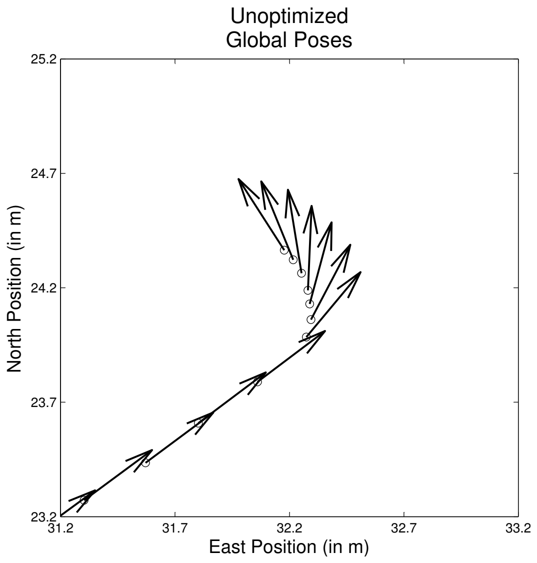

Figure 3.6: A close up view of Figure 3.3 around the odometry-based estimate of the end of the trajectory. See also the caption on Figure 3.4

code is open-source and among the most recent work in back-end optimization. Also, as stated in Chapter 1.2.3, the authors provide evidence that g^2o is at least as good as many other back-end optimization techniques both in accuracy and execution time.

In the results that follow and throughout the remainder of the dissertation we will forgo examining the effect of optimization on heading estimates. Our reason for this omission grows out of the way in which we develop our prior belief about global poses and loop closure transformations using Equations (3.13) and (3.14), respectively. In either case, each heading state is a linear function of other heading states. Because of this linearity nothing interesting changes to the optimized heading estimates when using different optimization approaches.

3.2.2 A Single Iteration

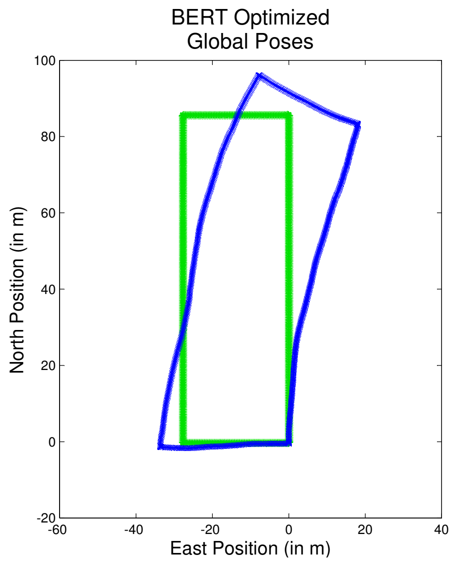

Figure 3.7: True global poses in green (gray) verses estimates from BERT as described in Algorithm 1 in blue (dark grey). Not shown is the fact that one iteration of g^{2}o returns identical global poses.

We first apply BERT according to Algorithm 1 and compare our results to one iteration of g^{2}o. Figure 3.6 presents the optimized global poses achieved by our approach. These global poses are identical to the global poses returned by the first iteration of g^{2}o when using a Gauss-Newton solver.

BERT runs in just 6.2 ms.[2] The first iteration of g^{2}o can take as much as 25.5 ms. Of this time about 10.5 ms is spent numerically approximating the Jacobian of Equation (3.11). Providing the g^{2}o with an analytical Jacobian should reduce the linearization time considerably, but subtracting all linearization time still makes the first iteration of g^{2}o around 9 ms longer than BERT. We assume that remaining difference is due to some initialization overhead that g^{2}o may require since it is a more general software tool. As will be shown below, subsequent iterations of g^{2}o take about the same amount of time as BERT.

In summary, BERT and the first iteration of g^{2}o produce the same global pose estimates in about the same time. However, the focus of BERT is optimization of the relative transformations. Accordingly, we compare the effect BERT and g^{2}o have on the relative transformations. BERT returns the optimized relative transformations without any extra computation. But, like all other back-end optimization algorithms, g^{2}o does not directly optimize transformations. Instead we must use the function $\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)$ defined in Equation (3.14) to find the new transformations that would be given by the optimized global poses g^{2}o provides.

Figure 3.7 plots the error in the estimated change of position in each relative transformation. A single iteration of g^{2}o causes the error to jump, especially near the end of the trajectory as shown in Figure 3.8. Before optimization the average error in this metric is about 2 cm. One iteration of g^{2}o causes that error to rise to about 8 cm (25% of an average transformation's change in position) near the end of the trajectory.

We should expect 735 independent odometry-like transformations to be almost unchanged by a single loop closure measurement. Figures 3.7 and 3.8 show that BERT conforms to this intuition, producing relative transformations that are only slightly adjusted from the original front-end estimates. We have also observed that the relative transformations returned from BERT using Algorithm 1 are identical to those one would find using the direct approach to optimizing the transformations described in Section 3.1.3. This bolsters our interpretation of global poses as mediating variables.

[2]For timing results we use a 1.66 GHz Intel® Atom™ CPU (using a single thread) running Ubuntu 12.04. This computer is compatible with the size, weight, and power limitations of our vehicle.

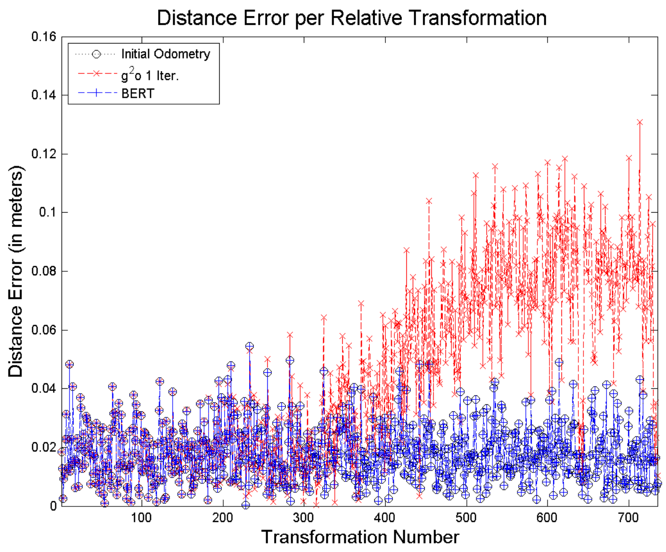

Figure 3.8: The error in distance for each relative transformation estimate. We should expect optimized estimates to change only slightly from the originals because the 735 independent odometry-like transformations are being updated with information from a single loop closure. The important feature to note is the degradation toward the end of the trajectory caused by one iteration of g^{2}o.

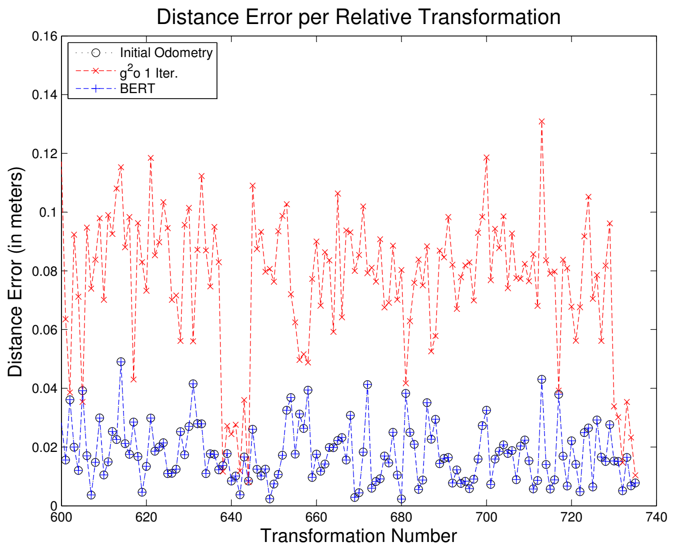

Figure 3.9: A close up of Figure 3.7 highlighting the last 135 relative transformations.

3.2.3 Further Development: Multiple Iterations

A question about BERT naturally arises from the results presented so far. The relative transformation results from g^{2}o were found using $\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)$. In a manner of speaking, this means each $\boldsymbol{\tau}_i^j$ connects $\boldsymbol{\rho}_i$ to $\boldsymbol{\rho}_j$. Since BERT returns the same global pose estimates as one iteration of g^{2}o, what is significant about the different relative transformations returned by the two algorithms?

Standard back-end optimization techniques aim to find "the most likely configuration of the [global] poses given the [relative transformations delivered by the front-end]." [6] This differs fundamentally from BERT in that BERT returns an a posteriori estimate of the joint distribution $\mathrm{p}(\boldsymbol{\tau}, \boldsymbol{\rho})$. Standard back-ends use $\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)$ to find new transformations implied by their optimized poses. Conversely, what can we learn by applying $\mathbf{g}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j\right)$ to find the new global poses implied by the optimized relative transformations BERT returns?

Figure 3.9 plots the error in position for each global pose along the trajectory. The equivalent global pose estimates from BERT using Algorithm 1 and from one iteration of g^{2}o show very little error in global position toward the end of the trajectory near the loop closure. This is intuitive given that a loop closure with the origin provides considerable information about the vehicle's true global pose.

However, if one were to recompose the relative transformation estimates returned by BERT, the error in global position at the end of the flight would still be significant (see again Figure 3.9). This observation motivates us to eliminate the global pose estimates obtained in the first application of BERT and repeat Algorithm 1. Doing so returns global pose and relative transformation estimates that are further refined. Therefore, like g^{2}o and other traditional back-ends, BERT can be iterated until the solution converges.

For the experiment in this chapter, Figure 3.10 shows that BERT converges to a global pose solution that is slightly better than g^{2}o. We expect that the modest difference is caused by slightly different linearization points used to calculate Jacobians during the composition steps of BERT (Algorithm 1, Lines 1 and 2). Figure 3.11 gives a view of the BERT solution from the same overhead perspective as earlier plots. Figure 3.12 shows the evolution of the average global position error verses computation time.

We consider these results for global pose estimates to be useful, but the driving philosophy behind BERT is a focus on relative transformations. To that end, we conclude this section with some observations on the error in relative transformation estimates.

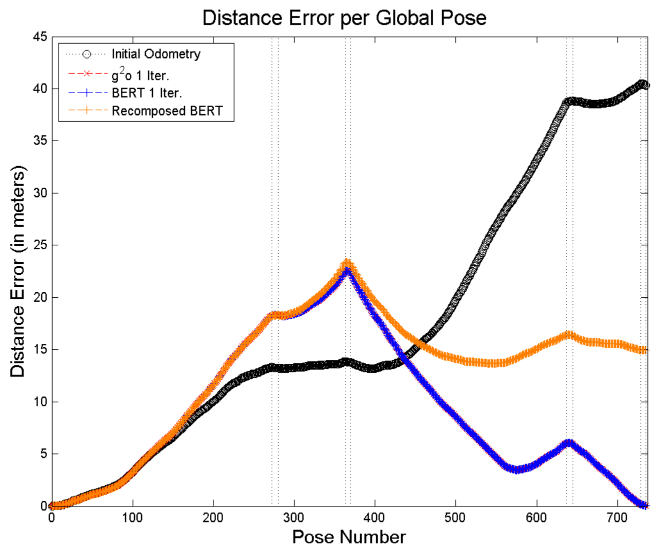

Figure 3.10: This figure shows the error in global position estimates for each saved image along the trajectory after one iteration of g^2o and BERT. The vertical dashed lines mark poses occurring in the turns of the trajectory to facilitate comparisons with Figures 3.3 , 3.6 , and 3.11. The global poses returned by one iteration of BERT and g^2o are identical. However, the global poses calculated by composing the relative transformations returned by BERT are significantly different. In particular, the global distance error at the end of the trajectory suggests the loop closure measurement might be reapplied to further improve the relative transformation estimates.

Equation (3.11), the squared Mahalanobis distance metric, measures the weighted deviation of optimized relative transformations from their initial values. Figure 3.13 (top) shows the evolution of Equation (3.11) per iteration of g^2o and BERT. Before the first iteration all of the deviation is concentrated in the difference between the measured and prior belief about the loop closure transformation. For g^2o, Equation (3.11) decreases monotonically with each iteration until it has converged to a final value. BERT converges to essentially the same value. However, in the first iteration BERT overshoots that value before settling into a steady state.

Figure 3.13 (top) also illustrates the error calculated by Equation (3.11) when the *true* relative transformations are compared to the original values. The result is two orders of magnitude higher than the minimum achieved by the optimization. The error surface defined by Equation (3.11) will not, in general, have a minimum value at the true solution. However, one can probably show that there is guaranteed to be a minimum near the true

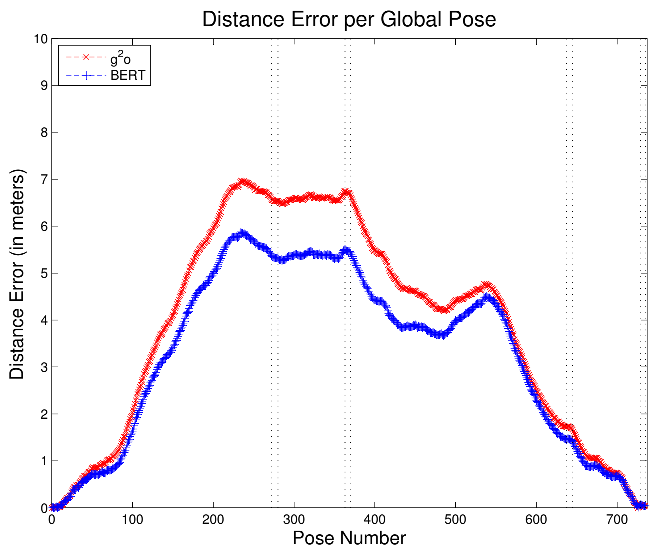

Figure 3.11: This figure shows the error in global position estimates for each saved image along the trajectory after g^2o and BERT are run to convergence. See also the caption for Figure 3.9. In the case of this example, g^2o and BERT produce final results that trend the same, but the error in the BERT solution is less than g^2o in the middle of the trajectory. We expect this modest improvement by BERT is due to different linearization points at each iteration. The difference in performance is probably problem dependent.

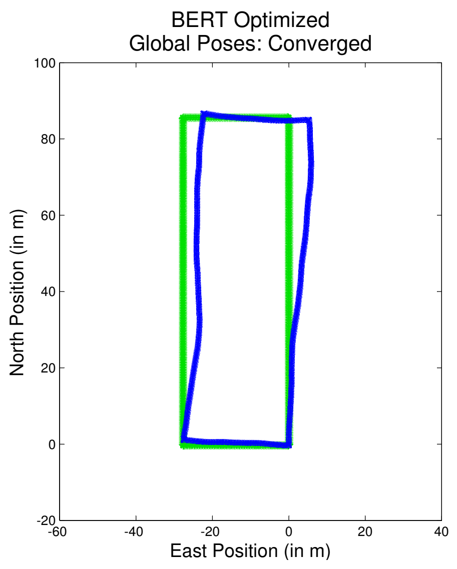

Figure 3.12: True global poses in green (gray) verses estimates from BERT run to convergence in blue (dark grey). Compare with Figures 3.3 and 3.6

solution such that the subjective quality of the map is very good. This should be more especially true as the ratio of loop closure to odometry-like transformations increases.

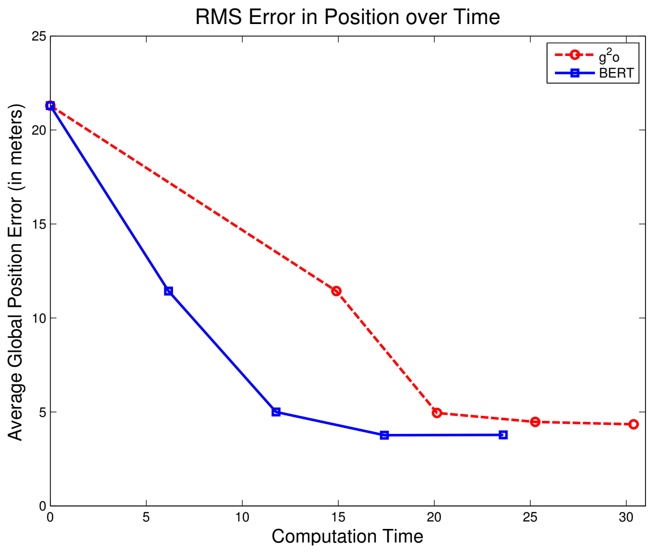

Figure 3.13: RMS Error in global position estimates as a function time accumulated over iterations of BERT and g^{2}o. Times shown for g^{2}o are given after subtracting all the linearization time out of each iteration; actual execution may be longer.

Still, the results presented in [58] are interesting on this point. They develop a map that consists of 3500 global poses, 3499 odometry-like transformations, and 2101 loop closure transformations. They show that the Multi-Level Relaxation (MLR) method developed in [52] reaches a near minimum value on this dataset after 238 iterations, taking 8.6 seconds of computation time. The authors of [58] even report that MLR "converged" to this minimum value "substantially faster" than the method proposed in [58]. Yet the subjective quality of the map given by MLR is obviously poor at that stage of optimization. MLR eventually produces[3] a "subjectively high quality map," but only after iterating for 1800 seconds of computation time [58]. This again goes to the point that Equation (3.11) is not a good measure of map quality.

The correct measure of map quality is the error calculated by comparing estimates to truth. Similar in form to Equation (1.4) we define the sum squared error

$$\bar{\epsilon}(\boldsymbol{\tau}) = \sum \left(\bar{\boldsymbol{\tau}}_i^j - \boldsymbol{\tau}_i^j\right)^\top \left(\bar{\boldsymbol{\tau}}_i^j - \boldsymbol{\tau}_i^j\right), \tag{3.15}$$

[3]See Figure 3 provided in [58]; their assessment of "subjective quality" does not require any straining at a gnat.

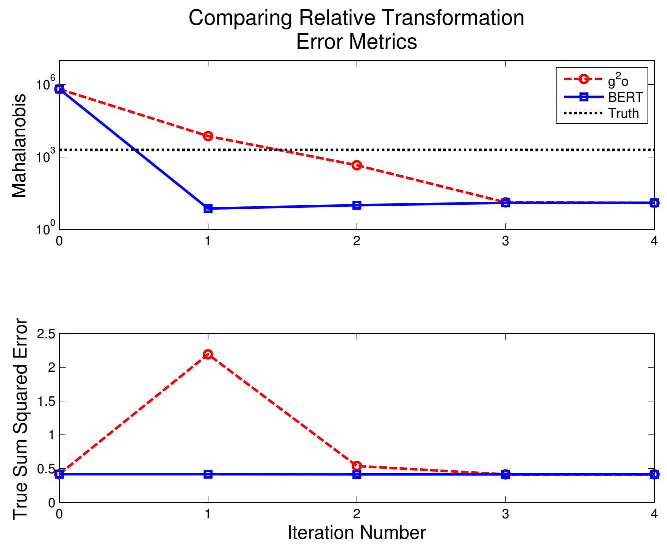

Figure 3.14: Two error metrics for relative transformations and the performance of g^{2}o and BERT against those metrics. The top plot shows the Mahalanobis distance defined by Equation (3.11). The horizontal dotted black line is the value of the Mahalanobis distance when the true relative transformations are plugged into Equation (3.11). This top plot also suggests that BERT has something like a second order response to the impulse the system experiences at loop closure. The bottom plot shows the true sum squared error of the relative transformations. For the scenario under test the loop closure should only introduce a minor improvement in the relative transformations. Both methods reflect this in their final values, but g^{2}o requires three iterations to produce a result that improves on the initial error. BERT monotonically reduces the error in the relative transformations at each iteration, eliminating undesirable transient behavior.

where $\bar{\boldsymbol{\tau}}_i^j$ indicates the true transformation from $\boldsymbol{\rho}_i$ to $\boldsymbol{\rho}_j$. Figure 3.13 (bottom) plots this metric for each iteration of g^{2}o and BERT. With g^{2}o, $\bar{\epsilon}(\boldsymbol{\tau})$ initially rises before settling into a new value that is slightly lower than before optimization. BERT, on the other hand, arrives at the same value of $\bar{\epsilon}(\boldsymbol{\tau})$ while making modest and monotonically decreasing changes at each iteration.

With each iteration BERT improves the estimates of the global poses *and* the relative transformations. In the context of relative navigation, especially given limited computing power, we consider this a noteworthy feature. It may be sufficient during online operation to conserve computing resources by running one iteration of BERT at a loop closure and waiting for additional loop closure measurements before optimizing further. Doing so with g^{2}o would be detrimental to the more immediately important relative transformation estimates.

3.3 Chapter 3 Conclusion

We recall here the context for our work. Our vehicle relies on the relative transformations for its front-end navigation. We also put a premium on computation time due to limited computational resources. We consider global pose estimates to be important but not time critical. The focus of our back-end optimization is to produce accurate relative transformations as efficiently as possible.

The contribution in this chapter is a new approach to back-end optimization, BERT, that uses properties of independence and conditional independence to efficiently estimate the joint distribution over relative transformations and global poses. We have illustrated in a simple scenario that BERT improves the estimates of the global poses *and* the relative transformations at each iteration of the algorithm. By comparison, a state-of-the-art back-end optimization tool produces similar global pose estimates at each iteration but requires multiple iterations to improve relative transformation estimates. This result is especially well-suited to our application, a small vehicle relying on relative transformations for its real-time navigation.

BERT offers other useful features. BERT naturally admits global pose *measurements* when available, e.g. from GPS. Also, no extra computation is required to provide the optimized relative transformations. Along with optimized transformation estimates BERT computes the optimized marginal covariance matrices for each of the relative transformations. Updating this measure of uncertainty is not even contemplated in other literature on back-end optimization except for [60], and there they only update a transformation's covariance matrix if the transformation is measured a second time by the front-end when the vehicle returns to the area.

To highlight another feature, it is common practice when detecting a loop closure to use the marginal covariance matrices for the two global poses involved to reject possible false positives. A place recognition algorithm may tag two reference images as sufficiently similar to warrant loop closure, but if the estimator suggests the expected value and uncertainty of the poses make the match untenable, it will be rejected. The marginal uncertainty of the global poses is always available without additional computation in BERT. Furthermore, BERT can be generally thought of as an any-time algorithm, offering a meaningful result at any iteration and any stage of Algorithm 1.

BERT, as described in this chapter, could also be applied without major modification in other scenarios where loop closure always occurs with the origin. For example, if a vehicle is navigating outdoors using a vision-based front-end approach like ours, it may receive intermittent GPS measurements. The vehicle could use each intermittent GPS measurement as a loop closure with the origin.

As another example, it is common practice to perform visual odometry (VO) with respect to a fixed reference image instead of measuring the relative transformation only between temporally successive images. It is also common to apply a windowed bundle adjustment [83] to smooth the initial VO measurements. BERT could be used in this context by treating the reference image as a temporary "global" origin.

In summary, BERT grows out of a new perspective on the back-end optimization problem. It provides the same information offered by traditional back-end optimization tools, but it also offers other benefits they do not provide. Most importantly for our overall system concept, BERT improves our estimates of the relative transformations in the map at every iteration.

Chapter 4

Extending BERT Beyond A Single Loop Closed at the Origin

In this chapter we consider how to extend the results from Chapter 3 to arbitrary map topologies. The example used in the initial development of BERT is a distinctly special case. Closing the loop with the global origin benefits from the fact that the origin is completely certain and therefore independent of all other poses. To extend BERT we need to generate loop closure hypotheses between arbitrary global poses, i.e. we need a prior belief about arbitrary modular joint distributions of the form $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$. Our prior belief about the global poses involved should treat those poses as correlated.

We begin with some discussion of the covariance matrix associated with the full joint distribution $\mathrm{p}\left(\boldsymbol{\tau}, \boldsymbol{\rho}\right)$. We then consider a scenario that is slightly adjusted from that of Chapter 3. We will close a single loop between the pose at the end of a trajectory and a pose in the middle. This scenario lets us illustrate some of the important theory behind extending BERT and serves as a stepping stone to completely general graph topologies. We then present results from running BERT in this scenario and draw some conclusions.

