STATE ESTIMATION FOR AN UNDERWATER ROBOT USING VISUAL AND INERTIAL CUES

Florian Shkurti

School of Computer Science
McGill University, Montréal

October 2011

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of
Master of Science

Abstract

This thesis addresses the problem of 3D position and orientation (pose) estimation using measurements from a monocular camera and an inertial measurement unit (IMU). While the algorithmic formulation of the problem is generic enough to be applied to any intelligent agent that moves in 3D and possesses the sensor modalities mentioned above, our implementation of the solution is particularly targeted to robots operating in underwater environments. The algorithmic approach used in this work is based on statistical estimators, and in particular the extended Kalman filter (EKF) formulation, which combines measurements from the camera and the IMU into a unique position and orientation estimate, relative to the starting pose of the robot. Aside from estimating the relative 3D trajectory of the robot, the algorithm estimates the 3D structure of the environment. We present implementation trade-offs that affect estimation accuracy versus real-time operation of the system, and we also present an error analysis that describes how errors induced from any component of the system affect the remaining parts. To validate the approach we present extensive experimental results, both in simulation and in datasets of real-world underwater environments accompanied by ground truth, which confirm that this is a viable approach in terms of accuracy.

Résumé

Cette thèse aborde le problème d'estimation de la position et de l'orientation 3D (pose) en utilisant des mesures provenant d'une caméra monoculaire et d'une unité de mesure inertielle (IMU). Tandis que la formulation algorithmique de ce problème est suffisamment générique pour être appliquée aux tous les agents intelligents qui se déplacent en 3D et possèdent les mêmes capteurs mentionnés ci-dessus, notre implémentation s'addresse en particulier des robots fonctionnant dans des environnements sous-marins. L'approche algorithmique utilisée dans ce thèse est basée sur des estimateurs statistiques et en particulier le Extended Kalman Filter (EKF), qui combine les mesures provenant de la caméra et de l'IMU dans une estimation de position et d'orientation unique, relative à la pose de départ du robot. En plus de l'estimation de la trajectoire relative du robot en 3D, l'algorithme estime la structure 3D de l'environnement. Nous présentons des compromis d'implementation qui affectent la précision d'estimation en fonction de l'utilisation du système en temps réel, et nous présentons aussi une analyse des erreurs qui décrit comment les erreurs introduites par un composant du système affectent les parties restantes. Pour valider l'approche, nous présentons des nombreux résultats expérimentaux, tant en matière de simulation et de banque de données des environnements sous-marins accompagnée de réalité de terrain, qui confirme que cette approche est viable en termes de précision.

Acknowledgements

My immediate family, Tasim, Adriana and my awesome sister Vjosana have always supported and encouraged me. Without my parents' bravery to immigrate to Canada at an age when people seek stability and stillness, most of the opportunities my sister and I enjoy today would have been impossible.

Some of the people who have influenced me the most happened to be teachers and professors who loved what they were doing. To my supervisors at McGill, Gregory Dudek and Ioannis Rekleitis: I thank you both, for your advice, feedback, enriching discussions, your regular help with experiments, as well as your very helpful comments on this thesis. Likewise for Greg Wilson, who was my advisor at the University of Toronto, and taught me among other things the discipline required for coding properly. Leoni Dalla, who taught me Analysis, managed to replace my fear of math with fascination. Spyros Papadakis, Kostas Efkarpidis, Kostas Bougesis, Stelios Kokkos, and Dimitris Bizelis were by far the most thought-provoking and enthusiastic teachers I have had in my high-school years. Anastasios Mourikis and Stergios Roumeliotis also deserve many thanks for their continued patience in answering my questions on pose estimation.

Without my labmates the entire grad school experience would have been much poorer, both in terms of fun, and in terms of sharing technical knowledge. So, thank you (like a G6) Junaed Sattar, Bir Bikram Dey, Anqi Xu, Yogesh Ghirdhar, Philippe Giguére, Malika Meghjani, wise and powerful Chris Prahacs, Nicolas Plamondon, Yasmina Schoueri, Amjad Mahairi, Milena Scaccia, and last but certainly not least,

may-the-force-be-with-you-and-live-long-and-prosper Patrick Virie. My labmates, together with Rosemary Ferrie and Marinos Rekleitis assisted me in collecting the underwater datasets used in the experimental section, and provided helpful comments for this thesis.

Finally, to my friends outside the lab and especially my roommates: you are amazing people.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. Outline

Algorithms for estimating the position and orientation (pose) of a mobile robot are considered significant enablers of robot autonomy. This is mainly because an intelligent agent that knows exactly or approximately its 3D position in the world will find it easier to map its environment, and plan subsequent actions, compared to an agent that does not. Thus, a large spectrum of robotics research has focused on addressing the so-called localization problem, also known as the pose or state estimation problem. If, in addition to the robot's pose, the algorithm also estimates the 3D structure of the environment in which the robot is traveling, then the problem is called Simultaneous Localization And Mapping (SLAM) in the robotics literature or Structure From Motion (SFM) in the computer vision literature.

This thesis addresses the SLAM problem for the underwater domain, using visual and inertial measurements as sensory information. We are especially interested in real-time localization (regardless of whether it is absolute or relative to a starting pose), aiming to use the estimates as feedback to a high-level planner and controller that will enable the robot to execute complex trajectories for mapping and coverage. Having accurate localization underwater will also facilitate many other lines of research, such as multi-robot rendezvous [**62**] between underwater and surface robots, and coordination of heterogeneous multi-robot teams in general. In addition, we are

interested in algorithms that perform well in shallow (30 meters or less) underwater environments that are feature-rich and have good visibility. We prefer to use passive sensor modalities that are not disruptive to fragile marine ecosystems, rather than active sensing, which is one of the reasons we chose to pursue fusion of visual and inertial data for our state estimation system. To better situate our approach in the spectrum of existing research, we will mention representative existing algorithms that use a variety of different sensors to accomplish the same goal: estimate the trajectory of a robot, and the structure of the world around it in some fixed frame of reference, which is usually the initial pose of the robot.

1.1.1. Range-based localization. The most widely-known example we can begin with is the satellite-based Global Positioning System (GPS) [**71, 25**] [**84**, Ch. 20], pioneered by Parkinson, Getting, and Easton in the late 1970s. GPS satellites transmit timestamped messages that are interpreted by GPS receivers, which measure the time-of-flight from all such visible satellites, and translate those time measurements into distances to the corresponding satellites. Knowing the precise orbital position of each satellite, the receiver is able to trilaterate[1] its position in a fixed frame of reference with respect to the transmitters. The fixed frame in this case happens to be the World Geodetic System 1984 (WGS84). GPS measurements are a very effective data source for state estimation algorithms because they are easy to obtain. The system does however have some limitations:

(a) GPS signal reception is usually not possible in indoor environments. In fact, receivers need signals from at least 4 GPS satellites in order to perform trilateration, and the signal propagation is assumed to be along the line-of-sight between the transmitter and the receiver. Indoor environments and in many cases dense urban environments cause the signal to bounce (known as multipath), thus decreasing its integrity as a time-of-flight measurement.

[1]Trilateration is the computation of a position given distance measurements to known positions. In contrast, triangulation uses angle measurements for the same purpose.

(b) GPS signal is absorbed very quickly by water, so receivers do not work underwater.

(c) Even in places where the signal integrity is high, the error associated with the trilateration is in the order of 10 meters, which is a suboptimal data source for inferring the 3D orientation of the receiver [2].

The principles behind GPS are present in other range-based localization algorithms, which share the same limitations. For instance, in underwater environments acoustic positioning systems have been developed, where a receiver triangulates its approximate position with respect to a set of beacons placed at known distances on the surface of the sea [**95**]. In wifi-enabled indoor environments, wireless signal strength is indicative of the distance to a router, so a collection of such measurements provides a rough characteristic signature of rooms. Examples of this work are illustrated in [**28, 85, 51**].

Another group of range-based localization algorithms use sensory inputs from laser rangefinders or sonars, to obtain dense measurements of precise distances to the robot's surroundings. As the robot moves, the range signature from the previous timestep is compared to the one of the current timestep. Then these algorithms typically try to optimize for the rotation and translation that best explains the difference in the range signatures. This relative positioning technique has been termed "scan matching" and representative work includes [**58, 64, 14, 70**]. Assuming the presence of an underwater laser rangefinder or sonar, scan matching can be used as a complementary technique to the one explained in this thesis, since the range data can improve the accuracy of the camera constraints, or even impose additional constraints that are independent from the ones obtained by the camera measurements.

One of the differences between the fixed frame used by the GPS system and the fixed frame used in the system presented herein is that our localization is computed relative to the initial pose, so it is not an absolute and global position estimate,

[2]The same is not true for 2D orientation

unless the initial position is known explicitly. Another difference is that the system presented here can be used indoors and underwater, while GPS cannot.

1.1.2. Vision-based localization. Aside from range-based techniques, another category of localization algorithms are vision-based pose estimation methods, which have become quite popular in recent years, not only due to the wide availability of cheap camera sensors, but mainly due to a series of technical advances and successful demonstrations of structure from motion systems. These systems use camera sensors, sometimes monocular, but often stereo. They try to infer the motion of the robot from the image flow of regions of interest. Clearly, this is a relative positioning system, which estimates the robot's trajectory from the starting pose. Illustrative examples include early work by Davison et al. [**19**] and more recent work [**68**] on structure from motion in a constrained indoor space, which focuses primarily on augmented reality applications, and not necessarily on long-term, wide-area pose estimation. That said, there are examples of vision-based pose estimation over hundreds-of-meters-long trajectories, such as in the work of Nistér et al. [**69**], Konolige et al. [**49**], Barfoot [**9**], and Sibley et al. [**83**]. We will omit discussing 3D pose estimation from artificial fiducial markers placed in the environment, i.e. we assume underwater environments that do not have human-generated structures. It is worth mentioning that in this particular line of research there exist two major ideas, which are sometimes used in conjunction: one approach advocates using statistical filters (e.g. extended Kalman filters, unscented Kalman filters, or particle filters) to incorporate the latest measurement into the current estimate, aiming for real-time operation; the other approach advocates using a history of measurements in order to achieve a more globally accurate pose estimate (e.g. bundle adjustment methods [**93, 83**]). A detailed analysis of this debate, highlighting the advantages of optimization methods over filtering is presented in [**87**].

Vision-aided localization techniques also include appearance-based localization. The seminal work of Cummins and Newman [**18**], which performed place recognition over a thousand-kilometer-long trajectory, exemplifies this line of research. The main

difference between appearance-based localization methods and the work presented here is that they do not attempt to estimate the pose of the robot, but rather to identify the place or area at which the robot is located.