4.1 Covariance of the Joint Distribution

The motivation for developing BERT is to optimize relative transformations. We saw in Chapter 3.1.3 that working with the joint distribution over only the relative transformations led to all of the transformations becoming correlated after the first loop closure. We then introduced global poses into the estimation problem. Estimating the joint distribution $\mathrm{p}\left(\boldsymbol{\tau}, \boldsymbol{\rho}\right)$ allowed us to apply properties of conditional independence to efficiently propagate loop closure information.

The relative transformations are inherently independent from each other when they are estimated in the front-end. When the vehicle designates a new reference image, its pose estimate relative to that image does not depend on the other relative transformations that brought the vehicle to that point. However, our estimates of global poses all depend on the relative transformations. We develop our prior belief about the global poses by composing the odometry-like relative transformations along the trajectory using

$$\boldsymbol{\rho}_{i+1} = \mathbf{g}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}\right) = \begin{bmatrix} n_i + \Delta f_i \cos\psi_i - \Delta s_i \sin\psi_i \\ w_i + \Delta f_i \sin\psi_i + \Delta s_i \cos\psi_i \\ \psi_i + \Delta\psi_i \end{bmatrix}, \tag{4.1}$$

where[1] $\boldsymbol{\rho}_i \triangleq [n_i \; w_i \; \psi_i]^\top$ is the pose of the i^{th} reference image in the global reference frame; $\boldsymbol{\rho}_i$ is located at the north and west positions n_i and w_i with a heading of ψ_i. The odometry-like relative change in pose $\boldsymbol{\tau}_i^{i+1} \triangleq [\Delta f_i \; \Delta s_i \; \Delta\psi_i]^\top$ is defined in the local reference frame associated with $\boldsymbol{\rho}_i$ and has components that represent the change in forward and side (i.e. left) directions and heading angle, respectively. Using Equation (4.1) to develop a prior belief about $\boldsymbol{\rho}_{i+1}$ leads to a strong correlation between it and our belief about pose $\boldsymbol{\rho}_i$. To a lesser degree $\boldsymbol{\rho}_{i+1}$ is also correlated to $\boldsymbol{\tau}_i^{i+1}$

We will symbolically represent the structure of the covariance matrix of $\mathrm{p}(\boldsymbol{\tau}, \boldsymbol{\rho})$ by first defining some notation. Let $\mathbf{P}_{\boldsymbol{\rho}_i}$ be the covariance matrix associated with the marginal distribution $\mathrm{p}(\boldsymbol{\rho}_i)$. Let $\mathbf{P}_{\boldsymbol{\tau}_i^j}$ be the covariance matrix associated with the marginal distribution $\mathrm{p}(\boldsymbol{\tau}_i^j)$. Let $\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}$ be the cross-covariance matrix relating $\boldsymbol{\rho}_j$ and $\boldsymbol{\rho}_i$. Finally, let $\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}$ denote the cross-covariance relating $\boldsymbol{\rho}_j$ and $\boldsymbol{\tau}_i^j$, with a similarly subscripted matrix relating $\boldsymbol{\rho}_i$ and $\boldsymbol{\tau}_i^j$. We note that each of the just defined covariance matrices is 3-x-3 for two dimensional navigation. Using this notation we write the covariance for $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$ as

$$\mathbf{P}_{ij} = \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}^\top_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j} & \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} \\ \mathbf{P}_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix}. \tag{4.2}$$

[1]We alert the reader here that we have changed the definition of our reference frames in this chapter. The community standard is a north-west-up global frame, making a positive change in heading a counterclockwise rotation from north. We have made the switch to allow more straightforward use of publicly available back-end optimization datasets, almost all of which are developed by the ground vehicle robotics community.

The covariance matrix $\mathbf{P}$ for the full joint distribution $\mathrm{p}\left(\boldsymbol{\tau}, \boldsymbol{\rho}\right)$ is composed of several such blocks that overlap at the global poses. Consider the Chapter 3 scenario, a single loop closure from the origin to the end of the trajectory. In this case we can write our prior belief about $\mathbf{P}$ as

$$\mathbf{P} = \begin{bmatrix}
\mathbf{P}_{\boldsymbol{\rho}_1} & \mathbf{0} & \mathbf{P}^{\top}_{\boldsymbol{\rho}_2\boldsymbol{\rho}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_1^2} & \mathbf{P}^{\top}_{\boldsymbol{\rho}_2\boldsymbol{\tau}_1^2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{P}_{\boldsymbol{\rho}_2\boldsymbol{\rho}_1} & \mathbf{P}_{\boldsymbol{\rho}_2\boldsymbol{\tau}_1^2} & \mathbf{P}_{\boldsymbol{\rho}_2} & \mathbf{0} & \mathbf{P}^{\top}_{\boldsymbol{\rho}_3\boldsymbol{\rho}_3} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_2^3} & \mathbf{P}^{\top}_{\boldsymbol{\rho}_3\boldsymbol{\tau}_2^3} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_3\boldsymbol{\rho}_2} & \mathbf{P}_{\boldsymbol{\rho}_3\boldsymbol{\tau}_2^3} & \mathbf{P}_{\boldsymbol{\rho}_3} & \mathbf{0} & \mathbf{P}^{\top}_{\boldsymbol{\rho}_4\boldsymbol{\rho}_3} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_3^4} & \mathbf{P}^{\top}_{\boldsymbol{\rho}_4\boldsymbol{\tau}_3^4} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_4\boldsymbol{\rho}_3} & \mathbf{P}_{\boldsymbol{\rho}_4\boldsymbol{\tau}_3^4} & \mathbf{P}_{\boldsymbol{\rho}_4} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
 & & & & & & & \ddots & & & \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{P}_{\boldsymbol{\rho}_{\mathrm{N}-1}} & \mathbf{0} & \mathbf{P}^{\top}_{\boldsymbol{\rho}_{\mathrm{N}}\boldsymbol{\rho}_{\mathrm{N}-1}} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}} & \mathbf{P}^{\top}_{\boldsymbol{\rho}_{\mathrm{N}}\boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{P}_{\boldsymbol{\rho}_{\mathrm{N}}\boldsymbol{\rho}_{\mathrm{N}-1}} & \mathbf{P}_{\boldsymbol{\rho}_{\mathrm{N}}\boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}} & \mathbf{P}_{\boldsymbol{\rho}_{\mathrm{N}}}
\end{bmatrix},$$

where $\mathbf{0}$ represents an appropriately sized matrix of all zeros.

The covariance matrix $\mathbf{P}$ in this case has some important structure before loop closure. Every element off of the 3-x-3 block diagonal is zero. Each cross-covariance term of the form $\mathbf{P}_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j}$ is also a zero matrix, meaning the odometry-like relative transformations are also independent of the pose from which they originate. These aspects of $\mathbf{P}$ are reflected in the Bayesian network structure discussed previously in Figure 3.2.

After loop closure $\mathbf{P}$ again exhibits important structure. Our a posteriori estimate of $\mathbf{P}$ only changes from the prior estimate in the block diagonal elements. Specifically, the only zeros filled in represent the cross-covariance between a relative transformation and the pose it originates from. Therefore none of the relative transformations become correlated with each other. This allows us to marginalize the global poses found after the first iteration of BERT to find a covariance over all odometry-like relative transformations that is still block diagonal. The relative transformations remain independent because of the mediating effect of the global poses.

We motivated the iteration of BERT in Chapter 3.2.3 by observing the difference between the relative transformations estimated by the first iteration of BERT and g^{2}o. The transformations produced by these two methods were different despite the fact that BERT and g^{2}o develop the same global pose estimates. The structure of $\mathbf{P}$ provides the appropriate justification for iterating BERT.

4.2 An Arbitrary Single Loop

We first extend BERT by again treating a scenario with a single loop closure. However, this time the loop closure originates from a global pose in between the global origin and the end of the trajectory. We observed in the previous section that the covariance of $\mathrm{p}\left(\boldsymbol{\tau}, \boldsymbol{\rho}\right)$ had no non-zero elements off the block diagonal when closing the loop at the global origin. This will not be the case generally. Therefore, our primary objective in this section is to develop an efficient method for calculating the cross-covariances needed to form the loop closure hypothesis $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j\right)$. We will also consider the structure of the loop closure hypothesis covariance matrix to help us understand how to propagate that loop closure information through the rest of the network.

4.2.1 Calculating the Cross-Covariance

We recall here Algorithm 2 which was first presented in Chapter 3.1.3. As previously conceived in Chapter 3, we begin every iteration of BERT by repeating Lines 1 and 2 of Algorithm 2 to form our prior belief about the global poses. Our prior belief about the poses is based on the odometry-like transformations of the form $\boldsymbol{\tau}_i^{i+1}$. Line 1 is simply the application of Equation (4.1). Line 2 requires that we calculate the marginal uncertainty and cross-covariance of the new pose based on the uncertainty in the prior pose and the odometry-like relative transformation between them. To extend BERT, we must reconsider this portion of Algorithm 2 to also compute our prior belief about the cross-covariance between poses involved in a loop closure.

Algorithm 2: BERT using a modular representation of $\mathrm{p}\left(\boldsymbol{\tau}, \boldsymbol{\rho}\right)$ for the single loop scenario.

```
  for ( $i = 1$; $i < \mathrm{N}$; $i$++) do
1     Compose $\boldsymbol{\rho}_i$ with $\boldsymbol{\tau}_i^{i+1}$ to find $\boldsymbol{\rho}_{i+1}$.
2     Calculate the joint covariance for $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1}\right)$.
  end
3 Apply the loop closure measurement in a Kalman update of $\mathrm{p}(\check{\boldsymbol{\tau}}_0^{\mathrm{N}})$.
4 Use Equations (3.9) and (3.10) to find $\mathrm{p}(\overset{\star}{\boldsymbol{\rho}}_{\mathrm{N}-1}, \overset{\star}{\boldsymbol{\tau}}{}_{\mathrm{N}-1}^{\mathrm{N}}, \overset{\star}{\boldsymbol{\tau}}{}_0^{\mathrm{N}})$ where $\mathbf{x}_1$ corresponds to
  $\boldsymbol{\rho}_{\mathrm{N}-1}$ and $\boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}$, and $\mathbf{x}_2$ corresponds to $\boldsymbol{\tau}_0^{\mathrm{N}}$.
  for ( $i = \mathrm{N}-1$; $i > 0$; $i$--) do
5     Extract the marginal distribution $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_i\right)$ from $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_i, \overset{\star}{\boldsymbol{\tau}}{}_i^{i+1}, \overset{\star}{\boldsymbol{\rho}}_{i+1}\right)$
6     Find $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{i-1}, \overset{\star}{\boldsymbol{\tau}}{}_{i-1}^{i}, \overset{\star}{\boldsymbol{\rho}}_i\right)$ using Equations (3.9) and (3.10) where $\mathbf{x}_1$ corresponds to
      $\boldsymbol{\rho}_{i-1}$ and $\boldsymbol{\tau}_{i-1}^{i}$, and $\mathbf{x}_2$ corresponds to $\boldsymbol{\rho}_i$
  end
```

We implement Line 2 by augmenting $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1})$ with the new estimate of $\boldsymbol{\rho}_{i+1}$; i.e. we find the covariance matrix for the joint state vector

$$\begin{bmatrix} \boldsymbol{\rho}_i \\ \boldsymbol{\tau}_i^{i+1} \\ \boldsymbol{\rho}_{i+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\rho}_i \\ \boldsymbol{\tau}_i^{i+1} \\ \mathbf{g}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}\right) \end{bmatrix}. \tag{4.3}$$

We can think of this augmentation like a time propagation step in an EKF. In that sense, the states $\boldsymbol{\rho}_i$ and $\boldsymbol{\tau}_i^{i+1}$ here are static and $\boldsymbol{\rho}_{i+1}$ evolves from zero to the value given by $\mathbf{g}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}\right)$. Therefore, implementing Line 2 of Algorithm 2 can be conveniently described using matrix multiplication.

Let $\mathbf{I}_n$ indicate an n-dimensional identity matrix. We define the Jacobians of Equation (4.1) to be

$$\mathbf{J}_{\boldsymbol{\rho}_i} \triangleq \frac{\partial \mathbf{g}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}\right)}{\partial \boldsymbol{\rho}_i} = \begin{bmatrix} 1 & 0 & a_i \\ 0 & 1 & b_i \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.4}$$

$$\mathbf{J}_{\boldsymbol{\tau}_i} \triangleq \frac{\partial \mathbf{g}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}\right)}{\partial \boldsymbol{\tau}_i^{i+1}} = \begin{bmatrix} \cos\psi_i & -\sin\psi_i & 0 \\ \sin\psi_i & \cos\psi_i & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.5}$$

where

$$a_i \triangleq -\Delta f_i \sin(\psi_i) - \Delta s_i \cos(\psi_i), \tag{4.6}$$

$$b_i \triangleq \Delta f_i \cos(\psi_i) - \Delta s_i \sin(\psi_i). \tag{4.7}$$

We wish to determine the covariance matrix, $\mathbf{P}_{\text{Odo}}$, for the joint distribution $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1}\right)$. The undetermined elements of $\mathbf{P}_{\text{Odo}}$ are the marginal uncertainty $\mathbf{P}_{\boldsymbol{\rho}_{i+1}}$ and the cross-covariance matrices $\mathbf{P}_{\boldsymbol{\rho}_{i+1}\boldsymbol{\rho}_i}$ and $\mathbf{P}_{\boldsymbol{\rho}_{i+1}\boldsymbol{\tau}_i^{i+1}}$. Again referring to the analogy between time propagation and augmentation, we can find these new values based on our old values using the appropriate Jacobians:

$$\begin{aligned}
\mathbf{P}_{\text{Odo}} &\triangleq \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^{i+1}} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}\boldsymbol{\tau}_i^{i+1}}^\top \\ \mathbf{P}_{\boldsymbol{\rho}_{i+1}\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}\boldsymbol{\tau}_i^{i+1}} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{J}_{\boldsymbol{\rho}_i} & \mathbf{J}_{\boldsymbol{\tau}_i} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^{i+1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{J}_{\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{J}_{\boldsymbol{\tau}_i}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{J}_{\boldsymbol{\rho}_i} & \mathbf{J}_{\boldsymbol{\tau}_i} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^{i+1}} & \mathbf{P}_{\boldsymbol{\tau}_i^{i+1}}\mathbf{J}_{\boldsymbol{\tau}_i}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}
\end{aligned}$$

$$= \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^{i+1}} & \mathbf{P}_{\boldsymbol{\tau}_i^{i+1}}\mathbf{J}_{\boldsymbol{\tau}_i}^\top \\ \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{J}_{\boldsymbol{\tau}_i}\mathbf{P}_{\boldsymbol{\tau}_i^{i+1}} & \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top + \mathbf{J}_{\boldsymbol{\tau}_i}\mathbf{P}_{\boldsymbol{\tau}_i^{i+1}}\mathbf{J}_{\boldsymbol{\tau}_i}^\top \end{bmatrix}. \tag{4.8}$$

Equation (4.8) represents the standard result we would have previously found at the end of one step through Lines 1 and 2 in Algorithm 2. Modifying the algorithm for general loop closure hypotheses begins here. We previously would have marginalized out $\boldsymbol{\rho}_i$ and $\boldsymbol{\tau}_i^{i+1}$, and then used $\boldsymbol{\rho}_{i+1}$ and the independent transformation $\boldsymbol{\tau}_{i+1}^{i+2}$ as our only inputs to Equation (4.1) in the next step through Lines 1 and 2. But suppose we need to generate a loop closure hypothesis using $\boldsymbol{\rho}_i$ and some temporally distant pose $\boldsymbol{\rho}_j$. In that case, we should not marginalize $\boldsymbol{\rho}_i$. Rather, we should retain our estimate of $\boldsymbol{\rho}_i$ in the jointly distributed random vector until we compose the network up to $\boldsymbol{\rho}_j$. Retaining $\boldsymbol{\rho}_i$ in the random vector will allow us to calculate the cross-covariance between $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$.

Therefore, after computing our prior belief about $\boldsymbol{\rho}_{i+1}$ we marginalize only the relative transformation, leaving the joint distribution $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\rho}_{i+1})$ with covariance matrix

$$\begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top \\ \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}} \end{bmatrix}.$$

We augment this distribution with the next odometry-like transformation so that we have $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\rho}_{i+1}, \boldsymbol{\tau}_{i+1}^{i+2})$ with covariance matrix

$$\begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top & \mathbf{0} \\ \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}} \end{bmatrix}.$$

This augmentation is trivial since $\boldsymbol{\tau}_{i+1}^{i+2}$ is independent of the two global poses in the joint distribution. We must now augment the distribution with the next global pose so that we have $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\rho}_{i+1}, \boldsymbol{\tau}_{i+1}^{i+2}, \boldsymbol{\rho}_{i+2})$. The covariance, $\mathbf{P}_{\mathrm{Odo}^{++}}$, for this distribution can be described with a process similar to Equation (4.8):

$$\begin{aligned}
\mathbf{P}_{\text{Odo}^{++}} &\triangleq \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_{i+2}\boldsymbol{\rho}_i}^\top \\ \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_{i+2}\boldsymbol{\rho}_{i+1}}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}} & \mathbf{P}_{\boldsymbol{\rho}_{i+2}\boldsymbol{\tau}_{i+1}^{i+2}}^\top \\ \mathbf{P}_{\boldsymbol{\rho}_{i+2}\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_{i+2}\boldsymbol{\rho}_{i+1}} & \mathbf{P}_{\boldsymbol{\rho}_{i+2}\boldsymbol{\tau}_{i+1}^{i+2}} & \mathbf{P}_{\boldsymbol{\rho}_{i+2}} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\boldsymbol{\rho}_{i+1}} & \mathbf{J}_{\boldsymbol{\tau}_{i+1}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{J}_{\boldsymbol{\rho}_{i+1}}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{J}_{\boldsymbol{\tau}_{i+1}}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\boldsymbol{\rho}_{i+1}} & \mathbf{J}_{\boldsymbol{\tau}_{i+1}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top\mathbf{J}_{\boldsymbol{\rho}_{i+1}}^\top \\ \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}}\mathbf{J}_{\boldsymbol{\rho}_{i+1}}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}} & \mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}}\mathbf{J}_{\boldsymbol{\rho}_{i+1}}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\boldsymbol{\rho}_i}^\top\mathbf{J}_{\boldsymbol{\rho}_{i+1}}^\top \\ \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_{i+1}}\mathbf{J}_{\boldsymbol{\rho}_{i+1}}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}} & \mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}}\mathbf{J}_{\boldsymbol{\tau}_{i+1}}^\top \\ \mathbf{J}_{\boldsymbol{\rho}_{i+1}}\mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{J}_{\boldsymbol{\rho}_{i+1}}\mathbf{P}_{\boldsymbol{\rho}_{i+1}} & \mathbf{J}_{\boldsymbol{\tau}_{i+1}}\mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}} & \mathbf{P}_{\boldsymbol{\rho}_{i+2}} \end{bmatrix}, \qquad (4.9)
\end{aligned}$$

where $\mathbf{P}_{\boldsymbol{\rho}_{i+2}} = \mathbf{J}_{\boldsymbol{\rho}_{i+1}}\mathbf{P}_{\boldsymbol{\rho}_{i+1}}\mathbf{J}_{\boldsymbol{\rho}_{i+1}}^\top + \mathbf{J}_{\boldsymbol{\tau}_{i+1}}\mathbf{P}_{\boldsymbol{\tau}_{i+1}^{i+2}}\mathbf{J}_{\boldsymbol{\tau}_{i+1}}^\top$.

There are some important aspects to notice in Equation (4.9). As should be expected, our marginal belief about $\mathrm{p}(\boldsymbol{\rho}_i)$ does not change. More importantly, comparing Equations (4.8) and (4.9) shows the bottom-right block 3-x-3 matrix in Equation (4.9) is the same as it would be had we calculated the covariance of $\mathrm{p}(\boldsymbol{\rho}_{i+1}, \boldsymbol{\tau}_{i+1}^{i+2}, \boldsymbol{\rho}_{i+2})$ without carrying along $\boldsymbol{\rho}_i$. Finally, the cross-covariance between $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_{i+2}$ (and by induction $\boldsymbol{\rho}_{i+3}$, $\boldsymbol{\rho}_{i+4}$, etc.) is a function of the Jacobians already calculated in the regular process of developing our prior belief about the poses. Therefore, the structure of the problem makes it is sufficient to save the Jacobians found by Equation (4.4) during our regular repetition of Lines 1 and 2 in Algorithm 2. To find $\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}$ we do not need to actually include extra blocks in calculations like Equation (4.9).

According to the foregoing analysis we need to compute $\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} = \mathbf{J}_{\boldsymbol{\rho}_{j-1}} \cdots \mathbf{J}_{\boldsymbol{\rho}_{i+1}}\mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i}$. We can further simplify this calculation by looking at the specific structure of the Jacobian given by Equation (4.4). Multiplying two matrices with this form gives

$$\begin{aligned}\mathbf{J}_{\boldsymbol{\rho}_{i+1}}\mathbf{J}_{\boldsymbol{\rho}_i} &= \begin{bmatrix}1 & 0 & a_{i+1}\\ 0 & 1 & b_{i+1}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & a_i\\ 0 & 1 & b_i\\ 0 & 0 & 1\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & (a_i + a_{i+1})\\ 0 & 1 & (b_i + b_{i+1})\\ 0 & 0 & 1\end{bmatrix}.\end{aligned} \tag{4.10}$$

Therefore, defining $\mathbf{J}_{ji} \triangleq \mathbf{J}_{\boldsymbol{\rho}_{j-1}} \cdots \mathbf{J}_{\boldsymbol{\rho}_{i+1}}\mathbf{J}_{\boldsymbol{\rho}_i}$, we can calculate the cross-covariance $\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}$ by saving only two scalar elements from each of the several Jacobians and performing one matrix multiplication:

$$\begin{aligned}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} &= \mathbf{J}_{ji}\mathbf{P}_{\boldsymbol{\rho}_i}\\ &= \begin{bmatrix}1 & 0 & \sum_{k=i}^{j-1} a_k\\ 0 & 1 & \sum_{k=i}^{j-1} b_k\\ 0 & 0 & 1\end{bmatrix}\mathbf{P}_{\boldsymbol{\rho}_i}.\end{aligned} \tag{4.11}$$

The ramifications of the foregoing analysis are significant. In Chapter 3.1.4 we discussed the similarities between BERT and the EKF-SLAM submapping approach of [27]. We noted one significant difference to be the extra map elements required by [27] to maintain the conditional independence between their feature-based SLAM submaps. The authors of [27] called these elements "overhead," and they consisted of features and robot poses that were not indigenous to a given submap.

The submaps of [27] correspond to modular distributions $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1})$. During our earlier development described in Chapter 3, we wondered if the estimate of $\boldsymbol{\rho}_i$ may need to be included in each modular joint distribution along the chain of distributions until we reached $\boldsymbol{\rho}_j$. This would be analogous to the overhead described in [27]. Including additional poses in the modular joint distributions would have caused them to quickly grow with the number of loop closures, destroying the efficiency of BERT. Thankfully, we need not include any extra elements in the modules that make up our full joint distribution, and the extra computation needed to generate an arbitrary loop closure hypothesis is negligible.

4.2.2 Structure of the Loop Closure Hypothesis

In the foregoing discussion we showed how to calculate the cross-covariance between $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$, and therefore we can now define the distribution $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j)$. We can in turn use our prior belief about $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j)$ to develop our belief about the loop closure transformation. Recall the measurement equation from Chapter 3.2.1:

$$\begin{aligned}\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right) &= \boldsymbol{\tau}_i^j. \\ &= \begin{bmatrix} (n_j - n_i)\cos\psi_i + (w_j - w_i)\sin\psi_i \\ -(n_j - n_i)\sin\psi_i + (w_j - w_i)\cos\psi_i \\ \psi_j - \psi_i \end{bmatrix}. \end{aligned} \tag{4.12}$$

We use Equation (4.12) to calculate the mean of our prior belief about the loop closure transformation. To calculate the covariance we define the necessary Jacobians of Equation (4.12) to be

$$\begin{aligned}\mathbf{J}_{\mathrm{LC}_i} &\triangleq \frac{\partial \mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)}{\partial \boldsymbol{\rho}_i} \\ &= \begin{bmatrix} -\cos(\psi_i) & -\sin(\psi_i) & -\Delta n \sin(\psi_i) + \Delta w \cos(\psi_i) \\ \sin(\psi_i) & -\cos(\psi_i) & -\Delta n \cos(\psi_i) - \Delta w \sin(\psi_i) \\ 0 & 0 & -1 \end{bmatrix}, \end{aligned} \tag{4.13}$$

and

$$\mathbf{J}_{\mathrm{LC}_j} \triangleq \frac{\partial \mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)}{\partial \boldsymbol{\rho}_j} = \begin{bmatrix} \cos(\psi_i) & \sin(\psi_i) & 0 \\ -\sin(\psi_i) & \cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.14}$$

where $\vartriangle n = (n_j - n_i)$ and $\vartriangle w = (w_j - w_i)$.

Similar to before, we can find the joint covariance, $\mathbf{P}_{\mathrm{LC}}$, for the loop closure hypothesis $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$ using

$$\begin{aligned}
\mathbf{P}_{\mathrm{LC}} &\triangleq \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}^\top_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j} & \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} \\ \mathbf{P}_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{\mathrm{LC}_i} & \mathbf{0} & \mathbf{J}_{\mathrm{LC}_j} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & \mathbf{J}^\top_{\mathrm{LC}_i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}^\top_{\mathrm{LC}_j} & \mathbf{I}_3 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{\mathrm{LC}_i} & \mathbf{0} & \mathbf{J}_{\mathrm{LC}_j} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_j} & \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}^\top_{\mathrm{LC}_j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_j} & \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} \\ \mathbf{J}_{\mathrm{LC}_i}\mathbf{P}_{\boldsymbol{\rho}_i} + \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{J}_{\mathrm{LC}_i}\mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} + \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}^\top_{\mathrm{LC}_j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix},
\end{aligned} \tag{4.15}$$

where $\mathbf{P}_{\boldsymbol{\tau}_i^j} = \mathbf{J}_{\mathrm{LC}_i}(\mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_j}) + \mathbf{J}_{\mathrm{LC}_j}(\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}^\top_{\mathrm{LC}_j})$. Though not immediately obvious, $\mathbf{P}_{\mathrm{LC}}$ also exhibits significant structure that helps us understand how loop closure information will propagate to other variables.

We will first consider $\mathbf{P}_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j}$, the cross-covariance between the originating pose of the loop closure, $\boldsymbol{\rho}_i$, and the loop closure transformation, $\boldsymbol{\tau}_i^j$. To keep expressions compact[2] we introduce the notation $\mathrm{c}_{\psi_i} \triangleq \cos(\psi_i)$ and $\mathrm{s}_{\psi_i} \triangleq \sin(\psi_i)$. Then observe that $\mathbf{P}_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j}$ can be manipulated using Equation (4.11) and the definitions of the various Jacobians such that

$$\begin{aligned}
\mathbf{P}_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j} &= \mathbf{J}_{\mathrm{LC}_i}\mathbf{P}_{\boldsymbol{\rho}_i} + \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} \\
&= \mathbf{J}_{\mathrm{LC}_i}\mathbf{P}_{\boldsymbol{\rho}_i} + \mathbf{J}_{\mathrm{LC}_j}\mathbf{J}_{ji}\mathbf{P}_{\boldsymbol{\rho}_i} \\
&= \begin{bmatrix} -\mathrm{c}_{\psi_i} & -\mathrm{s}_{\psi_i} & -\Delta n \mathrm{s}_{\psi_i} + \Delta w \mathrm{c}_{\psi_i} \\ \mathrm{s}_{\psi_i} & -\mathrm{c}_{\psi_i} & -\Delta n \mathrm{c}_{\psi_i} - \Delta w \mathrm{s}_{\psi_i} \\ 0 & 0 & -1 \end{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} + \\
&\qquad \begin{bmatrix} \mathrm{c}_{\psi_i} & \mathrm{s}_{\psi_i} & 0 \\ -\mathrm{s}_{\psi_i} & \mathrm{c}_{\psi_i} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \sum_{k=i}^{j-1} a_k \\ 0 & 1 & \sum_{k=i}^{j-1} b_k \\ 0 & 0 & 1 \end{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} \\
&= \left(\begin{bmatrix} -\mathrm{c}_{\psi_i} & -\mathrm{s}_{\psi_i} & -\Delta n \mathrm{s}_{\psi_i} + \Delta w \mathrm{c}_{\psi_i} \\ \mathrm{s}_{\psi_i} & -\mathrm{c}_{\psi_i} & -\Delta n \mathrm{c}_{\psi_i} - \Delta w \mathrm{s}_{\psi_i} \\ 0 & 0 & -1 \end{bmatrix} + \right. \\
&\qquad \left. \begin{bmatrix} \mathrm{c}_{\psi_i} & \mathrm{s}_{\psi_i} & \mathrm{c}_{\psi_i} \sum_{k=i}^{j-1} a_k + \mathrm{s}_{\psi_i} \sum_{k=i}^{j-1} b_k \\ -\mathrm{s}_{\psi_i} & \mathrm{c}_{\psi_i} & -\mathrm{s}_{\psi_i} \sum_{k=i}^{j-1} a_k + \mathrm{c}_{\psi_i} \sum_{k=i}^{j-1} b_k \\ 0 & 0 & 1 \end{bmatrix} \right) \mathbf{P}_{\boldsymbol{\rho}_i}.
\end{aligned} \tag{4.16}$$

The last line in Equation (4.16) suggests a relationship between the two matrices in parentheses. The upper-left 2-x-2 block and the bottom row of each matrix is the negative of the other matrix. Also, the first two elements of the last column in each matrix represent a rotated vector. In the case of the matrix on the left, the original vector is $[\Delta n \; \Delta w]^\top$. In the case of the matrix on the right, the original vector is $\left[\left(\sum_{k=i}^{j-1} a_k\right) \; \left(\sum_{k=i}^{j-1} b_k\right)\right]^\top$.

[2]This is an unfortunate necessity; the reader should beware of confusing Δs with s_{ψ_i} when they appear together.

We can decompose $[\Delta n \; \Delta w]^\top$ by observing that $\boldsymbol{\rho}_j$ can be found by repeatedly applying Equation (4.1) starting from $\boldsymbol{\rho}_i$. This leads to

$$\begin{aligned}\begin{bmatrix}\Delta n\\ \Delta w\end{bmatrix} &= \begin{bmatrix}n_j\\ w_j\end{bmatrix} - \begin{bmatrix}n_i\\ w_i\end{bmatrix}\\ &= \left(\begin{bmatrix}n_i\\ w_i\end{bmatrix} + \mathbf{R}_{\psi_i}\begin{bmatrix}\Delta f_i\\ \Delta s_i\end{bmatrix} + \mathbf{R}_{\psi_{i+1}}\begin{bmatrix}\Delta f_{i+1}\\ \Delta s_{i+1}\end{bmatrix} + \cdots + \mathbf{R}_{\psi_{j-1}}\begin{bmatrix}\Delta f_{j-1}\\ \Delta s_{j-1}\end{bmatrix}\right) - \begin{bmatrix}n_i\\ w_i\end{bmatrix},\end{aligned}$$

where

$$\mathbf{R}_{\psi_i} \triangleq \begin{bmatrix}\mathrm{c}_{\psi_i} & -\mathrm{s}_{\psi_i}\\ \mathrm{s}_{\psi_i} & \mathrm{c}_{\psi_i}\end{bmatrix},$$

leading further to

$$\begin{aligned}\begin{bmatrix}\Delta n\\ \Delta w\end{bmatrix} &= \begin{bmatrix}\sum_{k=i}^{j-1}\left(\Delta f_k \mathrm{c}_{\psi_k} - \Delta s_k \mathrm{s}_{\psi_k}\right)\\ \sum_{k=i}^{j-1}\left(\Delta f_k \mathrm{s}_{\psi_k} + \Delta s_k \mathrm{c}_{\psi_k}\right)\end{bmatrix}\\ &= \begin{bmatrix}\sum_{k=i}^{j-1} b_k\\ -\sum_{k=i}^{j-1} a_k\end{bmatrix}.\end{aligned} \tag{4.17}$$

The last equality follows from the definitions given in Equations (4.6) and (4.7). Using Equation (4.17) in Equation (4.16) allows us to draw the important conclusion that

$$\mathbf{P}_{\boldsymbol{\rho}_i \boldsymbol{\tau}_i^j} = \mathbf{0}. \tag{4.18}$$

In words, our prior belief about the loop closure transformation $\boldsymbol{\tau}_i^j$ is independent of our prior belief about the pose $\boldsymbol{\rho}_i$ from which it originates. This has intuitive appeal. The loop closure transformation $\boldsymbol{\tau}_i^j$ is defined in the reference frame of $\boldsymbol{\rho}_i$. By identifying them as independent we are saying we could move $\boldsymbol{\rho}_i$ anywhere without affecting our belief about $\boldsymbol{\tau}_i^j$.

We pause here to make a parenthetical comment on the relationship between Equation (4.17) and Equation (4.11) from the previous subsection. Equation (4.17) suggests we could further simplify our approach to calculating the cross-covariance between the poses involved in the loop closure hypothesis. Instead of saving a_k and b_k from each of the several Jacobians as suggested by Equation (4.11), we could simply use the difference between the estimated north and west positions of $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$. We will discuss further in Chapter 4.3 and Chapter 5 why this is a bad idea. Suffice it to say for now that we will want to adjust the a_k and b_k elements from each of the Jacobians individually based on additional criteria. We therefore leave Equation (4.11) as our preferred expression for $\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}$.

We now apply the result from Equation (4.18) in Equation (4.15) to simplify our prior belief about the uncertainty of the loop closure hypothesis $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$:

$$\mathbf{P}_{\mathrm{LC}} = \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} \\ \mathbf{0} & \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}^\top_{\mathrm{LC}_j} & \mathbf{J}_{\mathrm{LC}_i}\mathbf{P}^\top_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} + \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}^\top_{\mathrm{LC}_j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix}. \tag{4.19}$$

This expression for $\mathbf{P}_{\mathrm{LC}}$ begins to highlight its structure, but we can go further.

Consider the center block of $\mathbf{P}_{\mathrm{LC}}$. This is $\mathbf{P}_{\boldsymbol{\tau}_i^j}$, the marginal uncertainty associated with the loop closure transformation $\boldsymbol{\tau}_i^j$. Using Equation (4.12) we derive our belief about $\boldsymbol{\tau}_i^j$ from the two mutually dependent random vectors $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$. This contrasts with what happens when we use Equation (4.1) to compose the mutually independent random vectors $\boldsymbol{\rho}_i$ and $\boldsymbol{\tau}_i^{i+1}$ to derive $\boldsymbol{\rho}_{i+1}$. In this latter case we find that the marginal uncertainty of $\boldsymbol{\rho}_{i+1}$ is the sum of the marginal uncertainties (mapped through appropriate Jacobians) associated with $\boldsymbol{\rho}_i$ and $\boldsymbol{\tau}_i^{i+1}$. In other words, we find that $\mathbf{P}_{\boldsymbol{\rho}_{i+1}} = \mathbf{J}_{\boldsymbol{\rho}_i}\mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}^\top_{\boldsymbol{\rho}_i} + \mathbf{J}_{\boldsymbol{\tau}_i}\mathbf{P}_{\boldsymbol{\tau}_i^{i+1}}\mathbf{J}^\top_{\boldsymbol{\tau}_i}$ (see Equation (4.8)).

We will now show that $\mathbf{P}_{\boldsymbol{\tau}_i^j}$ can also be manipulated so that it is more intuitively expressed in terms of the marginal uncertainties $\mathbf{P}_{\boldsymbol{\rho}_i}$ and $\mathbf{P}_{\boldsymbol{\rho}_j}$. From Equation (4.19),

$$\mathbf{P}_{\boldsymbol{\tau}_i^j} = \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}^\top_{\mathrm{LC}_i} + \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}^\top_{\mathrm{LC}_j}.$$

The second term on the right hand side is already in our desired format, so we turn attention to $\mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}_{\mathrm{LC}_i}^\top$. From Equation (4.11) we note that $\mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}_{\mathrm{LC}_i}^\top = \mathbf{J}_{\mathrm{LC}_j}\mathbf{J}_{ji}\mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\mathrm{LC}_i}^\top$. Focusing on $\mathbf{J}_{\mathrm{LC}_j}\mathbf{J}_{ji}$ for a moment we see that it can be manipulated using earlier results to find

$$\begin{aligned}
\mathbf{J}_{\mathrm{LC}_j}\mathbf{J}_{ji} &= \begin{bmatrix} \cos(\psi_i) & \sin(\psi_i) & 0 \\ -\sin(\psi_i) & \cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \sum_{k=i}^{j-1} a_k \\ 0 & 1 & \sum_{k=i}^{j-1} b_k \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos(\psi_i) & \sin(\psi_i) & 0 \\ -\sin(\psi_i) & \cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\Delta w \\ 0 & 1 & \Delta n \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos(\psi_i) & \sin(\psi_i) & \Delta n \sin(\psi_i) - \Delta w \cos(\psi_i) \\ -\sin(\psi_i) & \cos(\psi_i) & \Delta n \cos(\psi_i) + \Delta w \sin(\psi_i) \\ 0 & 0 & 1 \end{bmatrix} \\
&= -\mathbf{J}_{\mathrm{LC}_i}.
\end{aligned} \tag{4.20}$$

Using Equation (4.20) we can now write the expression

$$\mathbf{P}_{\boldsymbol{\tau}_i^j} = \mathbf{J}_{\mathrm{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}_{\mathrm{LC}_j}^\top - \mathbf{J}_{\mathrm{LC}_i}\mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\mathrm{LC}_i}^\top. \tag{4.21}$$

Equation (4.21) allows for an intuitive description of the loop closure transformation's marginal uncertainty that confirms the intuition associated with Equation (4.18). The uncertainty in $\mathrm{p}(\boldsymbol{\rho}_j)$ already depends in part on the uncertainty in $\mathrm{p}(\boldsymbol{\rho}_i)$. Since $\boldsymbol{\tau}_i^j$ is defined in the reference frame of $\boldsymbol{\rho}_i$, the uncertainty in $\mathrm{p}(\boldsymbol{\tau}_i^j)$ arises from the uncertainty in $\mathrm{p}(\boldsymbol{\rho}_j)$ less the contribution from $\mathrm{p}(\boldsymbol{\rho}_i)$.

Using the preceding development we can express $\mathbf{P}_{\text{LC}}$, the uncertainty in the loop closure hypothesis $\text{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$, as

$$\mathbf{P}_{\text{LC}} = \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}^{\top} \\ \mathbf{0} & \mathbf{J}_{\text{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}_{\text{LC}_j}^{\top} - \mathbf{J}_{\text{LC}_i}\mathbf{P}_{\boldsymbol{\rho}_i}\mathbf{J}_{\text{LC}_i}^{\top} & \mathbf{J}_{\text{LC}_j}\mathbf{P}_{\boldsymbol{\rho}_j} + \mathbf{J}_{\text{LC}_i}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}^{\top} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{J}_{\text{LC}_j}^{\top} + \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}\mathbf{J}_{\text{LC}_i}^{\top} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix}. \tag{4.22}$$

4.2.3 Propagating Loop Closure Information

Now that we can generate loop closure hypotheses and understand the structure in their covariance matrices, we are prepared to examine how loop closure information will propagate through the network. In the initial application of BERT in Chapter 3 we treated the loop closure estimate as a measurement in a Kalman update. The update only affected those variables at the end of the trajectory because the global origin, at the other end of the loop closure, was anchored in its place. We now consider what happens if we again use a Kalman update to apply the loop closure estimate between two uncertain global poses.

We begin by examining the form of the Kalman gain. The random vector involved in the update is

$$\mathbf{x}_{\text{LC}} \triangleq \begin{bmatrix} \boldsymbol{\rho}_i \\ \boldsymbol{\tau}_i^j \\ \boldsymbol{\rho}_j \end{bmatrix}.$$

The measurement is simply one of the states, so the measurement model is defined as

$$\mathbf{H} \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \end{bmatrix}.$$

The innovation covariance, $\mathbf{S}$, in this case is the sum of the prior covariance $\mathbf{P}_{\boldsymbol{\tau}_i^j}$ developed above and the uncertainty in the loop closure as estimated in the front-end. Then the Kalman gain, $\mathbf{L}$, can be written as

$$\mathbf{L} \triangleq \mathbf{P}_{\text{LC}}\mathbf{H}^{\top}\mathbf{S}^{-1}$$

$$= \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}^\top \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_3 \\ \mathbf{0} \end{bmatrix} \mathbf{S}^{-1}$$

$$= \begin{bmatrix} \mathbf{0} \\ \mathbf{P}_{\boldsymbol{\tau}_i^j}\mathbf{S}^{-1} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}\mathbf{S}^{-1} \end{bmatrix}. \tag{4.23}$$

The important feature in the Kalman gain is the zero matrix in the upper 3-x-3 block. We can see the way that this zero matrix affects the flow of loop closure information by examining the corrections applied to the state and covariance in the update. Let $\boldsymbol{\eta}$ denote the innovation. We use the notation $\mathbf{x}_{\mathrm{LC}}^+$ and $\mathbf{P}_{\mathrm{LC}}^+$ to denote the updated values, and we see that

$$\begin{aligned} \mathbf{x}_{\mathrm{LC}}^+ &= \mathbf{x}_{\mathrm{LC}} + \mathbf{L}\boldsymbol{\eta} \\ &= \begin{bmatrix} \boldsymbol{\rho}_i \\ \boldsymbol{\tau}_i^j \\ \boldsymbol{\rho}_j \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{P}_{\boldsymbol{\tau}_i^j}\mathbf{S}^{-1} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}\mathbf{S}^{-1} \end{bmatrix} \boldsymbol{\eta}, \end{aligned} \tag{4.24}$$

$$\begin{aligned} \mathbf{P}_{\mathrm{LC}}^+ &= \mathbf{P}_{\mathrm{LC}} - \mathbf{L}\mathbf{H}\mathbf{P}_{\mathrm{LC}} \\ &= \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}^\top \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{P}_{\boldsymbol{\tau}_i^j}\mathbf{S}^{-1} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}\mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}^\top \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}^\top \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{P}_{\boldsymbol{\tau}_i^j}\mathbf{S}^{-1} \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}\mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}^\top \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{P}_{\boldsymbol{\rho}_i} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}^\top \\ \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix} - \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\tau}_i^j}\mathbf{S}^{-1}\mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\tau}_i^j}\mathbf{S}^{-1}\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}^\top \\ \mathbf{0} & \mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\tau}_i^j}\mathbf{S}^{-1}\mathbf{P}_{\boldsymbol{\tau}_i^j} & \mathbf{P}_{\boldsymbol{\rho}_j}\mathbf{S}^{-1}\mathbf{P}_{\boldsymbol{\rho}_j} \end{bmatrix}. \end{aligned} \tag{4.25}$$

Note that the prior mean value, the marginal covariance, *and* the cross-covariances associated with the loop closure origin $\boldsymbol{\rho}_i$ remain unchanged by the update. The fact that the marginal distribution $\mathrm{p}(\boldsymbol{\rho}_i)$ remains unchanged comports with the intuition from Equation (4.18). Since they begin as independent, changing the value of $\boldsymbol{\tau}_i^j$ should not affect our marginal belief about $\boldsymbol{\rho}_i$ any more than changing $\boldsymbol{\rho}_i$ should affect our marginal belief about $\boldsymbol{\tau}_i^j$.

It is more interesting to emphasize that the cross-covariances $\mathbf{P}_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j}$ and $\mathbf{P}_{\boldsymbol{\rho}_j\boldsymbol{\rho}_i}$ remain unchanged by the update. The cross-covariance $\mathbf{P}_{\boldsymbol{\rho}_i\boldsymbol{\tau}_i^j}$ was zero before the update, so $\boldsymbol{\rho}_i$ and $\boldsymbol{\tau}_i^j$ remain independent after the update. For the relationship between $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$ we recall from probability theory that the correlation coefficient for two scalar random variables is defined to be their covariance divided by the product of their standard deviations. The cross-covariance and marginal uncertainty associated with $\boldsymbol{\rho}_i$ remain unchanged by the update, but the marginal uncertainty of $\boldsymbol{\rho}_j$ gets smaller. Therefore, the update causes the elements of $\boldsymbol{\rho}_i$ to become more strongly correlated with $\boldsymbol{\rho}_j$. This result motivates an area of future work sketched out in Chapter 5 where we discuss the often cited analogy between back-end optimization and a mass-spring system.

4.2.4 Summarizing Insights From an Arbitrary Single Loop Closure

We have shown that an arbitrary loop closure hypothesis $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$ can be easily generated from values already calculated in the regular process of developing our prior belief about the global poses. We have also seen that, because of the structure in the covariance of $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j)$, updating our belief about the loop closure transformation does not affect our belief about the originating pose $\boldsymbol{\rho}_i$ in the loop closure. This suggests that we can update the entire network of modular joint distributions in this chapter's single loop example using the same approach used in Chapter 3. We will start our presentation of results by applying a slight variation of Algorithm 2 from Chapter 3, but the results will suggest some modifications that refine our understanding of how to apply the foregoing analysis. The results also illustrate some important considerations for further extending BERT to arbitrary networks.

4.3 Testing an Arbitrary Single Loop Closure

4.3.1 Test Scenario

To extend BERT we generated a new dataset with a more complex set of attributes. The data again simulates a robot moving in two dimensions through the hallways of a building. The trajectory includes some long stretches without loop closures as well as a few stretches through hallways with several loop closures. The stretches without loop closure allow the uncertainty in global poses to become large. Multiple loop closures offer the option of considering how loop closure information propagates in more complicated scenarios.

The trajectory in this dataset is about 280 meters long. New reference image poses are created at about every 0.5 meters change in position or 10 degrees change in heading, whichever comes first. This leads to 623 reference poses, 622 odometry-like transformations, and 64 loop closure transformations. All relative transformations in this dataset are corrupted by independent, zero mean Gaussian noise with a standard deviation of 10 cm for each component of position and 2 degrees for heading. We note that, compared to Chapter 3, the error on position is a significantly higher percentage of the actual change. The dataset is based on a path through the fourth floor of the Brigham Young University Clyde Building and is visualized in Figure 4.0

To test a single loop closure scenario we truncate the trajectory at pose 572 where the vehicle is just about 25 meters straight down the hallway west from the global origin. By this point the vehicle has traveled about 255 meters. The single loop closure in the test occurs with pose 144, a pose about 51 meters along the trajectory from its beginning. Figure 4.1 illustrates the initially estimated trajectory based on unoptimized odometry-like transformations.