1.2. Objectives

In this thesis, we assume that the robot moves freely in 3D space, and that it has at least two sensors at its disposal: a monocular camera and an inertial measurement unit (IMU). An IMU measures 3D rotational velocity and linear acceleration, in other words, the first derivative of the orientation and the second derivative of position. Our aim is the design of a robust and real-time 3D pose estimation algorithm,

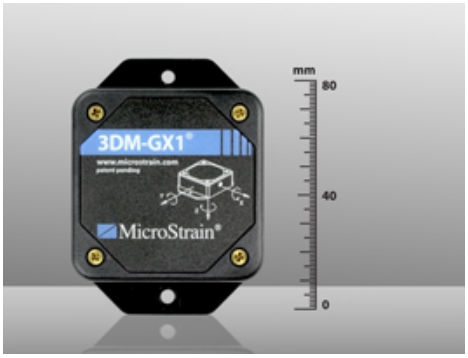


FIGURE 1.1. The particular IMU sensor that we are using, the MicroStrain 3DM-GX1, incorporates a MEMS (microelectromechanical system) accelerometer, gyroscope, and magnetometer.

which also estimates the 3D structure of the surrounding environment wherein the robot is moving. In other words, a real-time structure from motion algorithm that is aided by IMU measurements. The target robotic platform is the Aqua family of amphibious robots [**24, 75**]. These hexapod robots are equipped with three IEEE-1394 IIDC cameras, a low-cost IMU, and a pressure sensor which is used to provide noisy measurements of depth, measured from the surface of the water. The robot's swimming motion is a product of six paddles, which oscillate synchronously in two groups of three. If we naively integrate these IMU measurements over long time intervals, as we will see in later chapters, the 3D pose estimate diverges very quickly, due to the

FIGURE 1.2. The second generation of the Aqua family of underwater robots.

integration of the noise associated with these measurements. The main idea behind the algorithm implemented in this thesis is that the flow of points of interest tracked in consecutive images will impose motion constraints on the integration of the IMU measurements, which will prevent the divergence of the pose estimate.

The majority of the localization work mentioned previously deals with neither the underwater domain nor inertial measurements per se, and often assumes the existence of a mathematical motion model of the vehicle at hand. In our case, such a model is nonlinear, hard to justify, and susceptible to deviations due to currents and other factors [**31, 16, 72**]. In general, underwater environments are generally more challenging than most of their indoor counterparts for the following reasons:

(a) *They are prone to rapid changes in lighting conditions.* The canonical example that illustrates this is the presence of caustic patterns, which are due to refraction, non-uniform reflection, and penetration of light when it transitions from air to the rippling surface of the water [**98, 52**].

(b) *They often have limited visibility.* This is due to many factors, some of which include light scattering from suspended plankton and other matter, which

cause blurring and "snow effects". Another reason involves the incident angle at which light rays hit the surface of the water. Smaller angles lead to less visibility.

(c) *They impose loss of contrast and colour information with depth.* As light travels at increasing depths, different parts of its spectrum are absorbed. Red is the first color that is seen as black, and eventually orange, yellow, green and blue follow. This absorption sequence refers to clear water, and is not necessarily true for other types [**98**]. That said, this particular constraint will have a limited impact in our case because we are not using color information, only greyscale images.

Despite all the disadvantages mentioned above, visual cues are informative and relevant in the underwater domain, especially due to the passive nature of visual sensing, which is not disturbing to marine life and surrounding ecosystems, but also considering the fact that this work is targeting shallow (30 meters or less) underwater environments. IMU information is also passive and provides useful information about the vehicle's motion. Given the above constraints, pose estimation algorithms that rely on IMU and visual data are promising, because the IMU provides a noisy estimate of the vehicle's dynamics even in the presence of strong currents, while the camera imposes corrective motion constraints on the drifting IMU pose estimate, which is obtained by integration.

1.3. Contributions

In this thesis we extend the approach presented by Mourikis and Roumeliotis [**65**], and we present its evaluation in the underwater domain. Specifically, we add measurements from a depth sensor in the original algorithm, and we test the implementation both on simulated data and on real underwater datasets with approximate ground truth. We also evaluate and select feature tracking parameters. Finally, we examine how errors in particular components of the system affect the estimates, and what is the level of errors that should be expected given the IMU noise characteristics and the camera noise characteristics.

CHAPTER 2

BACKGROUND

2.1. Basic concepts

In this chapter we will introduce some of the basic terminology and mathematical tools and concepts that are required in order to understand the arguments and algorithms presented in the following chapters. Most existing work on visual and inertial pose estimation algorithms relies on a standard set of statistical tools and paradigms and on common computer vision techniques in order to merge the two sources of information – camera and IMU – into one coherent estimate. Conceptually, we are given two streams of information that need to be processed. IMU measurements come much more frequently than camera measurements (in the order of 2 to 5 times), as cameras typically capture images at 15-30Hz, while IMU measurements are often available at 30-100Hz. On the IMU side we get motion information in the form of 3D angular velocities and linear accelerations. On the vision side, we need to extract motion and spatial information from the incoming images. The way we do this is by tracking keypoints[1] (or other features, such as lines) of interest from one image to the next. The motion of these keypoints on consecutive images contains partial information about the 3D motion of the camera relative to the observed keypoints

[1]Since our current work deals with points (and not curves) of interest we are going to use the terms *keypoint* and *feature* interchangeably.

and objects in the world (partial because the distance between the camera and the 3D points of interest is not directly available).

2.1.1. Feature tracking. The feature tracking component of the system can be instantiated as one of many alternatives, such as SIFT [**57**], SURF [**11**], Shi-Tomasi [**77**], or Lucas-Kanade optical flow [**4**], just to name a few. The principle behind most of these alternatives is that reliable keypoints for tracking are the ones that are salient with respect to some objective function. This function usually favours corners, that is, points around which there is significant intensity variation in two major directions. The motivation behind this is that, ideally, salient points in one image will remain so in the next image, which will open up the possibility of matching identical points in consecutive images. The technical details of this process will be described in subsequent chapters, but at the moment it suffices to say that the main motivation behind using feature matching to track multiple points from one image to the next is in order to impose motion constraints on the consecutive camera frames. This means that the poses from which the camera observed each point are constrained to some specific trajectory, which in turn implies (since the camera is fixed on the robot) that the robot's motion is constrained. In general these constraints do not determine a unique trajectory, unless we have access to the exact relative motion performed from one camera pose to the next, an estimate of which is what the IMU measurements give us.

2.1.2. Kalman Filters. Fusing two (or more) sources of information is usually done via statistical estimators, the most widely used in this domain being the Kalman Filter (KF). In the KF framework we define a vector of random variables of interest (for instance the position, orientation and velocity of the robot) which we denote by $\mathbf{x}_t$ where t represents time. The goal of the filter is to generate a probabilistic estimate of $\mathbf{x}_t$, given the sensor measurements, which comprise a vector of observed data denoted by $\mathbf{z}_t$. Let $\hat{\mathbf{x}}_{t|t}$ denote an estimate of the state $\mathbf{x}_t$ given all sensor measurements up to and including time t. The Kalman Filter state estimate,

denoted by $\hat{\mathbf{x}}_{t|t}^{MMSE}$, is such that it minimizes the mean square error:

$$\hat{\mathbf{x}}_{t|t}^{MMSE} = \underset{\mathbf{x}_t}{\operatorname{argmin}}\ E[(\mathbf{x}_t - \hat{\mathbf{x}}_t)^T(\mathbf{x}_t - \hat{\mathbf{x}}_t) \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t] \tag{2.1}$$

From this definition we can show that the minimum mean square error estimate can be computed as the conditional expectation of the state given the sensor measurements:

$$\hat{\mathbf{x}}_{t|t}^{MMSE} = E[\mathbf{x}_t \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t] \tag{2.2}$$

Therefore, computing this estimate requires knowledge (or assumption) of an underlying conditional probability distribution $p(\mathbf{x}_t \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t)$ which encodes the degree of belief of being in state $\mathbf{x}_t$ given all the sensory inputs up to time t. In order to make the computation of the estimate mentioned above more tractable the Kalman Filter makes some simplifying assumptions:

(i) (Markov assumption) The state $\mathbf{x}_t$ depends only on the state $\mathbf{x}_{t-1}$

(ii) The measurement $\mathbf{z}_t$ depends only on the state $\mathbf{x}_t$ and not on the previous measurements $\mathbf{z}_1, ..., \mathbf{z}_{t-1}$

These assumptions have a critical effect on reducing the complexity of computing $\hat{\mathbf{x}}_{t|t}^{MMSE}$ because they enable us to compute the conditional probability distribution mentioned in a recursive manner, using Bayes' rule, as follows [2]:

$$p(\mathbf{x}_t \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t) = \underbrace{p(\mathbf{z}_t|\mathbf{x}_t)}_{\text{sensor model}} \int \underbrace{p(\mathbf{x}_t \mid \mathbf{x}_{t-1})}_{\text{state transition model}} \ p(\mathbf{x}_{t-1} \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{t-1}) d\mathbf{x}_{t-1} \tag{2.3}$$

The sensor model describes how the next sensor measurements depend on the current state, while the state transition model describes the transition probabilities from the set of previous possible states to the set of current possible states. At this point the Kalman filter makes more assumptions:

(iii) (Gaussian assumption) The state $\mathbf{x}_t$ given the measurements $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t$, the sensor model, and the state transition model are normally distributed.

[2] This makes the Kalman Filter an instance of a recursive Bayesian filter.

Mathematically, this is expressed as:

$$\mathbf{x}_t|\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{t|t}, \mathbf{P}_{t|t}) \tag{2.4}$$

$$\mathbf{x}_t|\mathbf{x}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t) \tag{2.5}$$

$$\mathbf{z}_t|\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t) \tag{2.6}$$

where $\mathcal{N}(\boldsymbol{\mu}_{t|t}, \mathbf{P}_{t|t})$ denotes the multivariate normal distribution with mean $\boldsymbol{\mu}_{t|t}$ and covariance matrix $\mathbf{P}_{t|t}$. This means that two parameters are sufficient statistics for the complete description of the conditional distribution $p(\mathbf{x}_t \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t)$, and thus the Kalman Filter need only compute two estimates, one for the mean and the other for the covariance:

$$\hat{\mathbf{x}}_{t|t}^{MMSE} = E[\mathbf{x}_t \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t] = \boldsymbol{\mu}_{t|t} \tag{2.7}$$

$$\hat{\mathbf{P}}_{t|t}^{MMSE} = E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}^{MMSE})(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}^{MMSE})^T \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t] = \mathbf{P}_{t|t} \tag{2.8}$$

This assumption implies a unimodal distribution, in other words, that there is a single best guess about the estimate of the state, which is often not true, as in reality there may be two different states that are equally plausible given the data.[3]

(iv) (Linearity assumption) The current state $\mathbf{x}_t$ is a linear function of the previous state $\mathbf{x}_{t-1}$, and the current measurement $\mathbf{z}_t$ is a linear function of the state $\mathbf{x}_t$. In both cases, the functions are affected by uncorrelated Gaussian noise processes. More formally,

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{w}_t \quad \text{where} \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t) \tag{2.9}$$

$$\mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t \quad \text{where} \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t) \tag{2.10}$$

where $\mathbf{w}_t$ is independent of $\mathbf{w}_{t-1}, \mathbf{w}_{t-2}, ..., \mathbf{w}_0$ and $\mathbf{v}_t$ is independent of $\mathbf{v}_{t-1}, \mathbf{v}_{t-2}, ..., \mathbf{v}_0$. The noise processes $\mathbf{w}_t$ and $\mathbf{v}_t$ model uncertainty caused

[3]This is one of the differences between Kalman Filters and Particle Filters, which are another class of Bayesian recursive statistical estimators.

by factors such as: our inability to exhaustively describe the physical system that governs the dynamics of $\mathbf{x}_t$ and $\mathbf{z}_t$, inaccurate sensor measurements, or small quantities that we simply opt to ignore. Similar concepts are seen in Taylor approximations of functions, for instance in the basic Newtonian equations of motion. The matrices $\mathbf{F}_t$ and $\mathbf{H}_t$ are deterministic; they encode our knowledge of how the state evolves and how we expect the measurements to depend on the state. It is also worth mentioning for completeness' sake that the linear state transition models for the Kalman filter might also include a control vector $\mathbf{u}_t$, which makes the state model:

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t \tag{2.11}$$

In our case however, we are ommitting this term because in this work we are not modeling the direct controls of the motion of the robot.