We use an off-the-shelf Matlab implementation[3] of the Levenburg-Marquardt (LM) algorithm as a baseline to compare our results against. We used this same algorithm during the early development of Chapter 3. It provided the same results as those returned by g^{2}o, though at considerably longer execution times. In this chapter we choose to do our

[3]Available at http://www.mathworks.com/matlabcentral/fileexchange/17534-lmfnlsq-solution-of-nonlinear-least-squares

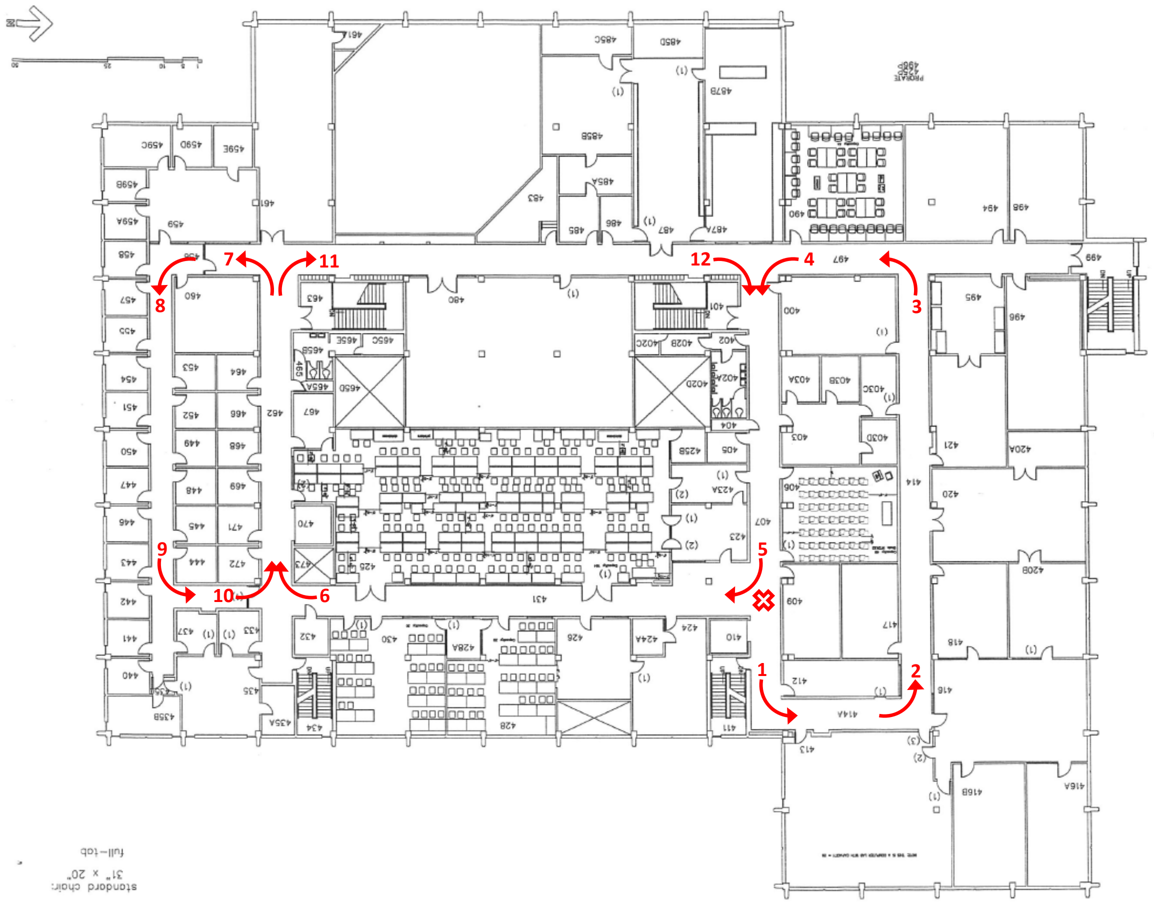

Figure 4.1: A Schematic Representation of the Clyde Dataset. The beginning and end of the trajectory correspond to the red 'X' down and to the right of the center of the image. The vehicle proceeds from the origin along straight paths between the turns around corners in the hallways. The turns are marked in the image by curved red arrows and are numbered in the sequence taken by the trajectory. The heading of the vehicle is always aligned with the forward direction of motion. Loop closures occur when reference poses are within 0.1 meters and have the same heading. All of the loop closures occur along the hallways between turns 4 and 5 and between turns 6 and 7.

implementation in Matlab for easy prototyping since our focus is only on the accuracy of the results.

4.3.2 A First Attempt at Extending BERT

As an initial attempt at extending BERT we apply several iterations of Algorithm 3 to the data. Let N + 1 be the total number of poses in the trajectory (numbered from 0 to N). The pose $\boldsymbol{\rho}_0$ is the certain global origin, and it is defined in the Clyde dataset

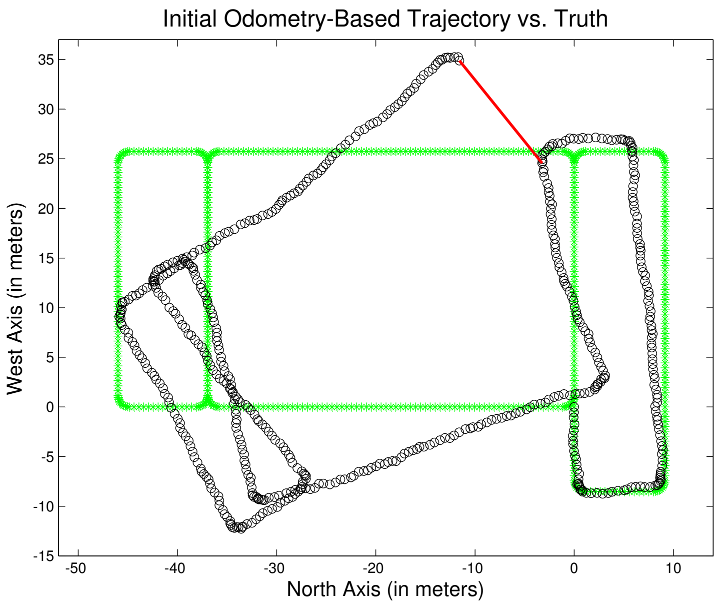

Figure 4.2: Clyde Dataset Trajectory: Odometry vs. Truth. The true positions of the reference poses are marked with green dots. Black circles mark the belief about reference poses based on initial odometry-like transformation estimates. The single loop closure used for this chapter's results occurs between the poses connected by a red line.

as $[n_0, w_0, \psi_0]^\top = [0, 0, -\frac{\pi}{2}]^\top$. Algorithm 3 is essentially the same as Algorithm 2, just spelled out in more detail. The only difference is summarized in Lines 6 - 8 where we take the additional steps needed to compute the joint covariance for the distribution $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j\right)$ containing the loop closure transformation. For reasons that will appear below, we designate Algorithm 3 as a naive approach to BERT.

Figure 4.2 shows an overhead view of the results from one iteration of BERT using Algorithm 3 and one iteration of LM compared to truth. The results from BERT are obviously worse than LM. BERT has closed the gap between the end of the trajectory and the other pose involved in the loop closure, but in the process the middle of the trajectory has been skewed to the east. Figure 4.3 further highlights this performance by plotting the Euclidean distance between the true global position and the position estimates for each pose.

However, we can also consider the performance of BERT and LM after one iteration by looking at the error in the relative transformation estimates. Figure 4.4 plots the Euclidean distance between the true and the estimated change in position for each transformation. We see the same behavior exhibited by LM (i.e. $\mathrm{g}^2\mathrm{o}$) in Chapter 3. Namely, LM increases the

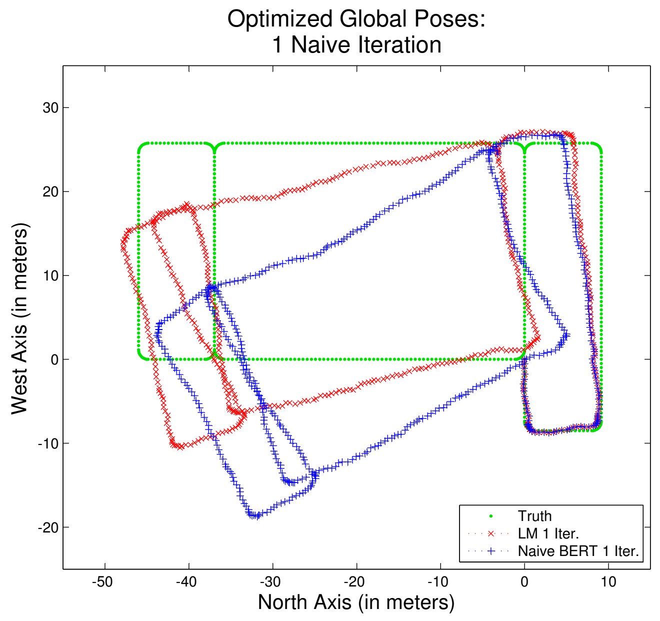

Figure 4.3: Overhead View of Optimized Poses After One Iteration. Information from the loop closure has been propagated through the network using Algorithm 3. Performance of BERT is obviously worse than a single iteration of LM, but the solution is plausible enough that one might hope for BERT to converge after additional iteration.

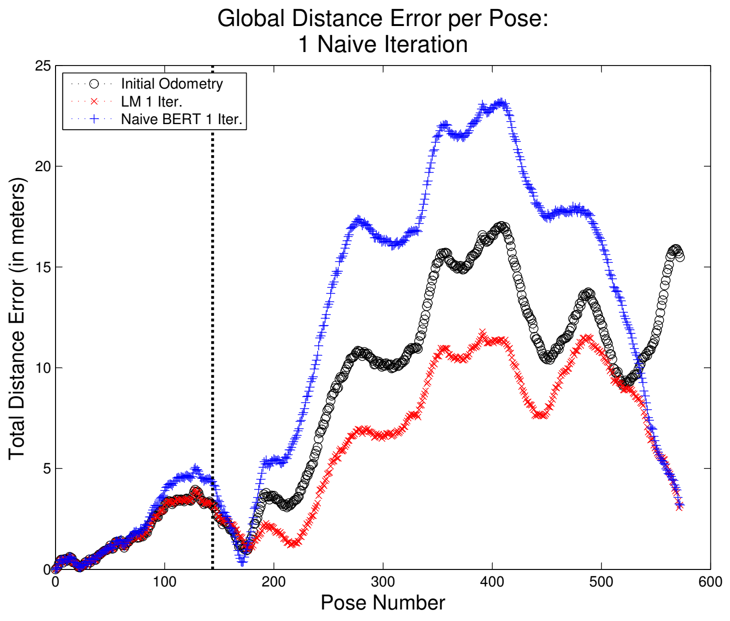

Figure 4.4: Global Pose Distance Error After One Iteration. This plot offers an alternative view of the results shown in Figure 4.3. The vertical axis of the plot represents the Euclidean distance between true and estimated poses. The vertical dotted line marks the originating pose in the loop closure. In this case, that is pose number 144. The only noticeable improvement BERT makes over the unoptimized result occurs toward the end of the trajectory.

Algorithm 3: One iteration of an initial naive attempt to extend BERT for a single loop closure between arbitrary poses.

```
   for $i = 0 : \mathrm{N} - 1$ do
1      Use $\boldsymbol{\rho}_i$ with $\boldsymbol{\tau}_i^{i+1}$ in Eq. (4.1) to find $\boldsymbol{\rho}_{i+1}$.
2      Calculate the Jacobians $\mathbf{J}_{\boldsymbol{\rho}_i}$ and $\mathbf{J}_{\boldsymbol{\tau}_i^{i+1}}$ according to Eq. (4.4) and Eq. (4.5).
3      Calculate the joint covariance for $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1}\right)$ as given in Eq. (4.8).
4      Save the values $a_i$ and $b_i$ as defined in Equations (4.6) - (4.7).
   end
5  Calculate the expected value of the loop closure transformation $\boldsymbol{\tau}_i^j$ using Eq. (4.12).
6  Calculate the cross-covariance between $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$ using Eq. (4.11)
7  Calculate the Jacobians $\mathbf{J}_{\mathrm{LC}_i}$ and $\mathbf{J}_{\mathrm{LC}_j}$ using Eq. (4.13) and Eq. (4.14).
8  Calculate the joint covariance for $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j\right)$ using Eq. (4.22).
9  Apply the loop closure measurement to $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j\right)$ in a Kalman update.
   for $i = \mathrm{N} - 1 : -1 : 0$ do
10     Extract the updated marginal distribution $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{i+1}\right)$
11     Use Equations (3.9) and (3.10) to find $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_i, \overset{\star}{\boldsymbol{\tau}}{}_i^{i+1}, \overset{\star}{\boldsymbol{\rho}}_{i+1}\right)$.
   end
```

error in the relative change in position, especially toward the end of the trajectory. BERT performs as should be expected in that the single loop closure causes almost no change in the initial estimates.

The increased error caused by LM in Figure 4.3 seems less pronounced here than in Chapter 3, Figures 3.7 and 3.8. This is due to the increased level of noise in the original front-end estimated transformations. In Chapter 3 the average initial error for the change in position was 2.1 cm for an average true change of 30 cm. In this current dataset, the average initial change in position error is 12.4 cm for a true change of 50 cm. That's a jump from about 7% to 25% of the true change.

Given that BERT delivers superior relative transformations after one iteration, one might hope that further iteration would allow BERT to improve its estimates of global poses. However, Figure 4.5 illustrates that this is clearly not the case. At each iteration the global pose estimates from BERT get considerably worse. The relative transformation estimates from BERT also degrade with each iteration as shown in Figure 4.6. Something is lacking in the algorithm.

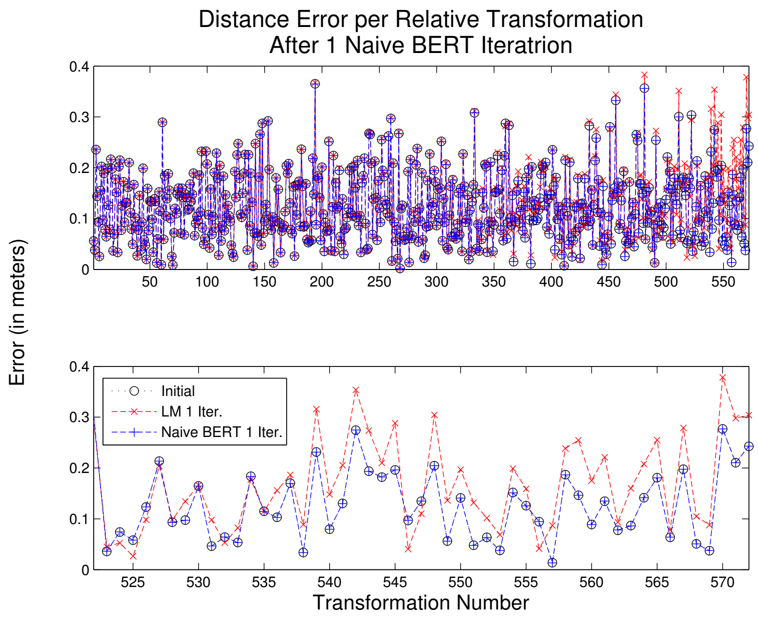

Figure 4.5: Relative Transformation Distance Error After One Iteration. The behavior here is similar to that witnessed in the initial tests of BERT in Chapter 3. The degradation caused by one iteration of LM is less pronounced this time due to the higher level of error already present in the initial estimates of the transformations. BERT again conforms to the expectation that a single loop closure ought to not change several hundred other loop closures very much individually. The top figure shows all the relative transformations in the test; the bottom shows a close-up of the end of the trajectory.

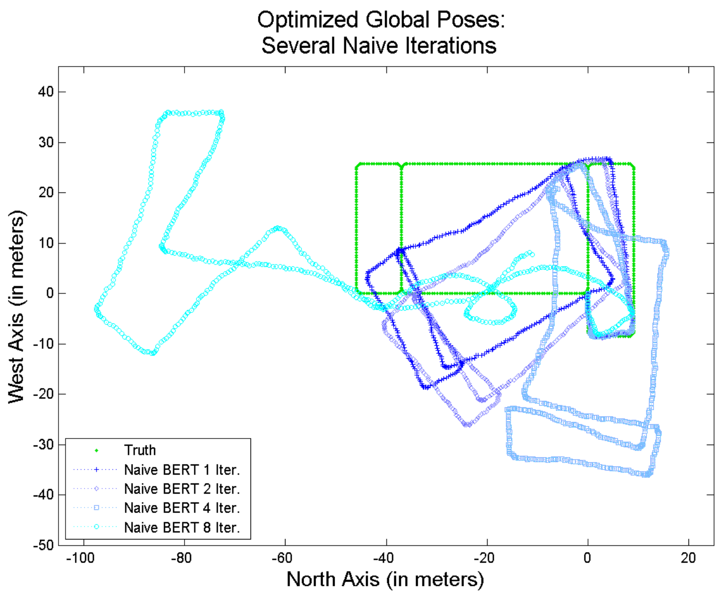

Figure 4.6: Overhead View of Optimized Poses After Several Iterations. The results from BERT using Algorithm 3 get worse at every iteration as the solution completely diverges from the truth.

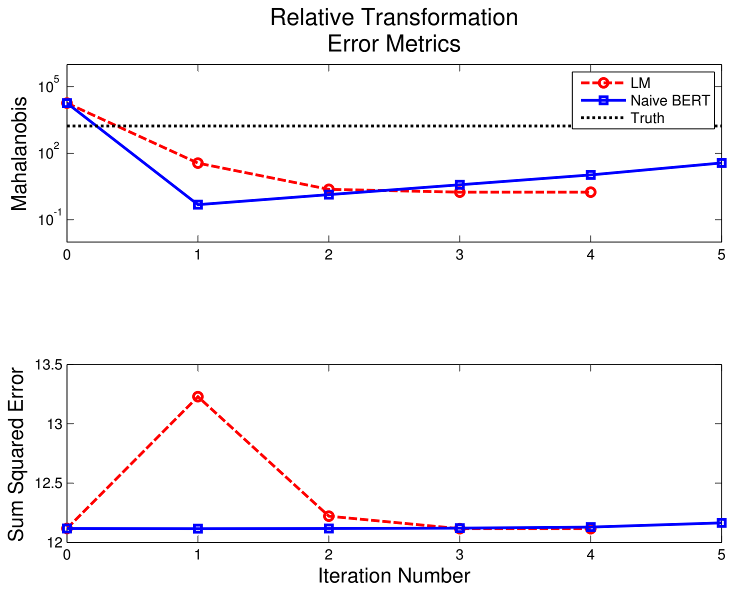

Figure 4.7: Relative Transformation Error Metrics For the Naive First Attempt to Extend BERT. The top plot shows the result of computing Equation (3.11) after each iteration of LM and BERT. Equation (3.11) is a measure of how well the optimized relative transformations agree with their original estimates. In the results presented here BERT agrees well with the original estimates after one iteration as it should for a single loop closure and several hundred odometry-like transformations. However, the metric grows quickly with each subsequent iteration. The bottom plot shows after every iteration the result of calculating the sum squared error defined in Equation (3.15). Again, iterating BERT degrades the accuracy of the relative transformation estimates. In both top and bottom plots, LM produces qualitatively the same results as in Chapter 3.

4.3.3 Linearizing Around the True State

If a vehicle performs SLAM using only onboard sensors then all of the position information available to the vehicle is relative; the vehicle receives no global information about position whatsoever. Even loop closure is a relative measurement, so a SLAM algorithm must balance the information in all the available relative position estimates in a way that produces an accurate global map. The key to make this work is to maintain the integrity of the correlations between elements of the map. With correct correlations an adjustment to one map element will correctly affect other map elements.

The front-end delivers relative transformation estimates that are inherently independent. In BERT we develop global pose estimates from those transformations to allow the transformations to remain independent. In the process of developing our prior belief about poses we introduce a strong correlation between consecutive global poses as well as a modest correlation between those poses and the transformation they share. Those correlations are a function of the Jacobians used to map uncertainty from one random vector to another. If loop closure information does not seem to be affecting the map correctly we should suspect the correlations, and therefore the Jacobians, to be part of the problem.

To test this conjecture we manipulate the free variables that make up the Jacobians $\mathbf{J}_{\boldsymbol{\rho}_i}$, $\mathbf{J}_{\boldsymbol{\tau}_i}$, $\mathbf{J}_{\mathrm{LC}_i}$, and $\mathbf{J}_{\mathrm{LC}_j}$. Using the results presented earlier in this Chapter, all of these Jacobians can be written in terms of global heading angle and relative change in position estimates. For the results in this subsection we replace those estimates with true values, but *only* when calculating Jacobians. Therefore, only the covariance matrices are directly changed by introducing these true values. We will relax the use of true values in linearization when we present results in Chapter 4.3.5, and we will discuss further options for improving linearization in Chapter 5.

Modifying the Jacobians with true values, we proceed to again iteratively apply Algorithm 3. Figure 4.7 shows an overhead view of the results. BERT eventually converges to a solution that resembles the truth, but BERT's final solution is skewed worse than both the first and the final solutions produced by LM. Figure 4.8 further illustrates this result with the Euclidean distance between true and estimated global positions.

It is interesting to note that in this scenario the very first iteration of BERT actually performs about as well as the converged LM solution. BERT in its first iteration has not completely closed the distance between the loop closure poses, but most of the global pose estimates represent a considerable improvement over the initial unoptimized estimate. If something could be done to prevent subsequent iterations from bending the solution to the west we might hope to arrive at a solution at least as good or maybe better than LM.

To that end we make an observation about the solution LM produces. If we look in Figures 4.7 and 4.8 at the LM solution we see that the first and last iterations of LM produce results that seem indistinguishable from each other *and* from the initial estimate for poses temporally preceding the originating pose of the loop closure. Zooming in for a closer look in Figure 4.9, and calculating the distance between the first and last iterations of LM confirms this observation. Multiple iterations of LM make no change in the global pose

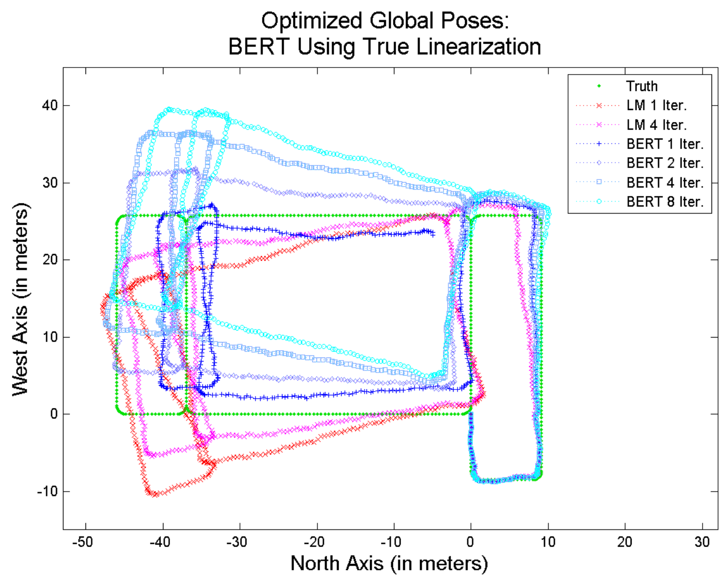

Figure 4.8: Overhead View of Optimized Poses Using True Linearization Points in Algorithm 3. Compared to the results in Figure 4.5, BERT now converges to a solution resembling truth, though less accurate than the LM solution.

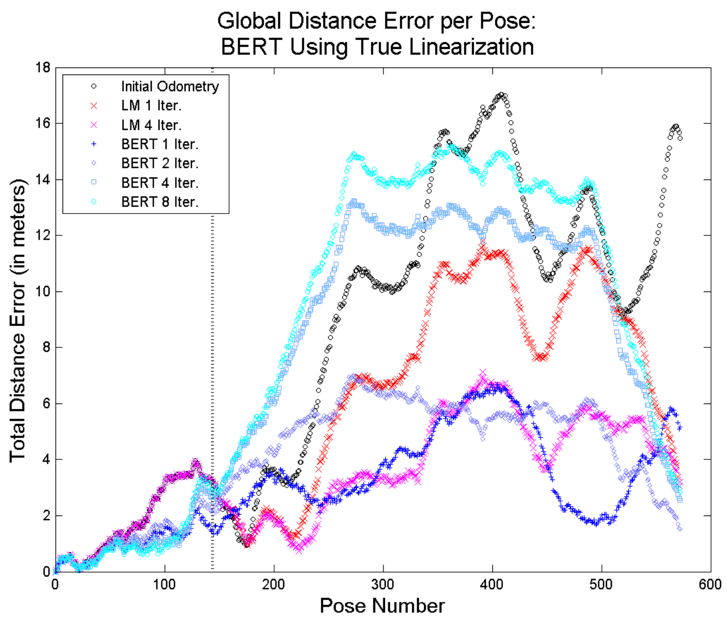

Figure 4.9: Global Pose Distance Error Using True Linearization Points in Algorithm 3. It is interesting to note here that the first iteration of BERT using true linearization points produces global pose estimates closer to the truth than the converged solution of LM for most of the poses. However, subsequent iterations of BERT generate errors greater than the unoptimized solution in many poses.