The recursive algorithm that computes the estimate of the Kalman Filter can be described in two steps: the propagation (or prediction) and the update (or correction) step. In the propagation step the filter predicts what the next state will be according to its linear state transition model, having access only to measurements and state information up until the previous time unit. For ease of notation we will denote $\hat{\mathbf{x}}_{t|t}^{MMSE}$ by $\hat{\mathbf{x}}_{t|t}$ and $\hat{\mathbf{P}}_{t|t}^{MMSE}$ by $\hat{\mathbf{P}}_{t|t}$.

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1} \tag{2.12}$$

$$\hat{\mathbf{P}}_{t|t-1} = \mathbf{F}_t\hat{\mathbf{P}}_{t-1|t-1}\mathbf{F}_t^T + \mathbf{Q}_t \tag{2.13}$$

Following the propagation step the filter has access to the current measurements and it performs a correction of the estimated state and covariance, based on the residual

$\mathbf{r}_t$ between the expected measurement and the actual measurement:

$$\mathbf{r}_t = \mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1} \tag{2.14}$$

$$\mathbf{S}_t = \mathbf{H}_t\hat{\mathbf{P}}_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t \tag{2.15}$$

$$\mathbf{K}_t = \hat{\mathbf{P}}_{t|t-1}\mathbf{H}_t^T\mathbf{S}^{-1} \tag{2.16}$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\mathbf{r}_t \tag{2.17}$$

$$\hat{\mathbf{P}}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\hat{\mathbf{P}}_{t|t-1} \tag{2.18}$$

$\mathbf{S}_t$ is the covariance matrix of the residual. Intuitively, if this matrix is 'large' by some measure then we should not trust the residual that the current measurement is suggesting, but if it is 'small' then we should probably trust it. $\mathbf{K}_t$ is called the 'optimal Kalman gain' and it is computed in such a way that it minimizes the trace of the covariance matrix $\hat{\mathbf{P}}_{t|t}$. This is not an arbitrary criterion, as it is equivalent to minimizing the mean square error in Eq. (2.1).

As a final remark about the Kalman Filter we note that the recursive algorithm mentioned above gives rise to an unbiased and consistent estimator, as long as the initial conditions $\hat{\mathbf{x}}_{0|0}$ and $\hat{\mathbf{P}}_{0|0}$ are selected accurately. Mathematically, this means that Eq. 2.7 and 2.8 are preserved during each step of the algorithm.

By definition, if the linearity, Markov and Gaussian assumptions truly describe the dynamics of the state of interest, then the Kalman Filter is the optimal estimator with respect to the minimum mean square error criterion. There are examples however of dynamical systems that have nonlinear state transition and sensor models. As we will show in later chapters, including orientation in the state vector leads to a nonlinear transition model. Similarly, we will see that the sensor model for the camera measurements is also nonlinear. In these cases the sensor and state models are described by:

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}) + \mathbf{w}_t \tag{2.19}$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t \tag{2.20}$$

where **f** and **h** are nonlinear functions. To make the estimation problem tractable in these cases, the two functions are linearized and the equations of the Kalman Filter are used, this time without any guarantee of optimality, or consistency. This is the approach of the Extended Kalman Filter (EKF), which we are using in this thesis.

2.1.3. What does an IMU measure and in what frame of reference? We mentioned earlier that an IMU measures angular velocities and linear accelerations. A question that comes to mind then is: in what coordinate frame are they measured? An IMU consists of a 3-axis accelerometer and a 3-axis gyroscope. An accelerometer measures *g-force*, which is a misnomer for accelerations relative to a free-falling frame. A single-axis accelerometer placed on the surface of the Earth will measure $1g$ upwards [4], which, from the point of view of the resting device, is caused by the force exercised from the Earth's surface to the resting device. A free-falling accelerometer in a vacuum would measure $0g$.

A more intuitive way to think about this is by imagining the accelerometer as a box with a small metal ball inside it (see Fig. 2.1). Whenever the ball touches

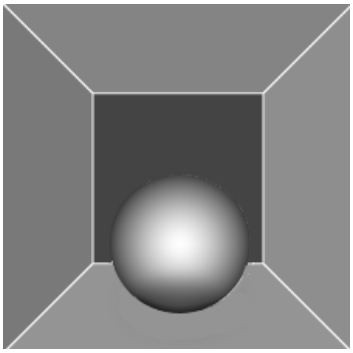


FIGURE 2.1. When the box rests on the ground the ball will rest on its floor, thus the accelerometer will measure 1g upwards.

the walls of the box, the accelerometer outputs a signal proportional to the force exerted to the ball from the walls that touch it. For instance, if the ball-box system is free-falling in vacuum, the ball will not touch the walls of the box. From the point of view of a non-accelerating external observer, the box-ball system has $1g$ acceleration

[4] Assuming an idealized accelerometer this depends on the position on the surface of the Earth

downwards. From the point of view of the box the ball has $0g$ acceleration. In other words, the coordinate frame in which the accelerometer measurements are expressed in is the box itself, i.e. the enclosing case of the accelerometer. To further illustrate this, if the box-ball system lies still on the surface of the Earth, an external observer sees that the system has $0g$ acceleration, however, from the point of view of the box, the ball is subject to a force from the walls of the box. The accelerometer will output $1g$ pointing upwards in that case. Accelerometers do not measure acceleration with respect to an external observer's frame, or a global frame of reference such as the one we are going to use in our calculations. Thus, we will need to make this conversion ourselves.

Finally, the measurements of the gyroscope are easier to interpret: they are the angular velocities measured with respect to the axes of the aforementioned box. From now on, we will refer to this box as *the IMU frame* $\{I\}$, and to the external observer's frame as *the global frame* $\{G\}$, which will be non-accelerating, and more specifically, static, as in the examples above.

2.2. Related work

Previous work in the domain of visual and inertial state estimation has been evaluated mainly on road vehicles, such as cars and terrestrial mobile robots, as well as on aerial vehicles, such as quadrotors and small-scale helicopters. Aside from the algorithm presented by Mourikis and Roumeliotis [**65**], which will be described in full detail in the following chapter, there are a number of other algorithmic approaches that use both vision and inertial data that have been experimentally validated in the field over large-scale trajectories (kilometers or hundreds-of-meters long). On the other hand, there is a large number of approaches which have only been demonstrated to work reliably for small-scale pose estimation (room-size environments). In the former category falls the work of Jones and Soatto [**42**], Kelly and Sukhatme [**46**], and Eustice et al. [**27**] while in the latter category there are a large number of examples, which will be mentioned below.

2.2.1. Large-scale estimation. Jones and Soatto [**42**] demonstrated that their algorithm yields less than 1% localization error over a 31km-long trajectory in an urban environment. Their sensors included a monocular camera recording images at 30Hz and a BEI Systems IMU with update rate at 100Hz, both of them mounted on a sensor pack that was placed on a car. They used an Extended Kalman Filter (EKF) to combine the measurements from the two sensors, however, the implementation of the EKF was modified to perform online sample-consensus-based outlier rejection of both the IMU and the camera measurements [**94**]. Initially, the algorithm performs an autocalibration step, during which the IMU-camera sensor pack is subjected to 'rich' motion (with many degrees of freedom and in many directions) which allows the gravity vector and IMU-camera transformation to be estimated. After those two parameters have converged, the normal operation of the algorithm begins, where features are tracked in the scene, their depth continuously re-estimated by the filter. Both camera and IMU inlier measurements are used during updates, while predictions are done by simple time propagation of the motion model forward in time. On the purely vision side of their system, the algorithm relies on feature tracking from optical flow. Finally, they report that their system is able to process measurements in real-time on a desktop computer.

Kelly and Sukhatme [**46**] presented an algorithm that was shown to yield under 1% localization error over a 400m-long trajectory traversed by a small-scale helicopter, where ground truth was provided by GPS measurements. The input sensors of the vehicle were a stereo camera recording at 30Hz and an Inertial Science ISIS IMU recording at 100Hz. Their algorithm uses an Extended Kalman Filter to combine IMU measurements and visual odometry estimates computed from consecutive stereo images. During the prediction step the pose estimate is propagated using the IMU measurements, while the update step is driven by the visual odometry estimate. Feature tracking in consecutive stereo pairs was done using the Lucas Kanade Tracker [**5**], and the authors reported offline processing of the results, but not real-time operation on the Mini-ITX embedded vision computer onboard the helicopter. Their

work demonstrated that visual odometry combined with IMU measurements produces significantly better results compared to visual odometry alone.

Eustice et. al [**27**] showed successful use of the sparse information filter to combine visual and inertial data, while performing a robotic survey mapping of length $3.4km$ of the surface of the RMS Titanic. The total surface covered was approximately $3100m^2$. They also presented a technique for maintaining the consistency bounds of the filter's covariance. In their case, mapping and localization was done offline on collected datasets and the possibility of real-time operation of their system was not addressed.

2.2.2. Small-scale estimation. The work of Strelow and Singh [**88**] presents a batch pose estimation method whose input is a sequence of time-synchronized consecutive images and IMU measurements. The estimated pose output by their algorithm minimizes the reprojection error of the observed features, without deviating much from the IMU motion measurements. In other words, their work extends the visual bundle adjustment formulation, by imposing IMU motion measurements in the estimated trajectory. Close in spirit to our work is the work of Williams et al. [**97**], who extended the work of Mourikis and Roumeliotis [**65**] by tracking persistent lines, and not just keypoints, from the camera of a quadrotor trying to enter buildings. From an implementational standpoint, worthy of note is also the paper of Hertzberg et al. [**35**] which describes the authors' experiences of building a visual and inertial SLAM system using currently available open-source components. Corke [**17**] used IMU measurements to form an expected image motion field, and used its residual with respect to the image motion field produced by the flow of features in order to apply a correction to the pose estimate. Huster and Rock [**40**] fused inertial and visual measurements in order to localize a moving robotic arm with respect to a stationary object, aiming to facilitate the grasping process, particularly for underwater vehicles equipped with grippers. In their work, they argue that fusing the measurements via an Extended Kalman Filter is not a reliable technique, since the linearization of the process model is going to lead to underestimating the uncertainty, which may cause

the filter to diverge. As an alternative, they propose a coordinate change on the state vector that makes the linearizations unnecessary. In the same vein, Gemeiner et al. [**30**], as well as Langelaan and Rock [**53**] used the Unscented Kalman Filter (UKF) to fuse the measurements from the two sensors, in order to avoid said linearization. In this work we chose the EKF over the UKF mainly for reasons of computational efficiency, but future extensions of this work will consider the use of the UKF, as it has been shown to often outperform the EKF in terms of accuracy [**96**]. Rehbinder and Ghosh [**73**] used line correspondences from a monocular camera and combined them with inertial measurements in order to estimate the orientation, but not the position, of the camera. Aside from not estimating position, this technique cannot be applied underwater due to the general lack of man made structures that give rise to straight lines. Alenyà et al. [**2**] proposed contour tracking instead of feature tracking in consecutive images, which is something that our work could potentially benefit from. Diel et al. [**22**] used the residual of the epipolar constraints between consecutive camera frames as a residual for an Extended Kalman Filter that corrects the propagated IMU estimates. Finally, a very interesting example of small-scale visual and inertial estimation was proposed by Grimm and Grigat [**33**], who placed the IMU and camera sensor on a pen, aiming to use it for small-scale image mosaicing and handwriting recognition.