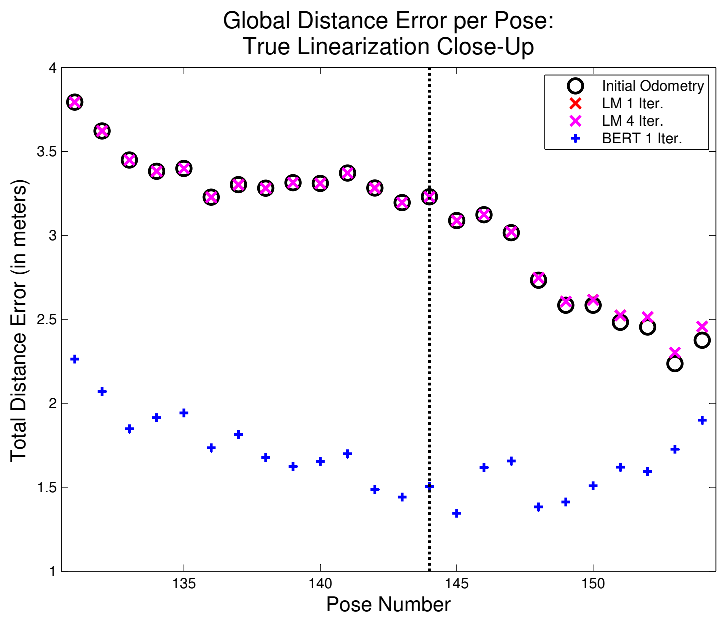

Figure 4.10: A Close-Up Look at Results From Figure 4.8. We see that before the originating pose in the loop closure that the global position estimates from LM are identical to the original estimates.

estimates before pose 144, and therefore LM also makes no change in the preceding relative transformation estimates.

The effect of LM on these variables leads us to another realization about the problem that we can associate with the result in Equation (4.18). Incorporating loop closure information should not affect global pose and relative transformation estimates that temporally precede the originating pose in the loop closure. Both the loop closure transformation and the odometry-like transformation that originate from pose 144 are independent of the pose as they are defined in its reference frame.

This insight can also be tied to the direct approach of optimizing relative transformations that we described in Chapter 3.1.3. A single covariance matrix for the joint distribution over all transformations would have no off diagonal elements relating the loop closure to the odometry-like transformations that lead up to pose 144. Therefore, applying the loop closure measurement to that single, large covariance matrix would have no effect on the earliest transformation estimates.

Algorithm 4: Modifying BERT for arbitrary poses in a single loop closure such that loop closure information only affects the transformations within the loop.

```
   for $i = 0 : \mathrm{N} - 1$ do
1      Use $\boldsymbol{\rho}_i$ with $\boldsymbol{\tau}_i^{i+1}$ in Eq. (4.1) to find $\boldsymbol{\rho}_{i+1}$.
2      Calculate the Jacobians $\mathbf{J}_{\boldsymbol{\rho}_i}$ and $\mathbf{J}_{\boldsymbol{\tau}_i^{i+1}}$ according to Eq. (4.4) and Eq. (4.5).
3      Calculate the joint covariance for $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1}\right)$ as given in Eq. (4.8).
4      Save the values $a_i$ and $b_i$ as defined in Equations (4.6) - (4.7).
   end
5  Calculate the expected value of the loop closure transformation $\boldsymbol{\tau}_i^j$ using Eq. (4.12).
6  Calculate the cross-covariance between $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$ using Eq. (4.11)
7  Calculate the Jacobians $\mathbf{J}_{\mathrm{LC}_i}$ and $\mathbf{J}_{\mathrm{LC}_j}$ using Eq. (4.13) and Eq. (4.14).
8  Calculate the joint covariance for $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j\right)$ using Eq. (4.22).
9  Apply the loop closure measurement to $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^j, \boldsymbol{\rho}_j\right)$ in a Kalman update.
   for $i = \mathrm{N} - 1 : -1 : \mathrm{N}_{\mathrm{LC}_i}$ do
10     Extract the updated marginal distribution $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{i+1}\right)$
11     Use Equations (3.9) and (3.10) to find $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_i, \overset{\star}{\boldsymbol{\tau}}{}_i^{i+1}, \overset{\star}{\boldsymbol{\rho}}_{i+1}\right)$.
   end
```

4.3.4 Optimizing Only the Subset Involved in Loop Closure

The realization presented at the end of the preceding subsection leads us to modify Algorithm 3. Let the global poses be numbered starting from the global origin $\boldsymbol{\rho}_0$, and let $\mathrm{N}_{\mathrm{LC}_i}$ identify the pose from which the loop closure transformation emanates. In the case of our current example, $\mathrm{N}_{\mathrm{LC}_i} = 144$. We will let $\mathrm{N}_{\mathrm{LC}_j} = 572$ designate the pose to which the loop closure transformation extends. We recall that in this experiment $\mathrm{N}_{\mathrm{LC}_j} = \mathrm{N}$, the last pose of the trajectory. Algorithm 4 presents our modified approach wherein we only propagate loop closure information back to the originating pose in the loop closure. Therefore, we update only those relative transformations that are part of the loop. For most of the results in this subsection we continue to use true state values in our Jacobians.

Figures 4.10 and 4.11 illustrate global pose results produced by BERT using Algorithm 4. We see now that BERT is behaving more like LM. In fact, the second iteration of BERT outperforms the fourth and final iteration of LM as can be most clearly seen in Figure 4.11. On a less positive note, we observe that the first iteration of BERT has a sharp discontinuity between $\boldsymbol{\rho}_{144}$ and $\boldsymbol{\rho}_{145}$, and there is significant distance between the loop closure poses $\boldsymbol{\rho}_{144}$ and $\boldsymbol{\rho}_{572}$. However, the optimized estimates for both poses are closer to their true values than were their initial estimates.

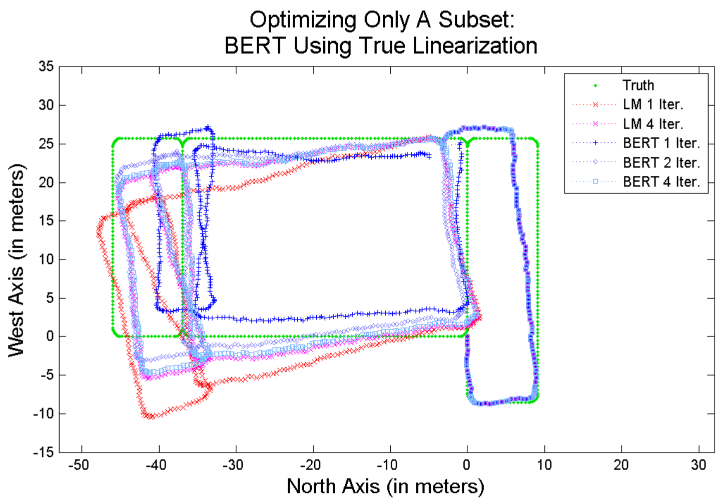

Figure 4.11: Overhead View of Optimized Poses Using True Linearization Points in Algorithm 4. The final BERT solution essentially matches the LM solution, but the first two iterations of BERT are arguably better than the final solution.

We turn to Figure 4.12 to illustrate the relative transformation results from LM and BERT using Algorithm 4. We will no longer report plots like Figure 4.4 showing the error in the relative change of position for each transformation. The illustration of the relative transformation error shown in Figure 4.12 provides a more visible measure of performance. We will also restrict the scale on the vertical access for the sum squared error to exclude the initial response of LM and give a more detailed look at the performance of BERT.

As before, BERT avoids the undesirable transient response in sum squared error that LM suffers from. The steady rise of BERT's sum squared error has also been noticeably muted compared to Figure 4.6. The first two iterations of BERT improve the sum-squared error in the relative transformations. However, beginning at the third iteration BERT allows a small rise in this error metric. BERT also settles into a final error value that is just slightly

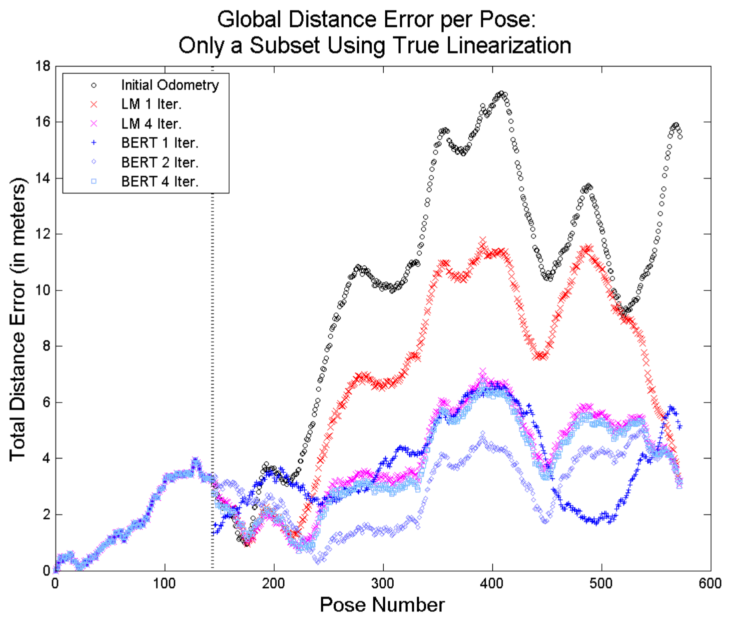

Figure 4.12: Global Pose Distance Error Using True Linearization Points in Algorithm 4. BERT acheives the most accurate solution at its second iteration.

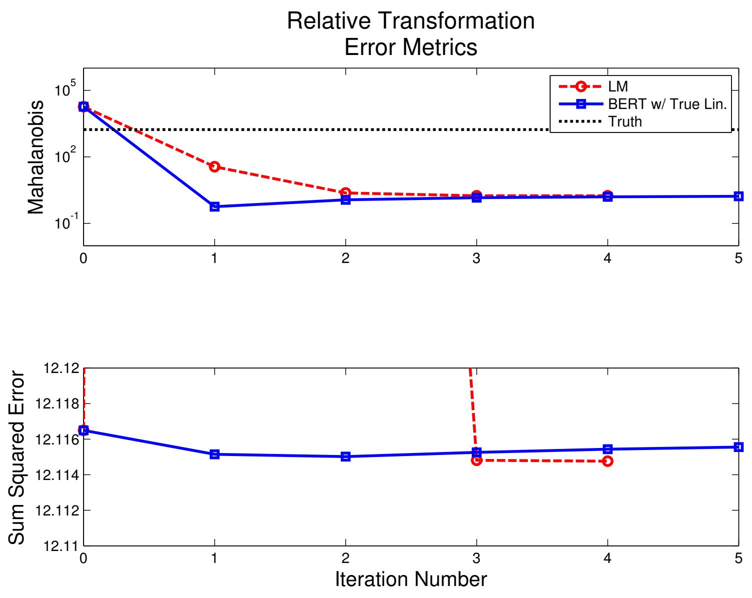

Figure 4.13: Relative Transformation Error Metrics Using True Linearization Points in Algorithm 4. BERT still reduces the sum squared error of the relative transformations, but it acheives its minimum at the second iteration. Exploring the topic of when to terminate iterations is a subject of future work discussed in Chapter 5

greater than that of LM, although the final value of BERT's sum-squared error is still an improvement over the unoptimized value.

We will need to keep an eye on the behavior of sum-squared error in the the relative transformations as we continue to extend BERT to more and more complicated networks. We expect in the long run that BERT will outperform LM on this count, but the fact that BERT first decreased the SSE before allowing it to rise again is a minor cause for concern. Further experimentation is needed because the results developed to date present too little change in the relative transformation estimates to allow for a stronger opinion of what to expect.

Finally, one might hope that we could relax the use of true states in the Jacobians by using Algorithm 4. However, Figure 4.13 shows that only optimizing relative transformations in the loop does not address the problem introduced by bad linearization. The results in Figure 4.13, generated with Algorithm 4 and using the current state estimates in the Jacobians, still diverge to an unacceptable solution. In the next subsection we take a first look at how to correctly calculate Jacobians.

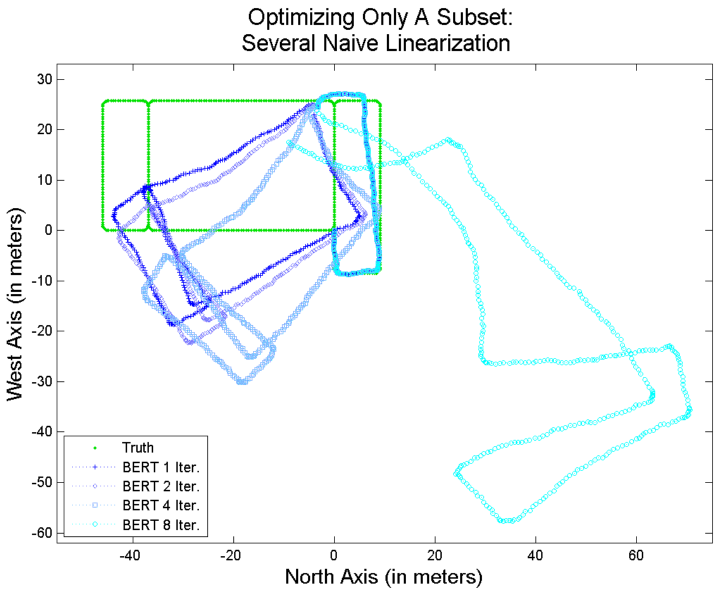

Figure 4.14: Overhead View of Optimized Poses Using Naive Linearization Points in Algorithm 4. Optimizing only those transformations that make up the closed loop does not avoid the problems introduced by bad linearization points in the Jacobians.

4.3.5 Using First Estimates in the Jacobians

Maintaining correct correlations between map elements is directly tied to the issue of an estimator's statistical consistency. We mentioned the topic of consistency in Chapter 1.2.1 in connection with EKF-SLAM. An estimator can become inconsistent when the filter estimated covariance becomes too small. This gives rise to incorrect correlation between map elements and leads to a divergent solution. Consistency has been a topic of much investigation in the EKF-SLAM community. We have already drawn analogies between BERT and the EKF-SLAM submapping technique of [27]. We look now to work in the EKF-SLAM community for solutions to our problem with correct linearization.

The authors of [42] were the first to draw significant attention to the issue of consistency in EKF-SLAM. They presented an experiment that they describe as a counter-example to the validity of the EKF-SLAM approach. In their experiment a stationary robot repeatedly measures the range and bearing of a single feature in a 2D environment. Because both robot and feature are known to be stationary, the feature's position relative to the robot is known to be constant. Yet the authors showed that the vehicle's estimate of its own global pose became increasingly confident after each measurement. This should not happen because repeating the measurement between the stationary objects should not introduce more information about the vehicle's global pose. The EKF-SLAM estimator was becoming overconfident, i.e. inconsistent.

EKF-SLAM consistency has since been a research question pursued by other authors. In [43] the authors perform simulations of a robot moving in 2D and estimating point features for its map. Through Monte Carlo simulation they identify accurate heading estimates as essential to maintaining the correct covariance properties that define the estimator's consistency. They suggest that a vehicle use a global heading measurement, such as from a magnetometer, to constrain the error in heading and eventually freeze the map to avoid incorporating faulty information.

The line of work that we have already cited beginning with [27] also gives cursory treatment of consistency. They assert, based on their experience with simulation and implementation, that using submaps damps the onset of inconsistency. They suggest that this happens because submaps have their own local reference frame. When they instantiate a new submap, they know with certainty that the vehicle is at the origin of that submap by definition. The decreased error and uncertainty in the states used to calculate Jacobians then leads to a more consistent map when the local maps are merged into a global one.

We believe the best, most principled treatment of statistical consistency is presented by the authors of [45]. We will discuss their most recent work in Chapter 5, but in [45] they propose a simple solution to the consistency problem. Instead of using the latest state estimates in the Jacobians, they use the first-ever estimates for those states. The authors call their SLAM estimator a First-Estimates Jacobian (FEJ) EKF.

Not using the latest, and presumably the best, state estimates in the Jacobians seems counterintuitive. However, the authors show analytically in [45] that linearizing about the latest state estimates causes the linearized error-state model to have an observable subspace of greater dimension than the underlying nonlinear system. In short, using the latest estimates in the Jacobian leads the filter to believe it has more information than it does. This leads to the overconfidence and incorrect correlation properties noted by other authors.

Before proceeding to some results, we make a note here about linearization and the traditional back-end optimization approach implemented with LM. In BERT and LM the Jacobians for each algorithm are a function of the relative transformation estimates. LM never changes those values in Equation (3.11), the Mahalanobis distance metric used as the LM objective function. Therefore LM always linearizes around the original estimates of the transformations. We believe this is why LM does not suffer from the same linearization problems as BERT though we haven't verified this with rigorous analysis.

In this subsection we take an FEJ approach to BERT to test whether it may also suffer from a problem analagous to that of feature-based EKF-SLAM. We again apply Algorithm 4 to the data, but in Lines 2 and 7 we will use the front-end estimated values needed to compute the Jacobians. The results are presented in Figures 4.14 - 4.16.

The overhead view in Figure 4.14 and the plot of the position error in Figure 4.15 show that BERT using FEJ gives results more similar to LM than the results presented previously. The first iteration of BERT using FEJ does not give as good improvement in global position estimates as that given by using true values in the Jacobians, but the final solution is nearly the same. We also still observe a large discontinuity between global poses

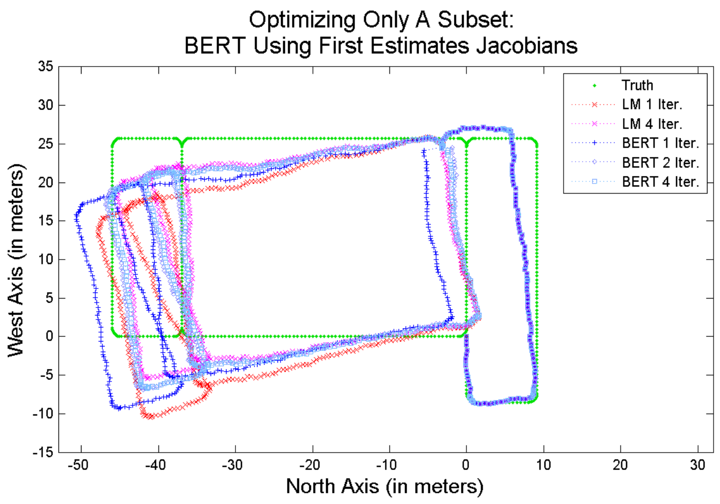

Figure 4.15: Overhead View of Optimized Poses Using FEJ Linearization Points in Algorithm 4. Notice that despite using unoptimized values to compute Jacobians BERT arrives at a solution essentially the same as LM. This suggests that an observability analysis similar to that of [45] might reveal even better linearization points and better results. However, an FEJ approach provides a reasonable result without any additional computation.

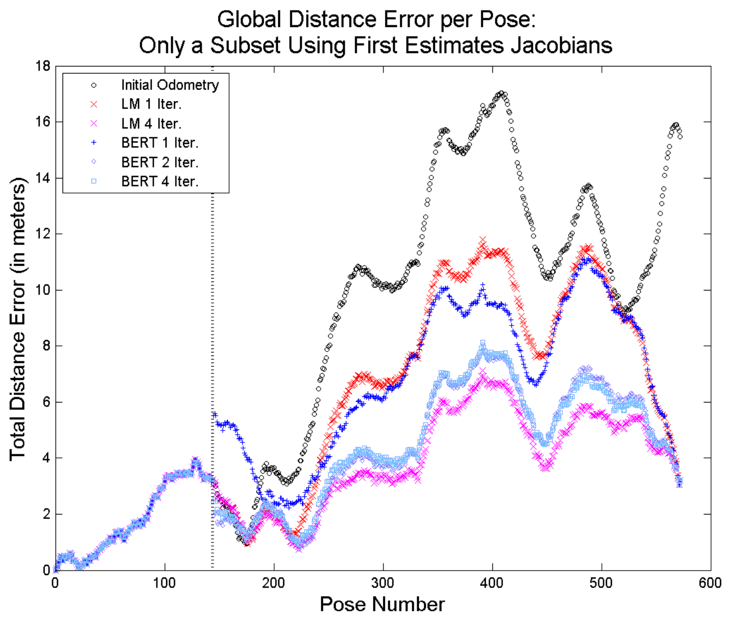

Figure 4.16: Global Pose Distance Error Using FEJ Linearization Points in Algorithm 4. Similar to Figure 3.10, minor differences between LM and BERT are likely due to slightly different linearization points. Using an observability analysis to arrive at better linearization points might make BERT consistently more accurate than LM (e.g. consider Figure 4.11 results using true values in linearization).

$\boldsymbol{\rho}_{144}$ and $\boldsymbol{\rho}_{145}$, though this time the discontinuity noticeably increases the position error in the several poses immediately after $\boldsymbol{\rho}_{144}$.

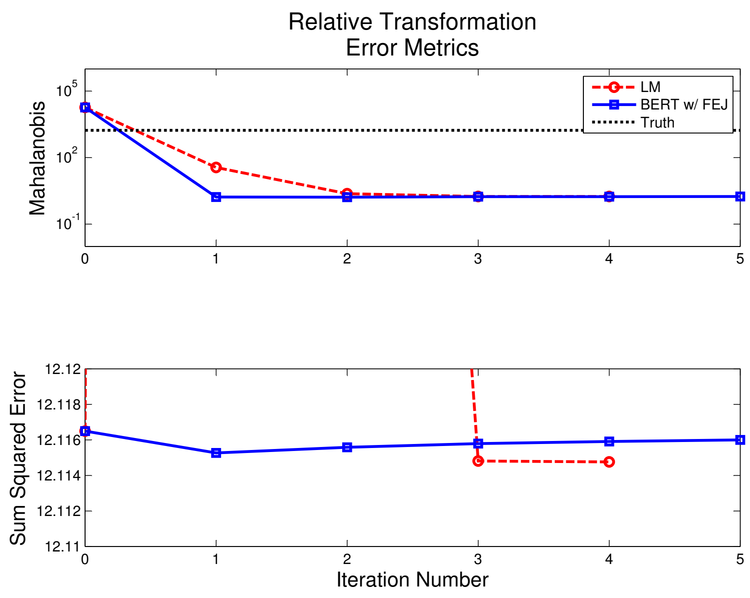

Figure 4.17: Relative Transformation Error Metrics Using FEJ Linearization Points in Algorithm 4. Performance is similar to that shown in Figure 4.12 where true state values were used in linearization.

Finally, we note in Figure 4.16 that the error in relative transformation estimates is similar to that found by using the true linearization points. Using FEJ still avoids the negative transient response seen in LM. However, the steady state value of the error is slightly higher using FEJ compared to using true values for linearization.

4.3.6 Coming Full Circle: BERT Using a Relative Global Origin

To conclude this Chapter we present one final variation of extending BERT. We can take a different lesson away from the realization presented in Section 4.3.4. Since the loop closure information doesn't change the transformations and poses preceding $\boldsymbol{\rho}_{\mathrm{LC}_i}$, why not treat an arbitrary loop closure the same way we handled a loop closure in Chapter 3. We can simply identify which transformations belong to the loop, set $\boldsymbol{\rho}_{\mathrm{N}_{\mathrm{LC}_i}} = [0, 0, 0]^\top$, and generate our prior belief about subsequent poses in the reference frame of $\boldsymbol{\rho}_{\mathrm{N}_{\mathrm{LC}_i}}$. We can recover a global map of poses after any iteration by shifting and rotating the optimized poses by our odometry-based estimate of $\boldsymbol{\rho}_{\mathrm{N}_{\mathrm{LC}i}}$. Alternatively, we can recover a global map after the optimization of transformations in the loop closure has converged by composing the odometry-like transformations beginning from the global origin $\boldsymbol{\rho}_0$. We summarize this approach in Algorithm 5 and present the results below.