STATE ESTIMATION

2.3. Preliminaries

2.3.1. Rotation representation. In our attempt to estimate the orientation and the position of a robotic vehicle relative to its initial pose, we are going to need to choose a way to represent rotations and translations, to describe rigid motion. For translations in Euclidean space this is easy, we can represent them as vectors. This will not work for rotations, as they don't commute. For rotations there are quite a few choices of parametrizations: rotation matrices, Euler angles, quaternions, Rodrigues parameters, Cayley-Klein parameters, and axis-angle. The special-orthogonal group of rotation matrices,

$$SO(3) = \{R \quad | \quad RR^T = I \quad detR = 1\} \tag{2.21}$$

has dimension 3, so the minimal number of parameters needed to describe a rotation is 3, but the representations mentioned above use up to 9. In Table 2.1 we present an overview of these alternatives, but a more detailed analysis can be found in [**76, 21, 26, 32, 79**]. Essentially, the main differences among the available rotation representations have to do with some of the following considerations:

- *The number of parameters.* Fewer parameters are appealing because the risk of numerical instabilities is lower. In the case of rotation matrices for example, one would need to enforce the orthonormality of the estimated rotation matrix after each major numerical operation, which is suboptimal.

- *Whether they have singularities or not.* Hopf and Stuelpnagel [**36, 89**] showed that it is impossible to represent arbitrary rotations using 3 numbers without any singularities. A specific example of this theorem is the representation of 3D rotations by Euler angles, which are susceptible to *gimbal lock* when the source and target frames have a collinear rotation vector. In fact, Hopf showed that at least 5 real numbers are needed to represent rotations without any singularities in a 1-1 and onto fashion.
- *The number of floating point operations required to concatenate two rotations.* For the axis-angle representation this number is particularly high because no direct formulas are currently known that can compose two such rotations. Instead, they are converted into one of the other representations. So, axis-angle, while appealing and intuitive to common logic, is not an efficient representation. Quaternions, on the other hand can be composed much more efficiently.
- *The number of floating point operations required to transform to other representations, if necessary.*

Clearly, the last two considerations become irrelevant if enough computing power is available, however they do become noteworthy issues when using embedded computers with limited-computing capabilities. Quaternions seem to satisfy most of these requirements, so they have become a standard for representing global orientations with respect to some fixed frame of reference. We are also going to be relying on them in this thesis:

$$\mathbf{q} = [\mathbf{u}sin(\theta/2) \quad cos(\theta/2)] = [q_1 \quad q_2 \quad q_3 \quad q_4] \tag{2.22}$$

In the context of Kalman filtering, the fact that unit quaternions are used in the representation of rotations means that the propagation and update steps will have to respect the unit norm constraint. The typical formulation of the Kalman filter, however, only addresses the solution of *unconstrained* stochastic systems. This implies that we have to: (a) modify the formulation of the Kalman filter so that the state

TABLE 2.1. Rotation representations

Name	Definition	Parameters	Singularities
Rotation matrix	$R \in \mathbb{R}^{3\times3}$ such that $R^T = R^{-1}$ and $detR = 1$	9	
Euler angles	(θ, ϕ, ψ) such that $R = R_x(\theta)R_y(\phi)R_z(\psi)$	3	If $\phi = n\pi, n \in \mathbb{Z} - \{0\}$ then θ and ψ refer to the same axis and 1DOF is lost (gimbal lock).
Quaternions	$\mathbf{q} = [\mathbf{u}sin(\theta/2) \quad cos(\theta/2)]$ rotation along the unit axis $\mathbf{u}$ by θ	4	
Classical Rodrigues Parameters	$\mathbf{s} = \mathbf{u}tan(\theta/2)$ rotation along the unit axis $\mathbf{u}$ by θ	3	If $\theta = n\pi, n \in \mathbb{Z} - \{0\}$ it is undefined.
Modified Rodrigues Parameters	$\mathbf{m} = \mathbf{u}tan(\theta/4)$ rotation along the unit axis $\mathbf{u}$ by θ	3	If $\theta = 2n\pi, n \in \mathbb{Z} - \{0\}$ it is undefined.
Cayley-Klein Parameters	$\mathbf{p} = [\alpha \quad \beta \quad \gamma \quad \delta]$ $\alpha = cos(\theta/2) + iu_z sin(\theta/2)$ $\beta = -(u_y - iu_x)sin(\theta/2)$ $\gamma = (u_y + iu_x)sin(\theta/2)$ $\delta = cos(\theta/2) - iu_z sin(\theta/2)$	4	
Axis-angle	(u_x, u_y, θ)	3	If $\theta = n\pi, n \in \mathbb{Z} - \{0\}$ it is singular.

vector is subject to a constraint, while maintaining the optimality of the estimate for linear systems, or (b) re-normalize the quaternions in the state vector after each propagation and update step, or (c) simply ignore the preservation of unit norm of quaternions during the two steps of the Kalman filter.

The first option was only recently proven to be a feasible approach [**99, 43, 86**]. The third was shown to introduce estimation errors that are difficult to correct [**80, 81**]. The second option has been the ad-hoc method of choice for at least four decades, and its effects on the Kalman filter have been studied in many works [**6, 8, 20, 82**]. In this thesis we are also going to follow (b), so we are going to normalize quaternions after propagations and updates.

Finally, it is worth mentioning that there exist transformations that convert from one representation to the other. For example, the Rodrigues formula will convert axis-angle representations to rotation matrices. Another example is the Cayley transform[5]:

$$R = (I - \lfloor \mathbf{s} \times \rfloor)^{-1}(I + \lfloor \mathbf{s} \times \rfloor) \tag{2.23}$$

which converts Classical Rodrigues Parameters to a rotation matrix. Its appeal comes into play in scenarios where we want to search over the space of rotations in the context of optimization problems. If our search is done over the rotation matrices, then we will need to use constrained optimization to impose orthonormality and $+1$ determinant, while if our search is done over vectors in $\mathbb{R}^3$ the Cayley transform will allow us to use unconstrained optimization.

2.3.2. Error quaternion. If $\hat{\mathbf{q}}$ is our estimate of the quaternion that represents the orientation of the robot, and $\mathbf{q}$ is the real orientation, then their difference, i.e. the error quaternion of unit length will be denoted by $\tilde{\mathbf{q}}$, where:

$$\mathbf{q} = \tilde{\mathbf{q}} \otimes \hat{\mathbf{q}} \tag{2.24}$$

and $\otimes$ stands for quaternion multiplication. If the error quaternion is close to identity then the error in the estimated orientation is small, which is good. On the other hand, an error quaternion that is sufficiently far from identity indicates a poor estimate, which we want to avoid. Notice that we haven't formally defined a distance over the space of unit quaternions. Since unit quaternions are points on the unit 4-sphere, we have a couple of choices when measuring distance between two such points: one is the Euclidean (straight line) distance,

$$d_E(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\| = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2 + (p_4 - q_4)^2} \tag{2.25}$$

and the other will be the smallest arc-length joining the two points on the sphere,

$$d_A(\mathbf{p}, \mathbf{q}) = arccos(\mathbf{p} \cdot \mathbf{q}) \tag{2.26}$$

[5] $\lfloor \mathbf{s} \times \rfloor$ denotes the cross-product matrix generated from the 3-vector $\mathbf{s}$

where we are treating $\mathbf{p}$ and $\mathbf{q}$ as 4-vectors of unit length, whose origin is at the center of the sphere. Clearly, the straight line distance will be smaller or equal to the arc-length distance, because the former follows a path in the interior of the sphere, as opposed to the latter, whose path is on the surface. At this point it is worth mentioning for completeness' sake that the multiplicative quaternion correction is not the only option that has been taken into consideration in the literature. Additive correction has also been studied, for instance in [**7, 59, 8**].

In the context of Kalman filtering, if we continue to regard quaternions as points or vectors on the unit 4-sphere, the uncertainty of our estimate $\hat{\mathbf{q}}$ of the random vector $\mathbf{q}$ will be expressed as the covariance matrix

$$cov(\tilde{\mathbf{q}}) = E\{\tilde{\mathbf{q}}^T\tilde{\mathbf{q}}\} - E\{\tilde{\mathbf{q}}\}^T E\{\tilde{\mathbf{q}}\} \tag{2.27}$$

where ^{T} is the vector transpose operation in this vectorial view of quaternions. There is a subtle, unsettled as of yet, potential problem with viewing quaternions as vectors and computing their covariance as in Eq. (2.27): the 4×4 covariance matrix will not have full rank because of the imposed unit-length constraint. Lefferts et. al [**54**] argue that this rank deficiency is hard to preserve numerically during the iterations of the Kalman filter, and that these numerical errors might make the covariance non-symmetric. To avoid this they suggested that the error quaternion be linearized by a 3-vector, so the 3×3 unconstrained covariance of said vector be used instead of the exact, 4×4 matrix. On the other hand, Bar-Itzhack et al. [**8**] argue that in practice this phenomenon does not occur when using additive quaternion correction, however their argument did not provide an analysis for the multiplicative correction, which we are using. There are a number of options available at this point for linearizing the error quaternion as a 3-vector [6]:

i) The so-called small angle approximation, whereby:

$$\tilde{\mathbf{q}} = \begin{bmatrix} \tilde{\mathbf{u}} sin(\tilde{\theta}/2) & cos(\tilde{\theta}/2) \end{bmatrix} \simeq \begin{bmatrix} \tilde{\mathbf{u}}\tilde{\theta}/2 & 1 \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{\theta}}/2 & 1 \end{bmatrix} \tag{2.28}$$

[6]The above list of options is not exhaustive (for instance, see [**10**]).

ii) Exact representation by classical Rodrigues parameters

$$\tilde{\mathbf{q}} = crp2quat(\tilde{\mathbf{u}}tan(\tilde{\theta}/2)) \tag{2.29}$$

which is undefined for $\tilde{\theta} = n\pi, n \in \mathbb{Z} - \{0\}$. The transformations $crp2quat$ and $quat2crp$ are shown in the Appendix section A.1.

iii) Exact representation by modified Rodrigues parameters

$$\tilde{\mathbf{q}} = mrp2quat(\tilde{\mathbf{u}}tan(\tilde{\theta}/4)) \tag{2.30}$$

which is undefined for $\tilde{\theta} = 2n\pi, n \in \mathbb{Z} - \{0\}$. Again, the transformations $mrp2quat$ and $quat2mrp$ are shown in the Appendix section A.2.