Algorithm 5: Applying BERT for arbitrary poses in a single loop closure such that loop closure information only affects the transformations within the loop.

```
   for $i = 0 : \mathrm{N}_{\mathrm{LC}_i} - 1$ do
1      Use $\boldsymbol{\rho}_i$ with $\boldsymbol{\tau}_i^{i+1}$ in Eq. (4.1) to find $\boldsymbol{\rho}_{i+1}$.
2      Calculate the Jacobians $\mathbf{J}_{\boldsymbol{\rho}_i}$ and $\mathbf{J}_{\boldsymbol{\tau}_i^{i+1}}$ according to Eq. (4.4) and Eq. (4.5).
3      Calculate the joint covariance for $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1}\right)$ as given in Eq. (4.8).
   end
4  Set a second, separate value for $\boldsymbol{\rho}_{\mathrm{LC}_i}$ that is $[0, 0, 0]^\top$ with a covariance matrix of zeros.
   Use this value to compose the prior belief about the remaining poses in the following.
   for $i = \mathrm{N}_{\mathrm{LC}_i} : \mathrm{N} - 1$ do
5      Use $\boldsymbol{\rho}_i$ with $\boldsymbol{\tau}_i^{i+1}$ in Eq. (4.1) to find $\boldsymbol{\rho}_{i+1}$.
6      Calculate the Jacobians $\mathbf{J}_{\boldsymbol{\rho}_i}$ and $\mathbf{J}_{\boldsymbol{\tau}_i^{i+1}}$ according to Eq. (4.4) and Eq. (4.5).
7      Calculate the joint covariance for $\mathrm{p}\left(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1}\right)$ as given in Eq. (4.8).
   end
8  Apply the loop closure measurement, equivalent to an estimate of $\boldsymbol{\rho}_{\mathrm{N}}$, in a Kalman
   update of $\mathrm{p}\left(\boldsymbol{\rho}_{\mathrm{N}-1}, \boldsymbol{\tau}_{\mathrm{N}-1}^{\mathrm{N}}, \boldsymbol{\rho}_{\mathrm{N}}\right)$.
   for $i = \mathrm{N} - 1 : -1 : \mathrm{N}_{\mathrm{LC}_i}$ do
9      Extract the updated marginal distribution $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_{i+1}\right)$
10     Use Equations (3.9) and (3.10) to find $\mathrm{p}\left(\overset{\star}{\boldsymbol{\rho}}_i, \overset{\star}{\boldsymbol{\tau}}{}_i^{i+1}, \overset{\star}{\boldsymbol{\rho}}_{i+1}\right)$.
   end
   if Global pose map is desired at the end of an iteration then
11     Rotate each optimized pose from the reference frame of $\boldsymbol{\rho}_{\mathrm{LC}_i}$ to the global
       reference frame using the odometry-estimated value of the heading of $\boldsymbol{\rho}_{\mathrm{LC}_i}$.
12     Add the odometry-estimated, globally referenced value of $\boldsymbol{\rho}_{\mathrm{LC}_i}$ to the rotated,
       optimized poses.
   end
```

Figures 4.17 and 4.18 show the global pose results from applying Algorithm 5 and recovering the globally referenced estimates of the optimized poses at each iteration. The

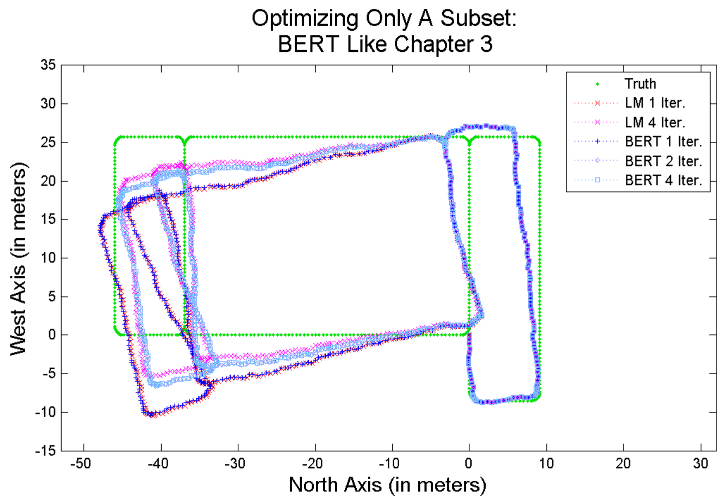

Figure 4.18: Overhead View of Optimized Poses Using Algorithm 5. Results for global pose estimates are almost identical to those given by LM.

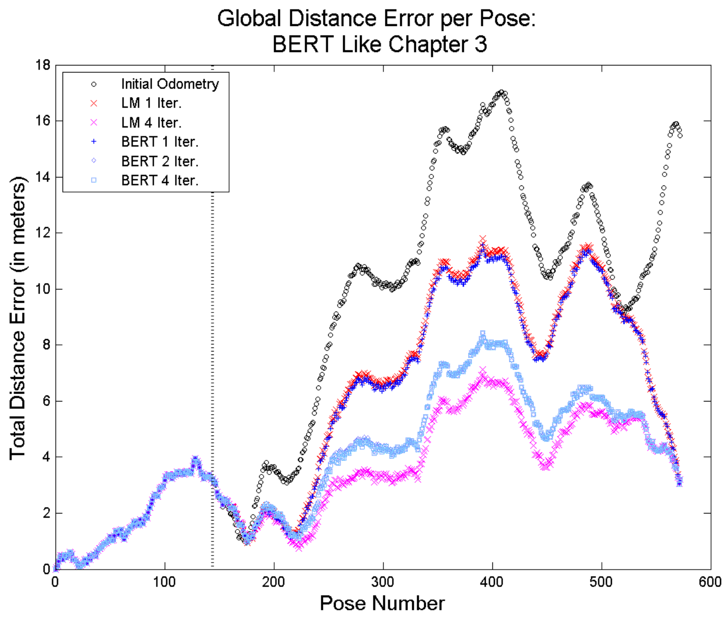

Figure 4.19: Global Pose Distance Error Using Algorithm 5. Results for global pose estimates are almost identical to those given by LM.

results are practically identical to LM. We feel confident that the small differences seen in this example, like the differences seen between BERT and g^{2}o in Chapter 3.2.3, are a minor effect of slightly different linearization points. It is also interesting to note that the global pose map recovered by Algorithm 5 has no discontinuity following $\boldsymbol{\rho}_{\mathrm{LC}_i}$.

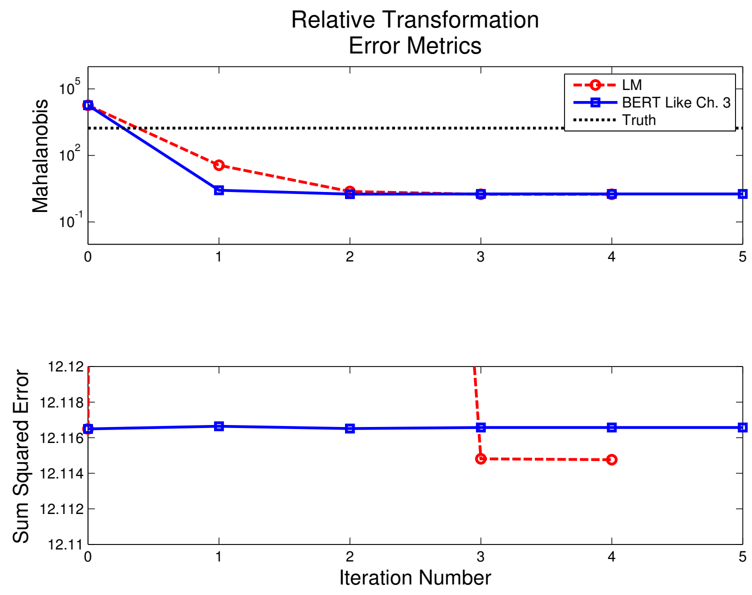

Figure 4.20: Relative Transformation Error Metrics Using Algorithm 5. These results are peculiar. The relative transformations that are not included in the closed loop are not changed by optimization, and the transformations in the loop are treated just as in Chapter 3. However, instead of small and monotonically better estimates of the transformations, we observe a very slight uptick in the sum squared error after the first iteration. Future work will need to evaluate the reason for this behavior.

Figure 4.19 shows that BERT again gives almost no change in the relative transformations as should be expected in this single loop closure scenario. However, BERT actually leads to an almost imperceptible increase in the sum squared error metric at the first iteration. As stated earlier, the change induced by a single loop closure is too small to draw any confident comparisons between the various ideas presented in this chapter for extending BERT. Further work will be required to analyze and differentiate the performance of variations on BERT with respect to the sum-squared error.

Taking the approach in Algorithm 5 inherits some nice traits from BERT presented in Chapter 3. We do not need to do anything special to develop a loop closure hypothesis. Since all of the poses are defined in the reference frame of $\boldsymbol{\rho}_{\mathrm{LC}_i}$, the pose $\boldsymbol{\rho}_{\mathrm{LC}_j}$ at the end of the trajectory will itself be the loop closure transformation. Also, in contrast to the several preceding subsections, we do not need to take any special measures to achieve proper correlations. However, Algorithm 5 will not be a panacea and may actually have limited applicability. We discuss these issues further in our Chapter 5.

4.4 Conclusions From the Arbitrary Single Loop Results

In this chapter we have presented some important principles for extending BERT to arbitrary loop closures and map topologies. We have identified at least two major results. First, we showed in Chapter 4.2.1 that almost no additional computation is required to calculate a prior belief about an arbitrary loop closure between two globally referenced poses. This contrasts with the EKF-SLAM submapping approach in [27]. During and after a loop closure, the algorithm in [27] requires additional states be added to their submaps that are analogous to our modular joint distributions $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1})$.

We derived another important result in Chapter 4.2.2 while examining the structure of the covariance matrix for the loop closure hypothesis $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{j}, \boldsymbol{\rho}_j)$. We found in our analysis that the loop closure transformation is independent of the pose from which it originates. Not only does this simplify and lend intuition to the covariance of $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{j}, \boldsymbol{\rho}_j)$, it also motivates the approach for extending BERT presented in Algorithm 5 of Chapter 4.3.6.

We have also identified an important problem when applying a loop closure to a network composed of globally referenced poses. Choosing appropriate values to calculate the Jacobians is critical to the performance of the algorithm. Motivated by another analogy to EKF-SLAM, we have presented an initial solution to this problem by using the front-end estimated values needed to compute the Jacobians.

Propagating loop closure information for a single loop closure should readily generalize to an arbitrary arrangement of multiple loop closures. The origin pose $\boldsymbol{\rho}_i$ in each loop closure hypothesis is adjusted only when updated information reaches it through its associated odometry-like transformation in the modular joint distribution $\mathrm{p}(\boldsymbol{\rho}_i, \boldsymbol{\tau}_i^{i+1}, \boldsymbol{\rho}_{i+1})$. We can use this fact to devise schemes for applying information from several loop closures; we discuss this further in Chapter 5.

Chapter 5

Conclusion

In this dissertation we have presented techniques and results that further the development of an autonomous air vehicle capable of exploring unknown, indoor environments. A small indoor flight vehicle requires quick and accurate feedback using a truly 3D exteroceptive sensor, but it must process that sensor data and produce timely navigation information using limited computational resources. Our work contributes to addressing this engineering tradeoff.

To conclude this dissertation we sketch out several avenues for future work, after which we provide some summary remarks.

5.1 Future Work

Much work remains in order to implement the system conceived in Chapter 1 and illustrated by the block diagram in Figure 5.0. Foe example, another student in our lab is currently pursuing the challenging task of implementing and refining the time-critical tasks represented by the blocks labeled "Visual Odometry," "Front-End Estimator," "Low-Level Planner," and "Position Controller." For the block labeled "Loop Closure" there are several existing techniques, but the best techniques currently rely on cameras that are limited to appropriately illuminated environments. The block labeled "High-Level Planner" also remains to be developed to incorporate the information gained from the background tasks into the vehicle's real-time decision making.

Our future work centers on extending our approach to back-end map optimization. As a completely new paradigm, we see considerable opportunity to further develop BERT. There are several interesting issues that could be considered. We will organize these as separate topics, but it should be apparent that they are closely related and often overlap.

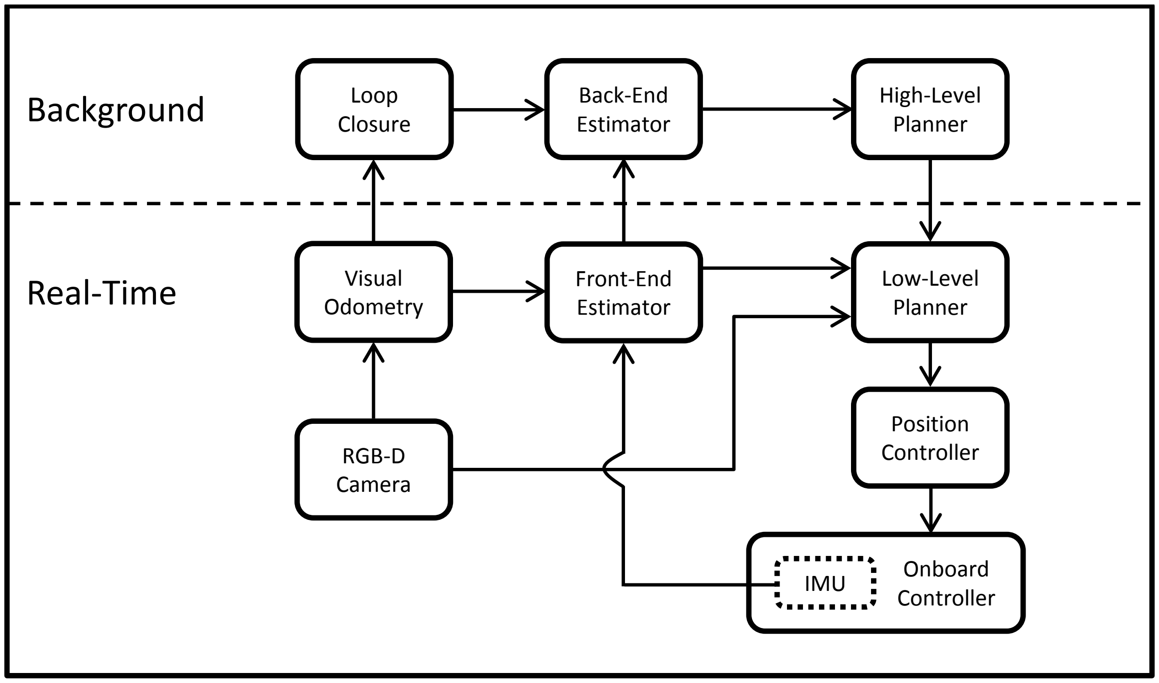

Figure 5.1: Our proposed navigation scheme. The system is based on graph-based SLAM. The time-critical navigation in the front-end is performed relative to reference frames associated with saved images of the environment. Refining the map happens when computational resources permit, and it only feeds directly into the vehicle's high-level decision making.

We expect all of the following areas of future work will contribute to the final goal of making BERT an effective tool for arbitrary problems in back-end optimization.

5.1.1 Observability-Constrained Linearization for Globally Referenced Poses

We begin with the ideas most relevant to the end of Chapter 4. We showed that the linearization points chosen for Jacobians are critical to obtain a good final solution from the optimization. We demonstrated this in Chapter 4.3.4 by contrasting results between the case where linearization occurs using true states and the case where linearization is performed using the most recent state estimates. We can reach essentially the same final solution as linearization about true states by instead using the first-ever estimated values in the Jacobians (dubbed FEJ).

However, we may find a performance improvement over FEJ by considering more sophisticated means of choosing linearization points. Note in Figures 4.10 and 4.11 that the global pose estimates in the first iteration of BERT using truth in linearization certainly outperforms the results for the first iteration of FEJ as shown in Figures 4.14 and 4.15. We have repeatedly expressed interest in the intermediate results of optimization because computational limitations (as well as other possible reasons we discuss below) may argue against running the optimization all the way to convergence for each new loop closure.

Our selection of first-ever estimates for computing Jacobians was motivated by a suspected connection between BERT and the statistical consistency research done for EKF-SLAM. We cited the authors of [45] for proposing the FEJ idea based on their observability analysis of an EKF-SLAM estimator. They showed in the EKF-SLAM case that using first-ever estimates in the Jacobians kept the observability properties of the linearized system in harmony with the observability of the true non-linear system. In subsequent work [46] the same authors propose additional methods for choosing linearization points that maintain the correct observability properties while improving the accuracy of state estimates. We propose that future work consider how a similar tack might be taken for BERT while seeking to maintain the efficiency that makes BERT attractive for our application.

We also make a few additional notes here in connection with this proposed area of future work. One might look at the results in Chapter 4.3.6 and conclude that we need not consider choosing good linearization points. Taking the approach in Chapter 4.3.6 does not require any special care when choosing linearization points and produces reasonable results. The Chapter 4.3.6 approach benefits from the fact that all pose states involved in a single loop closure are expressed in the same reference frame as the loop closure. It is therefore impossible to experience the problem identified in [45], i.e. the loop closure cannot introduce nonexistent information about the poses as expressed in the global frame. However, there are certainly scenarios where the Chapter 4.3.6 approach is not applicable, e.g. if we want to optimize the map using several loop closure transformation estimates simultaneously.

5.1.2 Further Analysis Related to Joint Covariance

One of the interesting differences between BERT and existing algorithms is that BERT operates on the covariance form of the joint Gaussian. As such, it is natural to consider concepts like correlation and consistency in the context of the problem. However, all we have done to this point is to make qualitative statements about the importance of correlations and the likely affect they have on the solution's consistency.

We can think of several questions that ought to be given a more quantitative or analytical answer. What would be the outcome of Monte Carlo simulations of BERT to assess its statistical consistency using the average NEES metric defined in [44] and applied in [43]? How do correlations between global poses evolve as you compose a prior belief about the trajectory? Do temporally distant but spatially proximate poses become more correlated as multiple loop closures are applied in the optimization? What happens to the correlation between global pose estimates when we marginalize their posterior estimates and compose a new prior belief from the optimized odometry-like transformations? What happens to the correlations between relative transformations and the poses they connect as we iterate on a single loop closure estimate or as multiple loop closure estimates are applied in the optimization? And, how might answers to these questions inform our further development of the algorithm?

We consider a solid analysis of questions related to covariance and correlation to be the key to further extending BERT to general back-end optimization problems.

5.1.3 Map Management

Map management is closely related to questions of correlation and joint distributions. In the SLAM literature, "map management" refers to the process used to limit the growth of the map over time by choosing map elements to remove. We have proposed in the present presentation of BERT to keep every reference image in a temporally unbroken sequence from the global origin to the current pose. Such a proposal may not always be desirable. If two images of the same location are sufficiently similar, memory and computation resources *might* incline us to remove one image and amalgamate the relative transformations associated with it into a new trajectory.

Pruning reference poses and combining relative transformations to save computer resources may sound good on the surface, but it may have counter-intuitive or undesirable consequences. For example, the authors of [55] show that estimating the whole trajectory of a vehicle allows their approach to back-end optimization to run faster than if they were to only keep the most recent pose (where, in either case, the filter also estimates the position of point features in the environment). For BERT as currently conceived, we use the entire sequence of independently estimated odometry-like transformations to compose our prior belief about global poses before applying estimates of the loop closure transformations. It is unclear at this time what consequences may arise from manipulating the map to remove some reference poses.

In connection to map management, we also reiterate here a principle that we consider to be fundamental to autonomous indoor flight. Any effect or action related to map management should be kept out of the real-time control of the vehicle. The current reference image and the globally topological map available in the front-end are sufficient for the immediate navigational needs of the air vehicle. Nonessential but helpful changes to the map should only be propagated to the vehicle's behavior through the mediation of the high and low level planners (see Figure 1.6).

5.1.4 The Common Mass-Spring Analogy

Since the early papers on graph-based SLAM, authors have referred to an analogy between the graphical network being optimized and a mass-spring system being relaxed to its lowest energy state. The authors of [51] were among the first to draw this comparison, and to our knowledge they are the only ones to have used the analogy to directly inform the development of their algorithm. Certainly, none of the recent leading papers in back-end optimization do more than pay lip service to the analogy. Globally referenced poses are supposed to represent masses, and the relative transformations connecting the poses are supposed to be like springs. The initially estimated mean and covariance of a transformation are supposed to represent the spring's resting length and spring constant, respectively.

We suspect the analogy, as currently conceived, is flawed. Even earlier work in EKF-SLAM (summarized in [35]) also draws an analogy between a network of springs and the map of the environment. The authors identify the *correlation between map elements* as the analog to a spring constant and assert that observations of the map elements serve to increase those correlations, i.e. tighten the springs.

We consider it more intuitively correct to identify correlation between map elements with the spring constants. In the limit as two global pose, for example, become almost completely correlated the relative transformation between them becomes almost completely sure. Our development of BERT has already benefited more than once by drawing on the EKF-SLAM literature, and we think we can improve the graph-based SLAM analogy to masses and springs by again referring to the prior EKF-SLAM work.

We also have an ephemeral idea for a new physical analogy to graph-based SLAM. We picture iterations of BERT for a single loop closure to be like tightening a single screw while constructing a complicated apparatus. We refer to Figure 4.1 and Figures 4.7 and 4.8 to motivate and illustrate this discussion.

Upon declaring a loop closure we have decided that two global poses should be connected (see the red line in Figure 4.1). After one iteration of BERT using Algorithm 3 we have screwed down the connection between the loop closure poses but they are not yet tightly connected (see Figures 4.7 and 4.8 to observe the discontinuity between the end of the trajectory and the other pose in the loop closure). We can continue to tighten the connection by iterating on the process, but we can see in Figure 4.7 that this causes the map (apparatus) to be skewed away from the desired final configuration.

We expect to make more connections between parts of the map. When one screws together several connected pieces of a complicated physical object, it is often best to only screw down each screw to be a little less than "finger tight" until a sufficient number of screws are in place to assure the parts align properly. We wonder if a similar phenomenon might occur in our back-end optimization problem if each new loop closure is only applied once or maybe twice until enough subsequent loop closures have accumulated that we feel it appropriate to further tighten the initial loop closure.

Any principled analogy between back-end optimization and familiar physical systems stands to improve our understanding of how to further develop BERT

5.1.5 Identifying False Positive Loop Closures

Incorporating information from loop closure is the key difference between SLAM and regular odometry. However, the more information one tries to extract from a loop closure the more critical it becomes to make sure that loop closure is not a false positive. For example, in our concept of relative navigation, when a loop closure is identified the vehicle simply notes there is a relative transformation between two of the reference poses and estimates its value. Other relative transformation estimates are only affected by the loop closure estimate during the back-end optimization. If we were only navigating through the globally topological map generated by the front-end, then a false positive loop closure could have only a minor impact. However, when trying to find the best globally metric explanation for all the relative transformations, a false positive loop closure can cause the final solution to strongly diverge from the truth.

Very little literature in graph-based SLAM addresses the problem of false positive loop closures. However, we believe the coupling of BERT with relative navigation makes for a system that should be robust to this problem. Below we will sketch out an approach for rejecting false positive loop closures with the comment that we believe this to be a fruitful and straight forward avenue for further applying BERT.

We will assume that the system saves all of the initial estimates of the relative transformations delivered by the front-end. This would be necessary to facilitate an FEJ approach to linearization if that were used. It would also seem to be prudent in the context of identifying false positive loop closures. If the back-end ever decides its map has become corrupted it could revert to the initial estimates and redo optimization using only those loop closures deemed most reliable. We note that using a relative navigation scheme makes this reversion possible because the vehicle would not suffer from a sudden change to the global pose estimates.