We argued earlier that quaternions were a better, singularity-free representation of rotations compared to the remaining options, including Rodrigues parameters. The reason we are suggesting representations that have singularities in the case of error quaternions is that we assume that $\tilde{\theta}$ is going to be small. In other words, we can combine a stable, unit quaternion representation for the global orientation, with locally stable, 3-vector representations for small corrections to the global orientation. In this thesis we are going to use the small-angle approximation to represent error quaternions, so the covariance in Eq. 2.27 will be approximated by:

$$cov(\tilde{\mathbf{q}}) \simeq cov(\tilde{\boldsymbol{\theta}}) = E\{\tilde{\boldsymbol{\theta}}^T\tilde{\boldsymbol{\theta}}\} - E\{\tilde{\boldsymbol{\theta}}\}^T E\{\tilde{\boldsymbol{\theta}}\} \tag{2.31}$$

2.3.3. Camera-IMU transformation. As mentioned previously, the camera frame $\{C\}$ and the IMU frame $\{I\}$ are fixed with respect to each other, both moving in unison on the rigid body of the robot with respect to the global reference frame $\{G\}$. The fixed transformation between the camera and the IMU consists, therefore, of a rotation and a translation. Let ${}^C_I\mathbf{q}$ denote the quaternion that rotates from the IMU to the camera frame, and ${}^I\mathbf{p}_C$ denote the origin of the camera frame in IMU coordinates. This transformation is a very important parameter of the estimator, and if misestimated, the reconstruction of the robot's 3D trajectory will be affected by systematic errors. That said, the effects of the quaternion are much higher

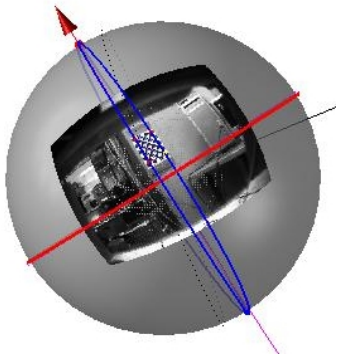


FIGURE 2.2. Visualisation of the final output from InerVis, showing the estimated vanishing points from the chessboard (blue), oriented properly using the IMU. It is assumed that the chessboard is placed vertically.

than the effects of an error in the translation. That is why, aside from trusting the information from the design specifications of the robot (e.g. CAD files), algorithms have been developed for the joint calibration of a camera-IMU setup. We are using the InerVis system, developed by Lobo and Dias [**55, 56**], which requires static images of a calibration chessboard together with IMU readings around the time the images were taken.

Its output is ${}^{C}_{I}\mathbf{q}$. In our case the difference between the estimated quaternion and the quaternion computed from the CAD files, was in the order of 1.5 degrees for roll, pitch, and yaw. For the translation component, ${}^{I}\mathbf{p}_{C}$, we used the values from the CAD files. That said, there are other calibration algorithms we could have used, such as [**46, 63**].

2.4. Extended Kalman Filter

The state vector at time t_k that we are interested in estimating has the following form:

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{X}_{IMU_k} & {}^{C_1}_{G}\mathbf{q} & {}^{G}\mathbf{p}_{C_1} & \dots & {}^{C_N}_{G}\mathbf{q} & {}^{G}\mathbf{p}_{C_N} \end{bmatrix}^T \tag{2.32}$$

where $\mathbf{X}_{IMU_k}$ is a 16×1 vector expressing the state of the IMU, and the pairs $\{{}^{C_i}_{G}\mathbf{q} \quad {}^{G}\mathbf{p}_{C_i}\}$ represent the transformations between the global frame and the i^{th} camera frame. In other words, the state vector contains the current state of the IMU

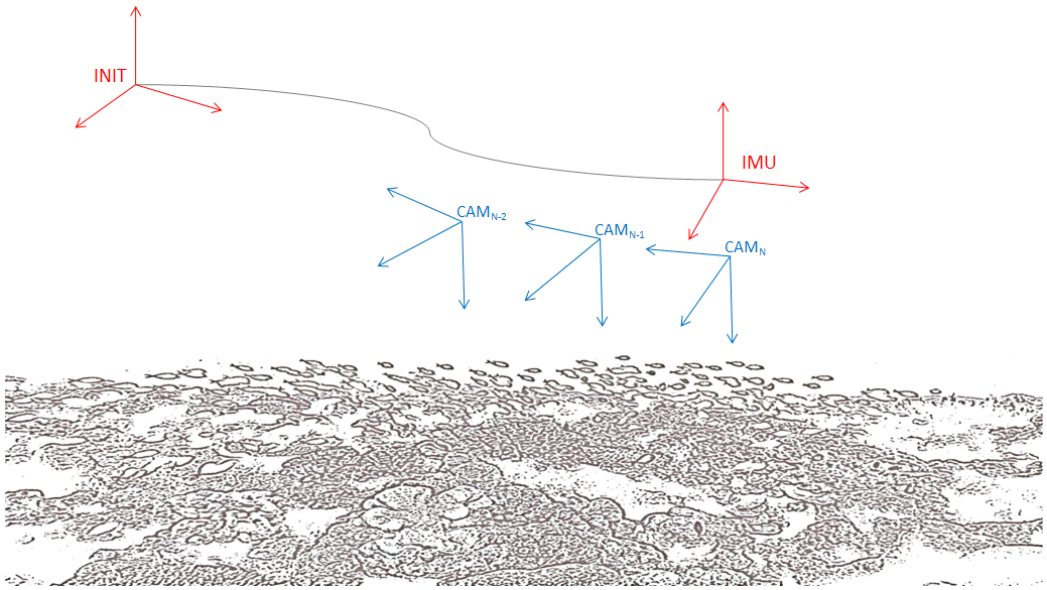


FIGURE 2.3. The IMU frame, a history of three consecutive camera frames, and the initial frame that represents the global frame of reference G.

motion characteristics and a brief history of snapshots of consecutive poses representing where the camera frame $\{C\}$ was in 3D space when images were recorded 2.3. In particular, ${}^{C_i}_{G}\mathbf{q}$ is a unit quaternion that expresses the rotation from the global frame to the frame of camera C_i, and ${}^{G}\mathbf{p}_{C_i}$ is the origin of camera frame C_i given in coordinates of the global reference frame G. In total, the state vector $\mathbf{X}_k$ is of length $16 + 7N$, where N is the number of camera frames that we wish to remember in the state vector. As will be explained in more detail later, camera frames are appended to the state vector every time an image is recorded, and the oldest one is removed whenever there are N_{max} camera frames in the history. The IMU state vector, consists of the following:

$$\mathbf{X}_{IMU_k} = \begin{bmatrix} {}^{I}_{G}\mathbf{q} & \mathbf{b}_g & {}^{G}\mathbf{v}_I & \mathbf{b}_a & {}^{G}\mathbf{p}_I \end{bmatrix} \tag{2.33}$$

where ${}^{I}_{G}\mathbf{q}$ is a quaternion that expresses the rotation from the global frame to the IMU frame, $\mathbf{b}_g$ is the bias of the gyroscope, ${}^{G}\mathbf{v}_I$ is the velocity of the IMU frame in coordinates of the global frame, $\mathbf{b}_a$ is the bias of the accelerometer, and ${}^{G}\mathbf{p}_I$ is the origin of the IMU frame in global coordinates.

The final detail to notice at this point regarding the choice of the state vector is that, unlike the traditional formulation of EKF-SLAM [**23**], the state vector in this formulation does not contain 3D landmarks, although the algorithm will provide estimates for them. This simply means that we are not modeling the cross-correlation between landmarks and the robot's motion, or the cross-correlation among landmarks, which allows for efficient updates, as will be seen below.

We will denote the estimates of the true state $\mathbf{X}_k$ by $\hat{\mathbf{X}}_k$. Following the discussion on the small angle approximation of error quaternions shown in Eq. (2.28), the error vector will be of size $15 + 6N$ and will look like this:

$$\tilde{\mathbf{X}}_k = \begin{bmatrix} \tilde{\mathbf{X}}_{IMU_k} & {}^{C_1}_{G}\tilde{\boldsymbol{\theta}} & {}^{G}\tilde{\mathbf{p}}_{C_1} & \dots & {}^{C_N}_{G}\tilde{\boldsymbol{\theta}} & {}^{G}\tilde{\mathbf{p}}_{C_N} \end{bmatrix}^T \tag{2.34}$$

where the error in the IMU state is:

$$\tilde{\mathbf{X}}_{IMU_k} = \begin{bmatrix} {}^{I}_{G}\tilde{\boldsymbol{\theta}} & \tilde{\mathbf{b}}_g & {}^{G}\tilde{\mathbf{v}}_I & \tilde{\mathbf{b}}_a & {}^{G}\tilde{\mathbf{p}}_I \end{bmatrix} \tag{2.35}$$

As mentioned in the introduction of this thesis, we are going to use an Extended Kalman Filter (EKF) that will propagate the motion of the robot forward in time, based on IMU readings, and will correct the drift using the reprojection errors from the tracking of visual features, together with depth information from the pressure sensor. The algorithm consists of four steps (see Fig. 1):

(a) In the propagation step the linear acceleration and angular velocity measurements from the IMU are integrated, in order to provide a short-term, noisy estimate of the robot's motion. Only the IMU state vector, $\mathbf{X}_{IMU_k}$ and the corresponding IMU covariance matrix are modified during this phase; the state and covariance of the camera frames remains intact.

(b) When the camera records an image, the camera frame is appended to the state vector and the covariance grows analogously. Also, current features are matched to previous features. If the maximum number of camera frames in the state vector has been reached, then old frames are pruned.

```
average IMU readings to compute mean gravity
compute initial roll and pitch from mean gravity
while true do
  if IMU message ready then
    propagate
  end if
  if Camera message ready then
    extract features from current image
    match features between current and previous image
    augment the state and the covariance
    depth-based update
    for each terminated feature trail do
      compute the feature's 3D position
      compute the feature's residual
      if the feature is not an outlier then
        vision-based update
      end if
    end for
  end if
end while
```

Algorithm 1: The general structure of the state estimation algorithm

(c) Depth measurements from a pressure sensor are available at the same rate as the IMU, however, we sample them at the same rate as the camera update rate to perform the depth-based update.

(d) If there exist feature tracks that ceased to appear in the current image, the 3D position of said features is estimated and is regarded as 'true'. Assuming said features are not deemed to be outliers, the residual between the 'true' feature positions and the estimated feature positions gives rise to the vision-based update, which corrects the entire state vector, and in particular, the IMU integration drift.

2.4.1. Propagation. The purpose of this step is for the filter to predict the position and orientation of the robot's IMU frame at time t_{k+1} based on the state at time t_k and a theoretical 3D kinematic model that approximates the rigid motion a moving body in 3D. If we regard time as a continuous variable, the motion model

that we use is:

$$\begin{aligned} {}^{I}_{G}\dot{\mathbf{q}}(t) &= \frac{1}{2}\begin{bmatrix} {}^{I}\boldsymbol{\omega}(t) \\ 0 \end{bmatrix} \otimes {}^{I}_{G}\mathbf{q}(t) \\ {}^{G}\dot{\mathbf{v}}_I(t) &= \mathbf{R}^T({}^{I}_{G}\mathbf{q}(t)){}^{I}\boldsymbol{\alpha}(t) + {}^{G}\mathbf{g} \\ {}^{G}\dot{\mathbf{p}}_I(t) &= {}^{G}\mathbf{v}_I(t) \end{aligned} \tag{2.36}$$

where ${}^{I}\boldsymbol{\omega}(t)$ and ${}^{I}\boldsymbol{\alpha}(t)$ are the true angular velocity and linear acceleration of the IMU frame in IMU coordinates, and $\mathbf{R}()$ is the rotation matrix equivalent to the quaternion. The derivation for the time derivative of the quaternion can be found in [**91**]. In this motion model we are modeling neither hydrodynamic drag nor Coriolis forces. The model of the IMU noises is the one used in Chatfield [**15**], whereby the errors and biases affecting the IMU measurements are given by [7]:

$$\begin{aligned} \boldsymbol{\omega}_m &= {}^{I}\boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \\ \boldsymbol{\alpha}_m &= \mathbf{R}({}^{I}_{G}\mathbf{q})({}^{G}\boldsymbol{\alpha} - {}^{G}\mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a \end{aligned} \tag{2.37}$$

where ${}^{G}\boldsymbol{\alpha}$ is the acceleration of the IMU frame as seen by the non-accelerating external observer of the global frame, ${}^{G}\mathbf{g} \simeq [0 \quad 0 \quad 9.81]$. Here we take $\boldsymbol{\alpha}_m$ to be in m/s^2 and the $\boldsymbol{\omega}_m$ to be in rad/s, but actually the IMU readings for the acceleration come in units of g. Like Chatfield [**15**] we are going to assume that the bias and noise processes involved in the IMU measurements are random walks and normally distributed, respectively:

$$\begin{aligned} \dot{\mathbf{b}}_a &\sim \mathcal{N}(\mathbf{0}, \sigma^2_{wa}\mathbf{I}_{3\times3}) \\ \dot{\mathbf{b}}_g &\sim \mathcal{N}(\mathbf{0}, \sigma^2_{wg}\mathbf{I}_{3\times3}) \\ \mathbf{n}_a &\sim \mathcal{N}(\mathbf{0}, \sigma^2_{na}\mathbf{I}_{3\times3}) \\ \mathbf{n}_g &\sim \mathcal{N}(\mathbf{0}, \sigma^2_{ng}\mathbf{I}_{3\times3}) \end{aligned}$$

[7] We are not modeling the Earth's rotation because we are assuming that the IMU will not be sensitive enough to measure it accurately.

The choice of these noise parameters has significant effect on the performance of the filter, but a procedure on how to optimally choose them, such as the one presented in [**47**], is not going to be the main focus of this thesis. Instead, suitable values that seemed to work in practice are going to be provided in the experimental section.

Now that we have a nonlinear model of how the IMU works, and how the IMU readings would affect the state vector in an idealized continuous-time setting, we can proceed in showing the EKF propagation step for discrete time, which happens every time we get IMU readings (in our case approximately 50 times per second):

$$\begin{aligned}
\hat{\boldsymbol{\omega}} &= \boldsymbol{\omega}_m - \hat{\mathbf{b}}_g \\
\hat{\boldsymbol{\alpha}} &= \boldsymbol{\alpha}_m - \hat{\mathbf{b}}_a \\
{}^{I}_{G}\hat{\mathbf{q}}(t_{k+1}) &= \begin{bmatrix} \frac{\hat{\boldsymbol{\omega}}}{\|\hat{\boldsymbol{\omega}}\|}\sin(\|\hat{\boldsymbol{\omega}}\|\frac{(t_{k+1}-t_k)}{2}) \\ \cos(\|\hat{\boldsymbol{\omega}}\|\frac{(t_{k+1}-t_k)}{2}) \end{bmatrix} \otimes {}^{I}_{G}\hat{\mathbf{q}}(t_k) \\
{}^{G}\dot{\hat{\mathbf{v}}}_I &= \mathbf{R}^T({}^{I}_{G}\hat{\mathbf{q}})\hat{\boldsymbol{\alpha}} + {}^{G}\mathbf{g} \\
{}^{G}\dot{\hat{\mathbf{p}}}_I &= {}^{G}\hat{\mathbf{v}}_I \\
\dot{\hat{\mathbf{b}}}_a &= \mathbf{0} \\
\dot{\hat{\mathbf{b}}}_g &= \mathbf{0}
\end{aligned} \tag{2.38}$$

Numerical integration of this system is done using 4^{th}-order Runge-Kutta, modified to account for the multiplicative correction of the quaternion. Notice the use of the 0^{th}-order quaternion integrator in the above system. The proof for why it is the approximate solution to Eq. (2.36) can be found in [**92**]. We have observed that 300-500 iterations of the Runge-Kutta integrator are sufficient, while less than 100 iterations produce noticeable numerical inaccuracies in terms of integration residual. The initial values of the IMU state vector are also very critical to the results of the filter. In particular, a very important assumption that this filter makes is that ${}^{G}\hat{\mathbf{v}}_I(t_0) = \mathbf{0}$, i.e. the robot starts from a resting position. Under this assumption, the initial acceleration measured by the accelerometer will be solely gravitational, which means that by aligning said vector to point downwards on the z-axis of the IMU, we

can compute the initial orientation, ${}^{I}_{G}\hat{\mathbf{q}}(t_0)$ of the robot uniquely modulo the yaw. Initial position is trivially set to zero. If the above assumption is not made then in the case where the filter starts estimating from a point in time when the robot is accelerating, the initial estimate of the orientation will have errors and there will be no good initial estimate of the velocity.

So far we have mentioned how the state vector $\mathbf{X}_{k|k}$ is propagated to $\mathbf{X}_{k+1|k}$, but we haven't mentioned how the covariance matrix will change:

$$\mathbf{P}_k = \begin{bmatrix} cov(\tilde{\mathbf{X}}_{IMU_k}) & cov(\tilde{\mathbf{X}}_{IMU_k}, \tilde{\mathbf{X}}_{CAM_k}) \\ cov(\tilde{\mathbf{X}}_{IMU_k}, \tilde{\mathbf{X}}_{CAM_k})^T & cov(\tilde{\mathbf{X}}_{CAM_k}) \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{II_k} & \mathbf{P}_{IC_k} \\ \mathbf{P}^T_{IC_k} & \mathbf{P}_{CC_k} \end{bmatrix} \tag{2.39}$$

The part of the covariance matrix that contains the cross-correlations between the stored camera frames, $\mathbf{P}_{CC_k}$, will remain intact during the propagation step, while the other components will change. In section A.3 of the Appendix we show that the error components of the IMU state vector change as follows:

$$\begin{aligned}
{}^{I}_{G}\dot{\tilde{\boldsymbol{\theta}}} &= -\lfloor \hat{\boldsymbol{\omega}} \times \rfloor \tilde{\boldsymbol{\theta}} - \tilde{\mathbf{b}}_g - \mathbf{n}_g && (2.40)\\
{}^{G}\dot{\tilde{\mathbf{v}}}_I &= -\mathbf{R}^T({}^{I}_{G}\hat{\mathbf{q}})(\lfloor \hat{\boldsymbol{\alpha}} \times \rfloor \tilde{\boldsymbol{\theta}} + \tilde{\mathbf{b}}_a + \mathbf{n}_a) && (2.41)\\
{}^{G}\dot{\tilde{\mathbf{p}}}_I &= {}^{G}\tilde{\mathbf{v}}_I && (2.42)\\
\dot{\tilde{\mathbf{b}}}_a &\sim \mathcal{N}(\mathbf{0}, \sigma^2_{wa}\mathbf{I}_{3\times 3}) && (2.43)\\
\dot{\tilde{\mathbf{b}}}_g &\sim \mathcal{N}(\mathbf{0}, \sigma^2_{wg}\mathbf{I}_{3\times 3}) && (2.44)
\end{aligned}$$

We can write the differential equations above in a more compact matrix form, as follows:

$$\dot{\tilde{\mathbf{X}}}_{IMU} = \mathbf{F}\tilde{\mathbf{X}}_{IMU} + \mathbf{G}\mathbf{n}_{IMU} \tag{2.45}$$

$$\mathbf{n}_{IMU} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{IMU}) = \mathcal{N}(\mathbf{0}, \begin{bmatrix} \sigma^2_{ng}\mathbf{I}_{3\times 3} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma^2_{wg}\mathbf{I}_{3\times 3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma^2_{na}\mathbf{I}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma^2_{wa}\mathbf{I}_{3\times 3} \end{bmatrix}) \tag{2.46}$$

and we can show (Appendix, section A.4) that the IMU covariance, $\mathbf{P}_{II_k}$, must satisfy the matrix differential equation:

$$\dot{\mathbf{P}}_{II} = \mathbf{F}\mathbf{P}_{II} + \mathbf{P}_{II}\mathbf{F}^T + \mathbf{G}\mathbf{Q}_{IMU}\mathbf{G}^T \tag{2.47}$$

We use a 4^{th}-order Runge-Kutta integrator again, to get the propagated IMU covariance $\mathbf{P}_{II_{k+1|k}}$ starting from $\mathbf{P}_{II_{k|k}}$, which is the IMU covariance from the last update step of the filter. As in the case of Eq. (2.36) the time interval of integration starts from the timestamp of the last IMU measurement and goes on until the timestamp of the current IMU measurement. It is worth mentioning that the initial value of the IMU covariance should reflect the initial uncertainty around the estimates. For example, the covariance of the initial position will be set to epsilon, because we know it exactly. The covariance of the initial velocity should be set to $\sigma_{vel}^2\mathbf{I}_{3\times3}$. Similarly, the covariance of the initial orientation should account for small errors in roll and pitch.

Now, all that remains for the propagation step is to show how to propagate $\mathbf{P}_{IC_{k|k}}$ to $\mathbf{P}_{IC_{k+1|k}}$. We show in section A.5 of the Appendix that:

$$\mathbf{P}_{IC_{k+1|k}} = \exp(\mathbf{F}(t_{k+1} - t_k))\mathbf{P}_{IC_{k|k}} \tag{2.48}$$

where t_{k+1} corresponds to the timestamp of the current IMU measurement and t_k is the timestamp of the last IMU measurement, and exp() in this case is the matrix exponential.