We propose that a new loop closure be initially applied in only one iteration of BERT. At that point it should be marked as tentative. The vehicle would then continue to navigate, collecting additional loop closure estimates along the way that are also only applied in one iteration of BERT. We assume the ratio of false to true loop closure detections is small.

Then recall the squared Mahalanobis distance metric discussed in Chapter 1.2.3:

$$\epsilon(\boldsymbol{\rho}) = \sum \left(\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right) - \boldsymbol{\tau}_i^j\right)^\top \boldsymbol{\Sigma}_{i,j}^{-1} \left(\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right) - \boldsymbol{\tau}_i^j\right), \tag{5.1}$$

We propose that the system compute the summands in Equation (5.1) for *only* the loop closure transformations using the global poses obtained in the optimization up to the current time. We believe that false positive loop closures will present an inordinately large difference between the original estimate of the loop closure and the corresponding value of $\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)$. This should happen because the global poses will mostly reflect the influence of the larger proportion of true loop closures. When a summand from Equation (5.1) exceeds a predetermined threshold, that loop closure could be thrown out and the network could be reoptimized using the initial front-end estimates or perhaps some intermediate solution saved before incorporating the false information.

5.1.6 Developing a Good Termination Metric

Finally, we have not yet addressed how to determine when to cease iterating BERT. In Chapters 3 and 4 we manually decided when to terminate iterations. We present here some suggestions for controlling the number of times BERT is applied for each loop closure.

As noted above, it may be appropriate to apply each loop closure only once. Our assumption of relative navigation in the front-end allows us to patiently develop the globally metric map. Collecting several loop closures, each applied only once, may produce an accurate map without trying to eek out all of the improvement possible from each loop closure. There may also be theoretical justification for this based on the optimality of the Kalman filter used to incorporate the loop closure information.

Some measure of correlation may also provide an appropriate termination metric. After every iteration of applying the loop closure information the correlation between poses should increase, especially those poses involved in the loop closure. After several iterations those poses may become sufficiently correlated that we can decide further iteration is unnecessary.

Perhaps most obvious, some global measure of change in the pose and/or transformation estimates could also be taken from one iteration to the next. If the latest iteration makes a small enough change, there should be no reason to continue. This is likely related to the way correlations in the network evolve from iteration to iteration. We could use Equation (5.1) for all of the transformations being optimized, letting $\boldsymbol{\tau}_i^j$ and $\boldsymbol{\Sigma}_{i,j}$ represent all of the transformations' mean and covariance from the *previous* iteration. We would evaluate the term $\mathbf{h}\left(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j\right)$ using the global poses computed in the *current* iteration. If $\epsilon(\boldsymbol{\rho})$ is sufficiently small we would terminate iteration.

5.2 Concluding Summary

We strongly believe that relative navigation is necessary to develop an autonomous air vehicle operating in general unknown environments without access to global pose information. The relative transformations are the only thing the vehicle can observe. Navigating along the relative transformations makes the vehicle more robust to the errors in global pose estimates that are likely to occur in an unstructured setting.

In Chapter 2 we presented analysis and results that demonstrate the effectiveness of a dynamic state estimator for multirotor helicopters. The estimator leverages an improved dynamic model for multirotor helicopters to extract more information from the frequent and easy to process data available from the vehicle's gyroscopes and accelerometers. We showed how we can use gyroscopes to propagate our belief about the vehicle's remaining dynamic states, and we derived an approach for tuning the estimator to minimize the use of heuristics and account for the correlation between the state estimates. We also presented an observability analysis that explains how the improved model leads to better information about the states estimated by the filter. The results from testing the filter showed its robustness to noise in the exteroceptive information and its ability to estimate biases that would otherwise require careful calibration before flight.

We described in Chapter 1 how the estimator developed in Chapter 2 could be used in a relative navigation scheme. The vehicle periodically saves representative images of the environment. The vehicle plans its immediate paths and control efforts relative to those saved images. The sequence of saved images constitutes a locally-metric, globally-topological map of the world. In other words, without any further action the vehicle is equipped with a map it can repeatably traverse. However, the map provides a very limited sense of how disparate map elements are related. Some back-end optimization process must refine the network of saved poses and relative transformations to produce a globally metric map. This relationship between relative navigation and a globally optimized map leads to the contributions in Chapters 3 and 4.

In Chapter 3 we presented a new approach to back-end optimization that we named BERT. Whereas existing back-end optimization tools treat the relative transformations as fixed constraints, BERT optimizes these transformations directly and relies on them as its abiding representation of the world. We showed that global poses serve as mediating random variables to help maintain the conditional independence properties inherent in the graph-based SLAM problem. When iterated, we showed for a simple example that BERT converges to essentially the same global pose estimates in approximately the same amount of time as g^2o, a state-of-the-art back-end optimization algorithm. However, we showed that g^2o takes several iterations before it improves on the initial relative transformation estimates. In contrast, we showed that BERT improves the transformation estimates at every iteration.

BERT approaches the back-end optimization process from a direction completely distinct from algorithms extent in the literature, and it therefore offers some additional distinct advantages. BERT operates on the covariance form of the Gaussian representing our belief about the map. Therefore, the joint and marginal covariance of any global pose or relative transformation in the map is always available without additional computation. Other algorithms have to extract the marginal covariance of a global pose from the joint global pose information matrix in order to perform such tasks as rejecting false positive loop closures. And no other algorithm besides BERT gives attention to changes in the marginal covariance of a relative transformation as the map evolves over time.

The development in Chapter 3 tested BERT in a special case scenario. In Chapter 4 we extended BERT to a more complex scenario that will serve as a building block for completely general back-end optimization problems. We showed in Chapter 4 that virtually no additional computation is needed to compute a prior belief about an arbitrary loop closure. However, we showed that computing this prior in a naive manner leads to a divergent solution. We addressed this problem by drawing an analogy with work described in the EKF-SLAM literature treating the topic of a filter's statistical consistency. We can modify how we develop a loop closure distribution by using the first-ever estimates of states needed to calculate Jacobians; doing this maintains the efficiency of the approach and allows the solution to converge. We also showed the efficacy of incorporating loop closure information by forming the prior estimates of poses in the loop using the reference frame of the pose from which the loop closure originates.

We consider the future work on every aspect of the system described in Figure 5.0 to be reasonably well defined and promising. We look forward to witnessing and participating in its development.

Bibliography

[1] S. Shen, N. Michael, and V. Kumar, "Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 20–25. vii, 8, 9, 10, 11, 13, 23, 36, 37, 41

[2] A. Bachrach, S. Prentice, R. He, and N. Roy, "RANGE - Robust Autonomous Navigation in GPS-denied Environments," *Journal of Field Robotics*, vol. 28, no. 5, pp. 644–666, Sept. 2011. vii, 10, 11, 12, 13, 15, 17, 18, 23, 24, 27, 36, 37, 41

[3] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "PIXHAWK : A System for Autonomous Flight using Onboard Computer Vision," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 2992–2997. vii, 15, 16, 17, 18, 36, 37

[4] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a Navigation System for Autonomous Indoor Flying," in *2009 IEEE International Conference on Robotics and Automation*, no. Section III, May 2009, pp. 2878–2883. 1, 5, 6, 7, 8, 13, 23

[5] V. Kumar and N. Michael, "Opportunities and Challenges with Autonomous Micro Aerial Vehicles," in *15th International Sympossium on Robotics Research*, 2011. 2, 3

[6] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A Tutorial on Graph-Based SLAM," *IEEE Intelligent Transportation Systems Magazine*, pp. 31–43, Jan. 2010. 3, 19, 23, 24, 30, 81

[7] J. C. Macdonald, "Mikrokopter Serial Control Tutorial," Tech. Rep., 2011. [Online]. Available: http://hdl.lib.byu.edu/1877/2747 5

[8] MAGGIC_LAB, "Mikrokopter Serial Control Files." [Online]. Available: http://hdl.lib.byu.edu/1877/2748 5

[9] J. C. Macdonald and R. Leishman, "BYU Indoor Flight System Circa June 2011," Tech. Rep., 2011. [Online]. Available: http://hdl.lib.byu.edu/1877/2937 5

[10] I. Sa and P. Corke, "System Identification, Estimation and Control for a Cost Effective Open-Source Quadcopter," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 2202–2209. 5

[11] S. Grzonka, G. Grisetti, and W. Burgard, "A Fully Autonomous Indoor Quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, Feb. 2012. 5, 7, 8, 9, 23

[12] G. Tournier, M. Valenti, J. P. How, and E. Feron, "Estimation and Control of a Quadrotor Vehicle Using Monocular Vision and Moire Patterns," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, no. August, 2006, pp. 21–24. 6

[13] P. Rudol and M. Wzorek, "Vision-based Pose Estimation for Autonomous Indoor Navigation of Micro-scale Unmanned Aircraft Systems," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 1913–1920. 6

[14] W. Li, T. Zhang, and K. Kuhnlenz, "A Vision-Guided Autonomous Quadrotor in An Air-Ground Multi-Robot System," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 2980–2985. 6

[15] P. Rudol, M. Wzorek, G. Conte, and P. Doherty, "Micro Unmanned Aerial Vehicle Visual Servoing for Cooperative Indoor Exploration," in *2008 IEEE Aerospace Conference*, Mar. 2008, pp. 1–10. 6

[16] S. P. Soundararaj, A. K. Sujeeth, and A. Saxena, "Autonomous Indoor Helicopter Flight Using a Single Onboard Camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 5307–5314. 6

[17] E. Rondon, L. Garcia-Carrillo, and I. Fantoni, "Vision-Based Altitude, Position and Speed Regulation of a Quadrotor Rotorcraft," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 628–633. 6

[18] J. Eckert, R. German, and F. Dressler, "An Indoor Localization Framework for Four-rotor Flying Robots Using Low-power Sensor Nodes," *Instrumentation and Measurement, IEEE Transactions on*, vol. 60, no. 99, pp. 1–9, 2011. 6

[19] S. Shen, N. Michael, and V. Kumar, "Autonomous Indoor 3D Exploration with a Micro-Aerial Vehicle," *2012 IEEE International Conference on Robotics and Automation*, pp. 9–15, May 2012. 10, 23

[20] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments," in *Proceedings of the First Symposium on Indoor Flight Issues*, 2008. 10

[21] ——, "Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-denied Indoor Environments," in *Proceedings of the SPIE Conference on Unmanned Systems Technology XI*, vol. 1. SPIE, 2009. 10

[22] A. Bachrach, R. He, and N. Roy, "Autonomous Flight in Unstructured and Unknown Indoor Environments," in *Proceedings of the European Micro Aerial Vehicle Conference and Flight Competition*, 2009, pp. 2–9. 10

[23] ——, "Autonomous Flight in Unknown Indoor Environments," *International Journal of Micro Air Vehicles*, vol. 1, no. 4, pp. 217–228, Dec. 2009. 10

[24] S. Ahrens, D. Levine, G. Andrews, and J. P. How, "Vision-Based Guidance and Control of a Hovering Vehicle in Unknown, GPS-denied Environments," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 2643–2648. 14, 15

[25] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “MonoSLAM: Real-time Single Camera SLAM.” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–67, June 2007. 14

[26] J. Civera, A. J. Davison, and J. Montiel, “Inverse Depth Parametrization for Monocular SLAM,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008. 14

[27] P. Pinies and J. D. Tardós, “Large-Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1094–1106, Oct. 2008. 14, 21, 22, 33, 62, 73, 74, 96, 120, 127

[28] M. Blosch, S. Weiss, D. Scaramuzza, and R. Seigwart, “Vision Based MAV Navigation in Unknown and Unstructured Environments,” in *IEEE International Conference on Robotics and Automation*, 2010, pp. 21–28. 14, 15, 23

[29] G. Klein and D. Murray, “Parallel Tracking and Mapping for Small AR Workspaces,” in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007. 15

[30] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. 16, 17, 23, 36

[31] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Fox, and N. Roy, “Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera,” in *International Symposium on Robotics Research*, 2011, pp. 1–16. 17, 18, 23

[32] R. Smith, M. Self, and P. Cheeseman, “Estimating Uncertain Spatial Relationships in Robotics,” in *Autonomous Robot Vehicles.* Springer-Verlag New York, Inc, 1990, pp. 167—-193. 19, 33

[33] M. Csorba, “Simultaneous Localisation and Map Building,” PhD Thesis, University of Oxford, Aug. 1997. 19

[34] S. Thrun and J. J. Leonard, “Simultaneous Localization and Mapping,” in *Handbook of Robotics*, B. Siliciano and O. Khatib, Eds. Springer, 2008, ch. 37, pp. 871–889. 19, 20, 22

[35] H. Durrant-Whyte and T. Bailey, “Simultaneous Localisation and Mapping (SLAM): Part I. The Essential Algorithms,” *Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. 19, 22, 133

[36] T. Bailey and H. Durrant-Whyte, “Simultaneous Localization and Mapping (SLAM): Part II,” *Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006. 19

[37] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent robotics and autonomous agents. The MIT Press, Aug. 2005. 19, 22

[38] "SLAM Summer School, 2009." [Online]. Available: http://www.acfr.usyd.edu.au/education/summerschool.shtml 19, 22

[39] P. Furgale and T. Barfoot, "Visual Path Following on a Manifold in Unstructured Three-Dimensional Terrain," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 534–539. 19

[40] J. Blanco, J. Fernandez-Madrigal, and J. Gonzalez, "Toward a Unified Bayesian Approach to Hybrid Metric–Topological SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 259–270, Apr. 2008. 19

[41] R. Paul and P. M. Newman, "FAB-MAP 3D: Topological Mapping with Spatial and Visual Appearance," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 2649–2656. 19

[42] S. Julier and J. Uhlmann, "A Counter Example to the Theory of Simultaneous Localization and Map Building," in *IEEE International Conference on Robotics and Automation*, vol. 4. Citeseer, 2001, pp. 4238–4243. 21, 120

[43] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM Algorithm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3562–3568. 21, 120, 132

[44] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li, *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc., 2002. 21, 32, 132

[45] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and Improvement of the Consistency of Extended Kalman Filter Based SLAM," in *Proceedings of the 2008 International Conference on Robotics and Automation (ICRA)*. IEEE, 2008, pp. 473–479. 21, 121, 122, 131

[46] ——, "Observability-Based Rules for Designing Consistent EKF SLAM Estimators," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 502–528, Dec. 2009. 21, 131

[47] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," in *Proceedings of the National conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002, pp. 593–598. 22

[48] M. Montemerlo and S. Thrun, "Simultaneous Localization and Mapping with Unknown Data Association using FastSLAM," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, pp. 1985–1991, 2003. 22

[49] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the FastSLAM Algorithm," *Proceedings of IEEE International Conference on Robotics and Automation*, no. 1, pp. 424–429, 2006. 22

[50] F. Lu and E. Milios, “Globally Consistent Range Scan Alignment for Environment Mapping,” *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997. 22, 23, 28, 29, 33

[51] A. Howard, M. J. Mataric, and G. Sukhatme, “Relaxation on a Mesh : a Formalism for Generalized Localization,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 1055–1060. 29, 30, 33, 133

[52] U. Frese, P. Larsson, and T. Duckett, “A Multilevel Relaxation Algorithm for Simultaneous Localization and Mapping,” *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 196–207, Apr. 2005. 29, 30, 33, 84

[53] M. Montemerlo and S. Thrun, “Large-Scale Robotic 3-D Mapping of Urban Structures,” in *International Symposium on Experimental Robotics*, 2004. 29, 30, 33

[54] S. Thrun and M. Montemerlo, “The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures,” *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, May 2006. 29, 30, 33

[55] F. Dellaert and M. Kaess, “Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing,” *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006. 29, 31, 33, 132

[56] M. Kaess, A. Ranganathan, and F. Dellaert, “iSAM: Incremental Smoothing and Mapping,” *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008. 29, 31, 33

[57] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “iSAM2 : Incremental Smoothing and Mapping Using the Bayes Tree,” *International Journal of Robotics Research*, 2012. 29, 31, 33

[58] E. B. Olson, J. J. Leonard, and S. Teller, “Fast Iterative Alignment of Pose Graphs with Poor Initial Estimates,” in *IEEE International Conference on Robotics and Automation*, no. May, 2006, pp. 2262–2269. 29, 31, 33, 68, 84

[59] G. Grisetti, D. Rizzini, C. Stachniss, E. B. Olson, and W. Burgard, “Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning,” in *Proc. of the IEEE Int. Conf. on Robotics \& Automation (ICRA)*, 2008, pp. 1880–1885. 29, 31, 33

[60] G. Grisetti, C. Stachniss, and W. Burgard, “Nonlinear Constraint Network Optimization for Efficient Map Learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428–439, Sept. 2009. 29, 31, 32, 33, 86

[61] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent, “Efficient Sparse Pose Adjustment for 2D mapping,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems.* Ieee, Oct. 2010, pp. 22–29. 29, 31, 33

[62] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 273–278. 29, 31, 32, 33

[63] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A General Framework for Graph Optimization," in *IEEE International Conference on Robotics and Automation*, 2011. 29, 31, 32, 33, 76

[64] M. Bosse, P. M. Newman, J. J. Leonard, M. Soika, W. Feiten, and S. Teller, "An ATLAS Framework for Scalable Mapping," *IEEE International Conference on Robotics and Automation*, pp. 1899–1906, 2003. 29, 32, 33

[65] C. Estrada, J. Neira, and J. D. Tardós, "Hierarchical SLAM : Real-Time Accurate Mapping of Large Environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005. 29, 32, 33

[66] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer-Verlag New York, Inc., 2006. 31, 62, 63, 65, 69, 150, 152

[67] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The Bayes Tree : An Algorithmic Foundation for Probabilistic Robot Mapping," in *International Workshop on the Algorithmic Foundations of Robotics*, 2010. 31

[68] L. Paz, P. Pinies, J. D. Tardós, and J. Neira, "Large-Scale 6-DOF SLAM With Stereo-in-Hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008. 33, 62, 73

[69] P. Pinies, L. M. Paz, and J. D. Tardós, "CI-Graph: An Efficient Approach for Large Scale SLAM," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 3913–3920. 33, 62, 73, 74

[70] P. Pinies, L. M. Paz, D. Galvez-Lopez, and J. D. Tardós, "CI-Graph Simultaneous Localization and Mapping for Three-Dimensional Reconstruction of Large and Complex Environments Using a Multicamera System," *Journal of Field Robotics*, vol. 27, no. 5, pp. 561–586, 2010. 33, 62, 73

[71] R. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain, "Improved Use of Accelerometers in Estimating Quadrotor Attitude and Velocity," *IEEE Robotics & Automation Magazine (in review; preprint available from the authors).* 37, 40, 41, 42, 54, 55, 68, 74, 75

[72] P. Martin and E. Salaün, "The True Role of Accelerometer Feedback in Quadrotor Control," *IEEE International Conference on Robotics and Automation*, pp. 1623–1629, 2010. 40

[73] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. 43

[74] R. Leishman, J. Macdonald, S. Quebe, J. Ferrin, R. Beard, and T. Mclain, "Utilizing an Improved Rotorcraft Dynamic Model in State Estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011. 47

[75] M. Vidyasagar, *Nonlinear Systems Analysis (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992. 48, 147

[76] R. W. Beard and T. McLain, *Small Unmanned Aircraft*. Princeton University Press, 2012. 52

[77] S. Hutchinson and F. Chaumette, "Visual Servo Control Part I: Basic Approaches," *IEEE Robotics and Automation Magazine*, vol. 12, no. 5, pp. 651–670, 2006. 53

[78] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based Maps," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 941–957, May 2010. 53, 60

[79] J. Redding, Z. Dydek, J. P. How, M. A. Vavrina, and J. Vian, "Proactive Planning for Persistent Missions Using Composite Model-Reference Adaptive Control and Approximate Dynamic Programming," in *American Control Conference*, 2011, pp. 2332–2337. 54

[80] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz," in *IEEE Int. Conf. on Robotics and Automation*, no. April, 2007, pp. 10–14. 60

[81] U. B. Kjaerulff and A. L. Madsen, *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer, 2008. 62, 63, 64, 71

[82] J. C. Macdonald, R. Leishman, R. W. Beard, and T. McLain, "Analysis of an Improved IMU-Based Observer for Multirotor Helicopters," *Journal of Intelligent & Robotic Systems (in review; preprint available from the authors)*. 74

[83] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment - A Modern Synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ser. ICCV '99. London, UK: Springer-Verlag, 2000, pp. 298–372. 87

[84] R. Hermann and A. Krener, "Nonlinear Controllability and Observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, Oct. 1977. 147

[85] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 2009. 153

[86] J. C. Macdonald, "Technical Note on Manipulating Gaussian Covariance Matrices," 2012. [Online]. Available: www.INeedAURL.byu.edu 155

Appendix A

Nonliner Obsevability Analysis: Theory and Example

Observability is a necessary condition for a filter to converge. Nonlinear observability analysis typically follows the approach presented in [84]. A similar and easy to follow treatment is given in [75, Chapter 7]. We adopt the notation of the latter source with some modifications for our presentation and refer the reader to [75] for a more detailed presentation of the theory.

A.1 Theory

Let X denote an open subset of $\mathbb{R}^{\mathrm{N}}$. Let $S(X)$ and $V(X)$ respectively designate the set of all scalar valued smooth functions and the set of all vector fields (i.e. column vectors of smooth functions) on X. Recall that the Lie derivative of a function $\kappa \in S(X)$ with respect to any vector field $\boldsymbol{\omega} \in V(X)$ is defined by the mapping

$$\mathbf{x} \mapsto \mathbf{d}\kappa(\mathbf{x}) \cdot \boldsymbol{\omega}(\mathbf{x}) : X \to \mathbb{R}, \tag{A.1}$$

where $\mathbf{d}\kappa$ is the gradient of κ with respect to $\mathbf{x} \in X$. This Lie derivative is given the notation $L_{\boldsymbol{\omega}}\kappa$. Note that the result of the Lie derivative is also in $S(X)$.

Now let $\mathbf{x} \in X$ represent the state of the nonlinear system. The states evolve in time according to the equation

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \sum_{j=1}^{p} \mathrm{u}_j \mathbf{g}_j(\mathbf{x}), \tag{A.2}$$

where the time varying scalars u_j represent the inputs driving the state evolution, p is the number of inputs, and $\mathbf{f},\ \mathbf{g}_j \ \in \ V(X)$. The nonlinear system also provides outputs $y_n = h_n(\mathbf{x})$, with $n \in [1, q]$.

We say that two states $\mathbf{x}_0, \mathbf{x}_1 \in X$ are **distinguishable** if there exists an input function $\mathbf{u} \triangleq [\mathrm{u}_1, \mathrm{u}_2, \ldots, \mathrm{u}_p]^\top$ such that the output function $\mathbf{h}(\mathbf{x}_0) \triangleq [h_1(\mathbf{x}_0), h_2(\mathbf{x}_0), \ldots, h_q(\mathbf{x}_0)]^\top$ does not equal $\mathbf{h}(\mathbf{x}_1)$. The system is said to be **locally observable at** $\mathbf{x}_0 \in X$ if there exists a neighborhood around $\mathbf{x}_0$ in which every other $\mathbf{x}$ in the neighborhood is distinguishable from $\mathbf{x}_0$. Finally, we say that the system is **locally observable** if it is locally observable at each $\mathbf{x}_0 \in X$.