2.4.2. Augmentation. The purpose of this step is to append a new camera frame to the history maintained in the state vector. This happens whenever an image is recorded, which in our case is approximately 15 times per second. The new frame to be appended will have the following coordinates with respect to the global frame:

$$\begin{matrix} {}^{C_{N+1}}_{G}\hat{\mathbf{q}} \end{matrix} = \begin{matrix} {}^{C}_{I}\mathbf{q} \otimes {}^{I}_{G}\hat{\mathbf{q}} \end{matrix} \tag{2.49}$$

$${}^{G}\hat{\mathbf{p}}_{C_{N+1}} = {}^{G}\hat{\mathbf{p}}_I + \mathbf{R}^T({}^{I}_{G}\hat{\mathbf{q}}){}^{I}\mathbf{p}_C \tag{2.50}$$

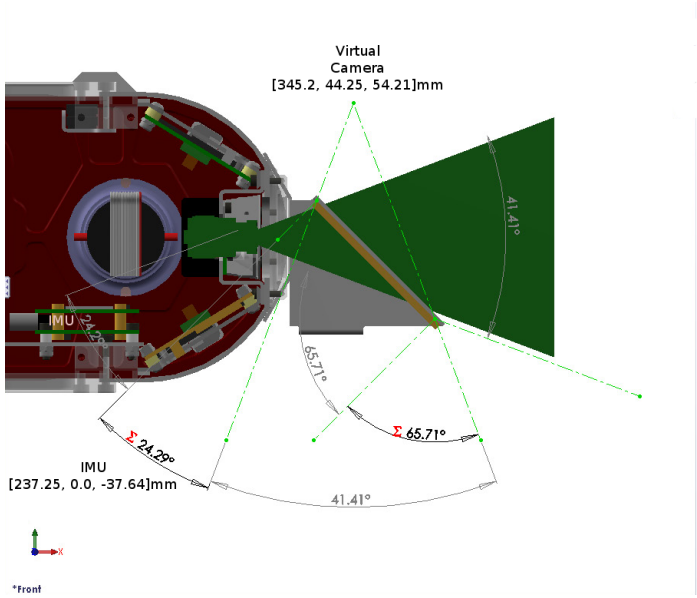


FIGURE 2.4. A schematic of the relationship between the IMU coordinate frame and the mounted camera coordinate frame.

where $\{{}^{C}_{I}\mathbf{q} \quad {}^{I}\mathbf{p}_{C}\}$ describe the transformation from the IMU to the camera frame on the robot, which we assumed to be fixed and known. Our vehicle is equipped with two cameras facing forward and one camera facing backwards. As the robot swims over the seafloor, most of the structures are below it. Therefore, in order to observe the seafloor a mirror is placed at a 45° angle in front of the back camera. This configuration results in a virtual downward-looking camera located behind the robot; see Fig. 2.4. The state vector will be augmented as follows:

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{X}_{IMU_k} & {}^{C_1}_{G}\mathbf{q} & {}^{G}\mathbf{p}_{C_1} & \dots & {}^{C_N}_{G}\mathbf{q} & {}^{G}\mathbf{p}_{C_N} & {}^{C_{N+1}}_{G}\mathbf{q} & {}^{G}\mathbf{p}_{C_{N+1}} \end{bmatrix}^T \tag{2.51}$$

and if $N+1 > N_{max}$ then $\{{}^{C_1}_{G}\mathbf{q} \quad {}^{G}\mathbf{p}_{C_1}\}$ is going to be removed from the state together with the corresponding blocks of the covariance matrix. In the Appendix (section A.6) we show that the cross-correlations between the newly-appended camera frame and

the existing frames in the history, as well as the IMU state are given by:

$$\mathbf{P}_{aug} = \begin{bmatrix} \mathbf{P} & \mathbf{P}\mathbf{J}^T \\ \mathbf{J}\mathbf{P} & \mathbf{J}\mathbf{P}\mathbf{J}^T \end{bmatrix} \tag{2.52}$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{R}({}^{C}_{I}\mathbf{q}) & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6N} \\ -\mathbf{R}^T({}^{I}_{G}\hat{\mathbf{q}})\lfloor {}^{I}\mathbf{p}_C \times \rfloor & \mathbf{0}_{3\times 9} & \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 6N} \end{bmatrix} \tag{2.53}$$

Finally, immediately after the augmentation step we match the features from the last image with the features from the current image, thus ensuring continuity of the flow/tracking of features until the most recent image. The details of this process will be explained in detail in the next chapter.

2.4.3. 3D position estimation of observed landmarks. Consider a single feature, f, that has been tracked in n consecutive camera frames, $C_1, C_2, ..., C_n$, which are stored in the state vector (see Fig. 2.5). Let ${}^{C_i}\mathbf{p}_f = \begin{bmatrix} {}^{C_i}X_f & {}^{C_i}Y_f & {}^{C_i}Z_f \end{bmatrix}$ be the real 3D position of feature f expressed in camera frame C_i coordinates, $i \in \{1, \ldots, n\}$. We are interested in estimating this as accurately as possible, because the vision-based update will depend on it. Then, we can write the following:

$${}^{C_i}\mathbf{p}_f = \mathbf{R}({}^{C_i}_{C_1}\mathbf{q}){}^{C_1}\mathbf{p}_f + {}^{C_i}\mathbf{p}_{C_1} \tag{2.54}$$

$$= {}^{C_1}Z_f \left(\mathbf{R}({}^{C_i}_{C_1}\mathbf{q}) \begin{bmatrix} \frac{{}^{C_1}X_f}{{}^{C_1}Z_f} & \frac{{}^{C_1}Y_f}{{}^{C_1}Z_f} & 1 \end{bmatrix}^T + \frac{1}{{}^{C_1}Z_f}{}^{C_i}\mathbf{p}_{C_1} \right) \tag{2.55}$$

If we let $\alpha_f = \frac{{}^{C_1}X_f}{{}^{C_1}Z_f}$, $\beta_f = \frac{{}^{C_1}Y_f}{{}^{C_1}Z_f}$, and $\gamma_f = \frac{1}{{}^{C_1}Z_f}$ then

$${}^{C_i}\mathbf{p}_f = {}^{C_1}Z_f \left(\mathbf{R}({}^{C_i}_{C_1}\mathbf{q}) \begin{bmatrix} \alpha_f & \beta_f & 1 \end{bmatrix}^T + \gamma_f {}^{C_i}\mathbf{p}_{C_1} \right)$$

$$= {}^{C_1}Z_f \begin{bmatrix} h_{i1}(\alpha_f, \beta_f, \gamma_f) \\ h_{i2}(\alpha_f, \beta_f, \gamma_f) \\ h_{i3}(\alpha_f, \beta_f, \gamma_f) \end{bmatrix} \tag{2.56}$$

Now, assuming a simple pinhole camera model, and disregarding the effects of the

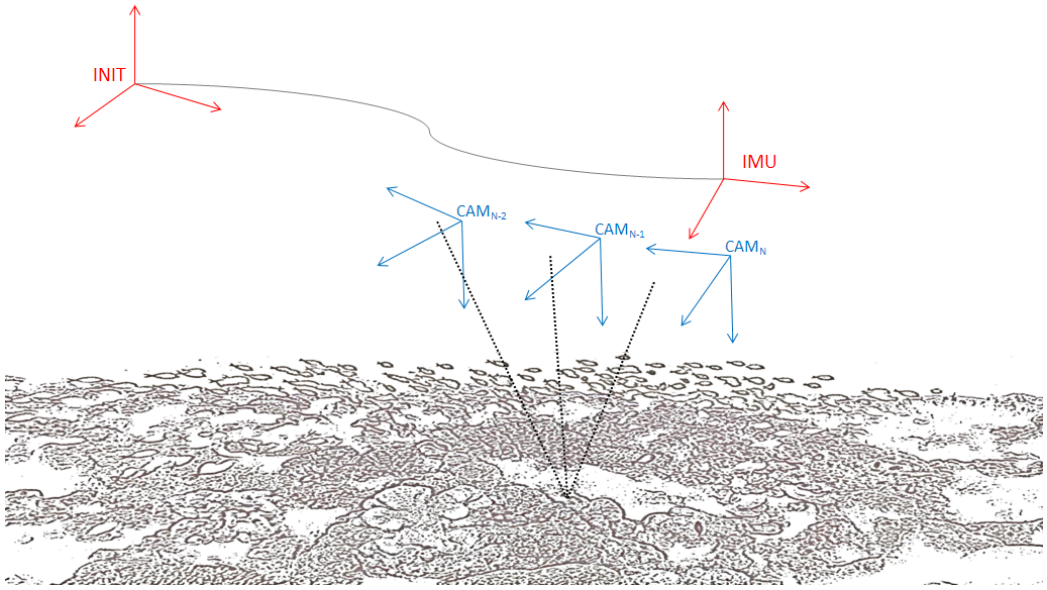


FIGURE 2.5. The 3D position estimation procedure is a procedure reminiscent of triangulation.

camera's calibration matrix, we can model the projection of feature f on the projection plane ${}^{C_i}Z = 1$ as follows:

$$\mathbf{z}_f^{C_i} = \frac{1}{h_{i3}(\alpha_f, \beta_f, \gamma_f)} \begin{bmatrix} h_{i1}(\alpha_f, \beta_f, \gamma_f) \\ h_{i2}(\alpha_f, \beta_f, \gamma_f) \end{bmatrix} + \mathbf{n}_f^{C_i} \tag{2.57}$$

where $\mathbf{z}_f^{C_i}$ is the 2×1 measurement, and $\mathbf{n}_f^{C_i}$ is noise associated with the process, due to miscalibration of the camera, motion blur, and other factors. If we stack the measurements from all cameras into a single $2n \times 1$ vector $\mathbf{z}_f$ and similarly for the projection functions $\mathbf{h}_i$ into a $2n \times 1$ vector $\mathbf{h}_f$, then we will have expressed the problem of estimating the 3D position of a feature as a nonlinear least-squares problem with 3 unknowns:

$$\underset{\alpha_f, \beta_f, \gamma_f}{\operatorname{argmin}} \|\mathbf{z}_f - \mathbf{h}_f(\alpha_f, \beta_f, \gamma_f)\| \tag{2.58}$$

Provided we have at least 3 measurements of feature f, i.e. provided we track it in at least 3 frames, we can use the Levenberg-Marquardt nonlinear optimization algorithm

in order to get an estimate ${}^{C_1}\hat{\mathbf{p}}_f$ of the true solution. Then, the estimated feature position in global coordinates can be obtained by:

$$ {}^{G}\hat{\mathbf{p}}_f = \frac{1}{\gamma_f}\mathbf{R}^T({}^{C_1}_{G}\hat{\mathbf{q}})\,[\alpha_f \quad \beta_f \quad 1]^T + {}^{G}\hat{\mathbf{p}}_{C_1} \tag{2.59} $$

One problem with this approach is that nonlinear optimization algorithms do not guarantee a global minimum, only a local one, provided they converge. Another problem is that if feature f is recorded around the same pixel location in all the frames in which it appears, then the measurements we will get are going to be linearly dependent, thus providing no information about the depth of f. In other words, feature tracks that have small baseline have to be considered outliers, unless we exploit other local information around the feature to infer its depth. This turns out to be important in the case where the robot is at rest, in which case almost all the feature tracks will have minimal baseline from one image to the next. Another potential pitfall is that the inter-camera transformations $\{{}^{C_i}_{C_1}\mathbf{q}, {}^{C_i}\mathbf{p}_{C_1}\}$ might be themselves noisy, which will also affect the solution of the least squares problem.