For the system to be locally observable at a given point $\mathbf{x}_0 \in X$, it is sufficient to show that there exist N linearly independent row vectors in the set

$$\left\{ \mathbf{d}h_n\big|_{\mathbf{x}_0} \right\} \cup \left\{ \mathbf{d}L_{\mathbf{z}_s} L_{\mathbf{z}_{s-1}} \cdots L_{\mathbf{z}_1} h_n\big|_{\mathbf{x}_0} \right\}, \tag{A.3}$$

where $s \geq 1$, and $\mathbf{z}_j \in \{\mathbf{f}, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$. In other words, if one can find enough linearly independent vectors in the gradients of the measurement equations or in the gradients of their Lie derivatives evaluated at $\mathbf{x}_0$, the system is locally observable at that point.

A.2 Example

We offer a simple example to make the theory more concrete. This example also relates directly to the analysis in Chapter 2.3. We will use it to explain why the improved dynamic model for a multi-rotor helicopter allows accelerometer measurements to give information about attitude, velocity, and gyroscope biases.

First, we define the system in our example. The state vector for the system is $\mathbf{x} \triangleq [\theta, u, \beta_{\mathrm{j_b}}]^\top$, where θ is the pitch angle about the helicopter's body-fixed $\mathbf{j}_\mathrm{b}$ axis, u is the linear velocity of the helicopter in its body-fixed $\mathbf{i}_\mathrm{b}$ axis, and $\beta_{\mathrm{j_b}}$ is the bias in the gyroscope measuring angular rate about the body-fixed $\mathbf{j}_\mathrm{b}$ axis. We will assume that all of the vehicle's remaining attitude, velocity, and bias states are equal to zero. We will also assume that θ is small such that $\sin\theta \approx \theta$.

Under the assumptions in this example, $\dot{\theta} = q$, the rotation rate about the body-fixed $\mathbf{j}_\mathrm{b}$ axis. We define the gyroscope measurement as the only input to the system; i.e. $\mathrm{u}_1 = \dot{\theta} + \beta_{\mathrm{j_b}}$. Then we can specify Equation (A.2) for this example as

$$\begin{bmatrix} \dot{\theta} \\ \dot{u} \\ \dot{\beta}_{\mathrm{j_b}} \end{bmatrix} = \mathbf{f}(\mathbf{x}) + \mathrm{u}_1 \mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} -\beta_{\mathrm{j_b}} \\ -g\theta - \frac{\mu}{m}u \\ \zeta \end{bmatrix} + \mathrm{u}_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{A.4}$$

where g is the acceleration due to gravity, μ is an aerodynamic coefficient that transforms velocity into a drag force, m is the mass of the vehicle, and ζ is a zero mean Gaussian random variable (i.e. the bias evolves as a random walk). We define the accelerometer measurement aligned with the body-fixed $\mathbf{i}_\mathrm{b}$ axis to be the only output y_1 from the system; it is modeled as $h_1 = -\frac{\mu}{m}u$.

To show that the system is locally observable at a point $\mathbf{x}_0 \in X$ we need to find three vectors that are linearly independent at $\mathbf{x}_0$. We begin with the gradient of the output:

$$\mathbf{d}h_1 = \begin{bmatrix} 0 & -\frac{\mu}{m} & 0 \end{bmatrix}. \tag{A.5}$$

Next we try a first order Lie derivative. It is obvious that $\mathbf{d}L_{\mathbf{g}_1}h_1 = [0, 0, 0]$, so we consider

$$\begin{aligned} \mathbf{d}L_{\mathbf{f}}h_1 &= \mathbf{d}\left(\begin{bmatrix} 0 & -\frac{\mu}{m} & 0 \end{bmatrix} \begin{bmatrix} -\beta_{\mathrm{j_b}} \\ -g\theta - \frac{\mu}{m}u \\ \zeta \end{bmatrix} \right) \\ &= \begin{bmatrix} g\frac{\mu}{m} & \left(\frac{\mu}{m}\right)^2 & 0 \end{bmatrix}. \end{aligned} \tag{A.6}$$

So far we have two linearly independent row vectors that are both equal to zero in the third column. To find a third vector, we need a Lie derivative that is a function of $\beta_{\mathrm{j_b}}$ so that the gradient of that Lie derivative may have a nonzero entry in the third column. The only option is

$$\begin{aligned}\mathbf{d}L_{\mathbf{f}}L_{\mathbf{f}}h_1 &= \mathbf{d}\left(\begin{bmatrix} g\frac{\mu}{m} & \left(\frac{\mu}{m}\right)^2 & 0\end{bmatrix}\begin{bmatrix}-\beta_{\mathrm{j_b}} \\ -g\theta - \frac{\mu}{m}u \\ \zeta\end{bmatrix}\right) \\ &= \begin{bmatrix}-g\left(\frac{\mu}{m}\right)^2 & -\left(\frac{\mu}{m}\right)^3 & -g\frac{\mu}{m}\end{bmatrix}. \end{aligned} \tag{A.7}$$

Since the three linearly independent vectors in Equations (A.5) - (A.7) do not depend on the states, the system is locally observable at each $\mathbf{x}_0 \in X$.

We now draw connections between the observability analysis in this example and the physical attributes of the system. A Lie derivative can be thought of as the rate of change of a function constrained by the evolution of the system. The gradient of the Lie derivative gives a measure of each state's contribution to that rate of change. We can use these intuitive notions to gain insight into why the single accelerometer measurement aligned with the body-fixed $\mathbf{i}_{\mathrm{b}}$ axis provides information about the linear velocity, pitch angle, and gyro bias in this example.

Consider the order of the Lie derivatives involved in the preceding analysis. The zeroth order Lie derivative of h_1, i.e. $\mathbf{d}h_1$, indicates the velocity is observable. This is natural; the specific force the accelerometer measures in this system is a scaled component of linear velocity, as discussed in Chapter 2.1. The first order Lie derivative shows that the pitch angle is observable. This agrees with the physical reality that a change in pitch angle directly causes the change in the specific force. Similarly, the change in rotation rate measured by the biased gyro causes the change in pitch that causes the change in velocity. In other words, a change in rotation rate has a second order effect on the quantity being measured by the accelerometer.

In summary, an observability analysis supports the assertion that the model presented in Chapter 2 is a correct model for the pose and velocity states of a multi-rotor air vehicle. The full state estimator can be shown to be observable, and the components in the observability analysis agree with the underlying physics of the system. We present a complete observability analysis for a multi-rotor air vehicle in Chapter 2

Appendix B

Details of Manipulating Jointly Gaussian Random Variables

In this appendix we present some derivations treating conditional, marginal, and joint distributions for Gaussian random vectors. First we present the intermediate steps that allow the conditional distribution $\mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)$ to be rewritten in a form that admits easier multiplication with the marginal distribution $\mathrm{p}(\mathbf{x}_2)$. As a corollary we make some observations about the relationship that exists between the information matrices for the conditional, marginal, and joint distributions. We conclude with a derivation showing how to recover the joint covariance matrix of $\mathrm{p}(\mathbf{x}_1, \mathbf{x}_2)$ without matrix inversion to allow information updated in the marginal to be distributed to the remaining variables in the joint.

Most of the motivation for these derivations comes from an apparently unpublished note written by Paul Newman and John Leonard of MIT in November 2002 titled "A Matrix Oriented Note on Joint, Marginal, and Conditional Multivariate Gaussian Distributions." We correct some small errors in the version we found of that note, and we extend the results to include the closed form expressions for recovering the optimized joint mean and covariance without inverting the optimized joint information matrix.

B.1 Rewriting the Conditional Distribution

Let a D-dimensional random vector $\mathbf{x}$ of jointly Gaussian variables be partitioned into two, disjoint sub-vectors such that $\mathbf{x} = \left[\mathbf{x}_1^\top, \mathbf{x}_2^\top\right]^\top$, where $\mathbf{x}_1$ is dimension D_1 and $\mathbf{x}_2$ is dimension D_2. Then the joint distribution $\mathrm{p}(\mathbf{x})$, with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, is partitioned such that

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix},$$
$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}.$$

Typical textbook derivations (e.g. [66], Chapter 2.3.1) define the conditional distribution $\mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)$ such that $\log(\mathrm{p}(\mathbf{x}_1|\mathbf{x}_2))$ is proportional to

$$\left(\mathbf{x}_1 - \boldsymbol{\mu}_{1|2}\right)^\top \boldsymbol{\Sigma}_{1|2}^{-1} \left(\mathbf{x}_1 - \boldsymbol{\mu}_{1|2}\right), \tag{B.1}$$

where $\boldsymbol{\mu}_{1|2}$ and $\boldsymbol{\Sigma}_{1|2}$ are given by

$$\boldsymbol{\mu}_{1|2} \triangleq \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\left(\mathbf{x}_2 - \boldsymbol{\mu}_2\right), \tag{B.2}$$

$$\boldsymbol{\Sigma}_{1|2} \triangleq \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}. \tag{B.3}$$

To simplify notation in the sequel we define

$$\mathbf{K} \triangleq \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}. \tag{B.4}$$

There are a few undesirable aspects of expressing the conditional distribution $\mathrm{p}\left(\mathbf{x}_1|\mathbf{x}_2\right)$ using (B.1). First, the mean vector $\boldsymbol{\mu}_{1|2}$ has dimension D_1. The marginal distribution $\mathrm{p}\left(\mathbf{x}_2\right)$ has a mean $\boldsymbol{\mu}_2$ of dimension D_2. To recover the joint distribution $\mathrm{p}\left(\mathbf{x}\right) = \mathrm{p}\left(\mathbf{x}_1|\mathbf{x}_2\right)\mathrm{p}\left(\mathbf{x}_2\right)$ would require that we sum exponents with different dimensions. It is also unattractive to leave the conditional distribution's functional dependence on $\mathbf{x}_2$ buried in the conditional mean.

Define the following variables:

$$\mathbf{M} \triangleq \begin{bmatrix}\mathbf{I}_{D_1} & -\mathbf{K}\end{bmatrix},$$
$$\mathbf{g} \triangleq \left(\mathbf{x} - \boldsymbol{\mu}\right) = \begin{bmatrix}\mathbf{x}_1 - \boldsymbol{\mu}_1 \\ \mathbf{x}_2 - \boldsymbol{\mu}_2\end{bmatrix},$$

where $\mathbf{I}_{D_1}$ is a D_1 x D_1 identity matrix. Note that $\mathbf{M}$ is size D_1 x D. Now observe that

$$\begin{aligned}\mathbf{M}\mathbf{g} &= \mathbf{x}_1 - \boldsymbol{\mu}_1 - \mathbf{K}\left(\mathbf{x}_2 - \boldsymbol{\mu}_2\right) \\ &= \mathbf{x}_1 - \boldsymbol{\mu}_{1|2}.\end{aligned} \tag{B.5}$$

Substituting Equation (B.5) into Equation (B.1) gives

$$\begin{aligned}\left(\mathbf{M}\mathbf{g}\right)^\top \boldsymbol{\Sigma}_{1|2}^{-1}\left(\mathbf{M}\mathbf{g}\right) &= \mathbf{g}^\top\mathbf{M}^\top\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{M}\mathbf{g} \\ &= \left(\mathbf{x} - \boldsymbol{\mu}\right)^\top \mathbf{A}\left(\mathbf{x} - \boldsymbol{\mu}\right),\end{aligned} \tag{B.6}$$

where $\mathbf{A} \triangleq \mathbf{M}^\top\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{M}$ is the information matrix associated with the conditional distribution. We can expand this expression for $\mathbf{A}$ to find that

$$\mathbf{A} = \begin{bmatrix}\boldsymbol{\Sigma}_{1|2}^{-1} & -\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{K} \\ -\mathbf{K}^\top\boldsymbol{\Sigma}_{1|2}^{-1} & \mathbf{K}^\top\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{K}\end{bmatrix}. \tag{B.7}$$

Equation (B.6) gives an expression for the conditional distribution where its functional dependence on both $\mathbf{x}_1$ and $\mathbf{x}_2$ is in a more standard form. We can similarly rewrite the marginal $\mathrm{p}\left(\mathbf{x}_2\right)$ such that $\log\left(\mathrm{p}\left(\mathbf{x}_2\right)\right)$ is proportional to

$$\left(\mathbf{x} - \mathbf{b}\right)^\top \mathbf{B}\left(\mathbf{x} - \mathbf{b}\right),$$

where the D-dimensional vector $\mathbf{b}$ is defined as

$$\mathbf{b} \triangleq \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mu}_2 \end{bmatrix},$$

and

$$\mathbf{B} \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{22}^{-1} \end{bmatrix}. \tag{B.8}$$

B.2 A Short Observation on Information Matrices

A nice piece of intuition arises from writing the conditional and marginal distributions in the manner just described. The block partitioned information matrix for the joint distribution can be written [66] as

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{1|2}^{-1} & -\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{K} \\ -\mathbf{K}^\top\boldsymbol{\Sigma}_{1|2}^{-1} & \boldsymbol{\Sigma}_{22}^{-1} + \mathbf{K}^\top\boldsymbol{\Sigma}_{1|2}^{-1}\mathbf{K} \end{bmatrix}. \tag{B.9}$$

Comparing Equations (B.7) and (B.8) to Equation (B.9), it is clear that conditioning $\mathbf{x}_1$ on $\mathbf{x}_2$ amounts to removing the marginal information associated with $\mathbf{x}_2$; i.e.

$$\mathbf{A} = \boldsymbol{\Sigma}^{-1} - \mathbf{B}.$$

Conversely, when we recover a joint distribution, perhaps with an improved marginal belief about $\mathbf{x}_2$, we are simply adding that information back in:

$$\boldsymbol{\Sigma}^{-1} = \mathbf{A} + \mathbf{B}.$$

B.3 Deriving the Updated Joint Covariance Without Matrix Inversion

Next we present the steps that lead to a closed form expression for the joint covariance that has been optimized to include new information from the marginal distribution. Let $\mathrm{p}(\check{\mathbf{x}}_2)$, with mean $\check{\boldsymbol{\mu}}_2$ and information matrix $\check{\boldsymbol{\Sigma}}_{22}^{-1}$, represent our updated belief about the marginal states after incorporating information from a measurement. We also define

$$\check{\mathbf{b}} \triangleq \begin{bmatrix} \mathbf{0} \\ \check{\boldsymbol{\mu}}_2 \end{bmatrix},$$
$$\check{\mathbf{B}} \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \check{\boldsymbol{\Sigma}}_{22}^{-1} \end{bmatrix}.$$

To recover the optimized joint distribution $\mathrm{p}\left(\overset{\star}{\mathbf{x}}\right)$, we must perform the multiplication $\mathrm{p}(\mathbf{x}_1|\mathbf{x}_2)\mathrm{p}(\check{\mathbf{x}}_2)$. The product leads to

$$\begin{aligned} \log\left(\mathrm{p}\left(\overset{\star}{\mathbf{x}}\right)\right) &\propto (\mathbf{x} - \boldsymbol{\mu})^\top\mathbf{A}(\mathbf{x} - \boldsymbol{\mu}) + (\mathbf{x} - \check{\mathbf{b}})^\top\check{\mathbf{B}}(\mathbf{x} - \check{\mathbf{b}}) \\ &\propto (\mathbf{x} - \overset{\star}{\boldsymbol{\mu}})^\top\overset{\star}{\boldsymbol{\Sigma}}{}^{-1}(\mathbf{x} - \overset{\star}{\boldsymbol{\mu}}), \end{aligned}$$

where the optimized joint covariance and mean are

$$\begin{aligned}\overset{\star}{\mathbf{\Sigma}} &\triangleq \left(\mathbf{A} + \check{\mathbf{B}}\right)^{-1} \\ &= \begin{bmatrix} \mathbf{\Sigma}_{1|2}^{-1} & -\mathbf{\Sigma}_{1|2}^{-1}\mathbf{K} \\ -\mathbf{K}^\top\mathbf{\Sigma}_{1|2}^{-1} & \check{\mathbf{\Sigma}}_{22}^{-1} + \mathbf{K}^\top\mathbf{\Sigma}_{1|2}^{-1}\mathbf{K} \end{bmatrix}^{-1}, \qquad \text{(B.10)} \\ \overset{\star}{\boldsymbol{\mu}} &\triangleq \overset{\star}{\mathbf{\Sigma}}\left(\mathbf{A}\boldsymbol{\mu} + \check{\mathbf{B}}\check{\mathbf{b}}\right). \qquad \text{(B.11)}\end{aligned}$$

We will use the following identity (Chapter 0.7.3 of [85]) for the inverse of a block partitioned matrix:

$$\begin{bmatrix} \mathbf{W} & \mathbf{X} \\ \mathbf{Y} & \mathbf{Z} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{\Delta}_1 & -\mathbf{W}^{-1}\mathbf{X}\mathbf{\Delta}_2 \\ -\mathbf{\Delta}_2\mathbf{Y}\mathbf{W}^{-1} & \mathbf{\Delta}_2 \end{bmatrix}, \qquad \text{(B.12)}$$

where

$$\begin{aligned}\mathbf{\Delta}_1 &\triangleq \left(\mathbf{W} - \mathbf{X}\mathbf{Z}^{-1}\mathbf{Y}\right)^{-1} \\ &= \mathbf{W}^{-1} + \mathbf{W}^{-1}\mathbf{X}\mathbf{\Delta}_2\mathbf{Y}\mathbf{W}^{-1}, \qquad \text{(B.13)}\end{aligned}$$

and

$$\mathbf{\Delta}_2 \triangleq \left(\mathbf{Z} - \mathbf{Y}\mathbf{W}^{-1}\mathbf{X}\right)^{-1}. \qquad \text{(B.14)}$$

Comparing Equation (B.10) and Equation (B.12) we have

$$\begin{aligned}\mathbf{W} &= \mathbf{\Sigma}_{1|2}^{-1}, \\ \mathbf{X} &= -\mathbf{\Sigma}_{1|2}^{-1}\mathbf{K}, \\ \mathbf{Y} &= -\mathbf{K}^\top\mathbf{\Sigma}_{1|2}^{-1}, \\ \mathbf{Z} &= \check{\mathbf{\Sigma}}_{22}^{-1} + \mathbf{K}^\top\mathbf{\Sigma}_{1|2}^{-1}\mathbf{K}. \qquad \text{(B.15)}\end{aligned}$$

Our goal is to recover the optimized joint covariance matrix $\overset{\star}{\mathbf{\Sigma}}$ without matrix inversion. We begin by finding the bottom right block $\overset{\star}{\mathbf{\Sigma}}_{22}$ of the new joint covariance; this corresponds to the bottom right block $\mathbf{\Delta}_2$ of Equation (B.12). Making the appropriate substitutions into Equation (B.14) leads to

$$\begin{aligned}
\overset{\star}{\mathbf{\Sigma}}_{22} &= \mathbf{\Delta}_2 \\
&= \left[\mathbf{Z} - \left(-\mathbf{K}^\top \mathbf{\Sigma}_{1|2}^{-1}\right)\left(\mathbf{\Sigma}_{1|2}\right)\left(-\mathbf{\Sigma}_{1|2}^{-1}\mathbf{K}\right)\right]^{-1} \\
&= \left[\mathbf{Z} - \mathbf{K}^\top \mathbf{\Sigma}_{1|2}^{-1}\mathbf{K}\right]^{-1} \\
&= \left[\check{\mathbf{\Sigma}}_{22}^{-1} + \mathbf{K}^\top \mathbf{\Sigma}_{1|2}^{-1}\mathbf{K} - \mathbf{K}^\top \mathbf{\Sigma}_{1|2}^{-1}\mathbf{K}\right]^{-1} \\
&= \check{\mathbf{\Sigma}}_{22}. \qquad \text{(B.16)}
\end{aligned}$$

This agrees nicely with intuition. The optimized uncertainty for the variables $\overset{\star}{\mathbf{x}}_2$ in the joint distribution is the same as the uncertainty in the updated marginal distribution.

Now consider the upper left block $\overset{\star}{\mathbf{\Sigma}}_{11}$; this corresponds to $\mathbf{\Delta}_1$ in Equation (B.12). We first recall from Equation (B.3) that $\mathbf{W} = \mathbf{\Sigma}_{1|2}^{-1}$ can be rewritten such that

$$\begin{aligned}
\mathbf{W}^{-1} &= \mathbf{\Sigma}_{1|2} \\
&= \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}\mathbf{\Sigma}_{21} \\
&= \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}\mathbf{\Sigma}_{22}\mathbf{\Sigma}_{22}^{-1}\mathbf{\Sigma}_{21} \\
&= \mathbf{\Sigma}_{11} - \mathbf{K}\mathbf{\Sigma}_{22}\mathbf{K}^\top. \qquad \text{(B.17)}
\end{aligned}$$

Then observe that

$$\begin{aligned}
\mathbf{W}^{-1}\mathbf{X} &= \left(\mathbf{\Sigma}_{1|2}\right)\left(-\mathbf{\Sigma}_{1|2}^{-1}\mathbf{K}\right) \\
&= -\mathbf{K} \\
&= \left(\mathbf{Y}\mathbf{W}^{-1}\right)^\top. \qquad \text{(B.18)}
\end{aligned}$$

Using Equations (B.19), (B.17), and (B.18) allows us to simplify Equation (B.13) such that

$$\begin{aligned}
\overset{\star}{\mathbf{\Sigma}}_{11} &= \mathbf{\Delta}_1 \\
&= \mathbf{W}^{-1} + \mathbf{W}^{-1}\mathbf{X}\mathbf{\Delta}_2\mathbf{Y}\mathbf{W}^{-1} \\
&= \mathbf{\Sigma}_{11} - \mathbf{K}\mathbf{\Sigma}_{22}\mathbf{K}^\top + \mathbf{K}\check{\mathbf{\Sigma}}_{22}\mathbf{K}^\top \\
&= \mathbf{\Sigma}_{11} - \mathbf{K}\left(\mathbf{\Sigma}_{22} - \check{\mathbf{\Sigma}}_{22}\right)\mathbf{K}^\top. \qquad \text{(B.19)}
\end{aligned}$$

Equation (B.19) also has an intuitive explanation. A decrease in the uncertainty of the conditioned variables (i.e. $\mathbf{x}_1$) is proportional to a decrease in the uncertainty of the conditioning variables (i.e. $\mathbf{x}_2$).

Finally, we want to find the updated cross-covariance terms $\overset{\star}{\mathbf{\Sigma}}_{12} = \left(\overset{\star}{\mathbf{\Sigma}}_{21}\right)^\top$. From Equation (B.12) we get

$$\overset{\star}{\mathbf{\Sigma}}_{12} = -\mathbf{W}^{-1}\mathbf{X}\mathbf{\Delta}_2$$

which from the foregoing derivations can be readily expressed as

$$= \mathbf{K}\check{\mathbf{\Sigma}}_{22}. \tag{B.20}$$

To restate it in one place, the joint covariance after incorporating the updated marginal information is given by

$$\begin{aligned}\overset{\star}{\mathbf{\Sigma}} &= \begin{bmatrix} \overset{\star}{\mathbf{\Sigma}}_{11} & \overset{\star}{\mathbf{\Sigma}}_{12} \\ \overset{\star}{\mathbf{\Sigma}}_{21} & \overset{\star}{\mathbf{\Sigma}}_{22} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{\Sigma}_{11} - \mathbf{K}\left(\mathbf{\Sigma}_{22} - \check{\mathbf{\Sigma}}_{22}\right)\mathbf{K}^\top & \mathbf{K}\check{\mathbf{\Sigma}}_{22} \\ \hat{\mathbf{\Sigma}}_{22}\mathbf{K}^\top & \check{\mathbf{\Sigma}}_{22} \end{bmatrix}.\end{aligned} \tag{B.21}$$

These terms are all available when the optimized joint information matrix is formed, therefore the optimized covariance matrix can be calculated directly without inverting the information matrix. Finding the new joint mean follows similar steps which we omit here in the interest of brevity. The result is suggested by Equation (B.21):

$$\overset{\star}{\boldsymbol{\mu}} = \begin{bmatrix} \boldsymbol{\mu}_1 - \mathbf{K}\left(\boldsymbol{\mu}_2 - \check{\boldsymbol{\mu}}_2\right) \\ \check{\boldsymbol{\mu}}_2 \end{bmatrix}. \tag{B.22}$$

The content of this appendix is also available as a separate document at [86].