2.4.4. Vision-based update. The purpose of this step is to correct the predictions made during the propagation phase. Consider again a single feature f, which has been tracked in n consecutive camera frames, $C_1, C_2, ..., C_n$, but stopped being tracked at the current frame, C_{n+1}. In this case we initiate the vision-based update. After having estimated the 3D position of feature f in global coordinates, as described previously, we expect that its projection on the image plane of camera frame C_i, according to the pinhole camera model, will be:

$$ \hat{\mathbf{z}}_f^{C_i} = \begin{bmatrix} \frac{{}^{C_i}\hat{X}_f}{{}^{C_i}\hat{Z}_f} & \frac{{}^{C_i}\hat{Y}_f}{{}^{C_i}\hat{Z}_f} \end{bmatrix} \tag{2.60} $$

where

$$ \begin{bmatrix} {}^{C_i}\hat{X}_f & {}^{C_i}\hat{Y}_f & {}^{C_i}\hat{Z}_f \end{bmatrix}^T = \mathbf{R}({}^{C_i}_{G}\hat{\mathbf{q}})({}^{G}\hat{\mathbf{p}}_f - {}^{G}\hat{\mathbf{p}}_{C_i}) \tag{2.61} $$

The actual measurement obtained from the feature tracks, on the other hand, is $\mathbf{z}_f^{C_i}$, so for a single feature f viewed from a camera frame C_i this gives rise to the residual

$$\mathbf{r}_f^{C_i} = \mathbf{z}_f^{C_i} - \hat{\mathbf{z}}_f^{C_i} \tag{2.62}$$

If we take the Taylor expansion of the function $\mathbf{r}_f^{C_i}$ about the point $({}_G^{C_i}\hat{\mathbf{q}}, {}^G\hat{\mathbf{p}}_f, {}^G\hat{\mathbf{p}}_{C_i})$ we will get the following linearization:

$$\mathbf{r}_f^{C_i} \simeq \mathbf{H}_f^{C_i}\tilde{\mathbf{X}} + \mathbf{U}_f^{C_i G}\tilde{\mathbf{p}}_f + \mathbf{n}_f^{C_i} \tag{2.63}$$

where $\mathbf{H}_f^{C_i}$ and $\mathbf{U}_f^{C_i}$ are the Jacobians of $\mathbf{r}_f^{C_i}$ at the chosen point of linearization (see section A.7 in the Appendix). $\mathbf{n}_f^{C_i} \sim \mathcal{N}(0, \mathbf{R}_f^{C_i})$ is the uncertainty in the residual, with $\mathbf{R}_f^{C_i} = \sigma_{im}^2\mathbf{I}_{2\times 2}$. Essentially, σ_{im} models the uncertainty in the camera measurements, and we are currently modeling it as being the same for all camera frames, regardless of whether a particular image has high levels of motion blur, or whether the viewed scene is nearby or far away. The total projection residual from all camera frames in which feature f was tracked is therefore a stack of the individual residuals:

$$\mathbf{r}_f \simeq \mathbf{H}_f\tilde{\mathbf{X}} + \mathbf{U}_f{}^G\tilde{\mathbf{p}}_f + \mathbf{n}_f \tag{2.64}$$

where $\mathbf{n}_f$ is the total uncertainty of the residual due to feature f. We make the assumption that observations of the same feature in consecutive camera frames are statistically independent, which makes the covariance of said uncertainty $\mathbf{R}_f = \sigma_{im}^2\mathbf{I}_{n\times n}$. The problem with the residual in Eq. (2.64) is that the state errors $\tilde{\mathbf{X}}$ are correlated with the errors in the feature position estimate ${}^G\tilde{\mathbf{p}}_f$, since the former was used to derive the latter, as described previously. So, using Eq. (2.64) as the EKF residual will bias the estimates. That is why we can use the closest residual that ignores the first-order dependence on ${}^G\tilde{\mathbf{p}}_f$:

Let $\mathbf{A}_f$ be a matrix such that $\mathbf{A}_f^T\mathbf{U}_f = \mathbf{0}$, in other words, the columns of $\mathbf{A}$ form an orthonormal basis of the null space of $\mathbf{U}_f^T$. While the choice of $\mathbf{A}$ is not unique, the final correction factor applied to the state and covariance will not be affected by this. The residual due to a single feature f that does not depend on the feature

position errors is:

$$\mathbf{r}'_f = \mathbf{A}_f^T \mathbf{r}_f \simeq \mathbf{A}_f^T \mathbf{H}_f \tilde{\mathbf{X}} + \mathbf{A}_f^T \mathbf{n}_f = \mathbf{H}'_f \tilde{\mathbf{X}} + \mathbf{n}'_f \tag{2.65}$$

where $\mathbf{n}'_f \sim \mathcal{N}(0, \mathbf{R}_f)$ due to the column-wise orthonormality of $\mathbf{A}$. Now, the total residual for all the features that ceased to be tracked and are part of the update, is a stack of the above individual residuals:

$$\mathbf{r}' = \mathbf{H}' \tilde{\mathbf{X}} + \mathbf{n}' \tag{2.66}$$

At this point we make another assumption, whereby observations of different features are statistically independent, which makes $\mathbf{n}' \sim \mathcal{N}(\mathbf{0}, \sigma_{im}^2 \mathbf{I})$. The residual of Eq. (2.66) is now in a form where the EKF framework can be applied. That said, Mourikis and Roumeliotis [**66**] further apply a numerical speed-up step whereby they use the QR decomposition of $\mathbf{H}'$ to reduce the dimension of said matrix, which is going to be used in the expensive matrix multiplications and inversion of the EKF:

$$\mathbf{H}' = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{R}' \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_1 \mathbf{R}' \tag{2.67}$$

where $\mathbf{Q}_1$ is unitary. From Eq. (2.66) we can write

$$\mathbf{r}'' = \mathbf{Q}_1^T \mathbf{r}' = \mathbf{R}' \tilde{\mathbf{X}} + \mathbf{Q}_1^T \mathbf{n}' \tag{2.68}$$

$$\mathbf{K} = \mathbf{P}_{k+1|k} \mathbf{R}'^T (\mathbf{R}' \mathbf{P}_{k+1|k} \mathbf{R}'^T + \sigma_{im}^2 \mathbf{I})^{-1} \tag{2.69}$$

$$\boldsymbol{\delta x} = \mathbf{K} \mathbf{r}'' \tag{2.70}$$

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K} \mathbf{R}') \mathbf{P}_{k+1|k} \tag{2.71}$$

where $\boldsymbol{\delta x}$ is the correction vector, of size $15 + 6N$ for the state vector. As mentioned in the beginning of this chapter, vectorial quantities on the state vector are updated additively, while for quaternions we take each linearized quaternion $\tilde{\boldsymbol{\theta}}$ in $\boldsymbol{\delta x}$ and we

correct the corresponding quaternions of the state vector as follows:

$$\begin{aligned} \boldsymbol{\delta q} &= \left[\tilde{\boldsymbol{\theta}}/2 \quad \sqrt{1 - \|\tilde{\boldsymbol{\theta}}\|/4}\right] \\ \hat{\boldsymbol{q}}_{k+1|k+1} &= \boldsymbol{\delta q} \otimes \hat{\mathbf{q}}_{k+1|k} \end{aligned} \tag{2.72}$$

It is worth mentioning at this point that the approximation of the error quaternion by $\tilde{\boldsymbol{\theta}}$ only holds for small angles, so if $\|\tilde{\boldsymbol{\theta}}\| >= 4$ the estimate will almost surely have diverged.

2.4.5. Depth-based update. The Aqua family of amphibious robots is equipped with a pressure sensor that is exposed to the water and measures its hydrostatic pressure. The sensor is calibrated so that its reference point is at the surface of the sea, and its pressure measurements are converted into depth measurements in a linear fashion. We model the incoming data at time t_k of absolute depth from the sea surface as:

$$z_k = d_k + n_k, \quad n_k \sim \mathcal{N}(0, \sigma^2_{depth}). \tag{2.73}$$

That said, we are interested in the difference between the robot's initial and current depth, i.e. in the displacement on the z-axis with respect to the global frame of reference, so our measurements are:

$$\begin{aligned} z'_k &= z_k - z_0 = d_k - d_0 + n_k = {}^G Z_{I_k} + n_k \\ n_k &\sim \mathcal{N}(0, \sigma^2_{depth}) \end{aligned} \tag{2.74}$$

The EKF state estimate for the depth change is ${}^G\hat{Z}_{I_k}$, so the measurement residual becomes $r_k = {}^G Z_{I,k} - {}^G\hat{Z}_{I,k} + n_k$. The covariance matrix of the uncertainty in the residual is

$$\begin{aligned} S_k &= \mathbf{H}_{depth}\mathbf{P}_{k|k-1}\mathbf{H}^T_{depth} + \sigma^2_{depth} \\ \mathbf{H}_{depth} &= \begin{bmatrix} 0_{1\times 14} & 1 & 0_{1\times 6N} \end{bmatrix} \end{aligned} \tag{2.75}$$

The Kalman gain is then given by $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T_{depth}S^{-1}_k$, so the state correction vector is $\mathbf{K}_k r_k$. This correction will potentially affect the entire state vector. The

EKF update equations for the $\mathbf{P}_{k+1|k+1}$ and $\mathbf{X}_{k+1|k+1}$ are identical to those presented in the vision-based update step.

We perform the depth-based update right before the augmentation step, every time an image is recorded. One very important issue that needs to be mentioned is the presence of numerical instabilities when the effects of these updates are compounded, thus for example, making the state covariance non-symmetric. We address this particular issue by enforcing the symmetry of the covariance after each update, by $\mathbf{P}_{k+1|k+1} \leftarrow (\mathbf{P}_{k+1|k+1} + \mathbf{P}^T_{k+1|k+1})/2$.

CHAPTER 3

FEATURE TRACKING

In the previous chapter we mentioned that features are tracked between consecutive pairs of images and that gives rise to feature paths, which enables us to estimate the 3D position of each point of interest in the environment. This is a crucial component of the algorithm and, ideally, it needs to satisfy certain requirements such as: speed of tracking, low rate of false positives, and long tracking lengths whenever necessary. Speed of tracking is important when real-time operation of the EKF is one of the objectives. This of course depends on the particular computing platform, but regardless of the available hardware, our ultimate future goal in terms of processing speed in this line of work is 5-10 camera frames per second. Accuracy, expressed as a low percentage of false positive matches, is also very important because matching errors will lead to 3D feature position estimation errors, which in turn will lead to errors in the update step. A more quantitative analysis of this effect will be presented in the next chapter. Finally, the third requirement, long tracking lengths, are necessary when the robot is moving slowly, or in other words, when the baseline between consecutive camera frames is too small to accurately estimate the depth at which the feature is seen.

Keeping all these considerations in mind, this chapter presents an experimental evaluation of different feature trackers, aiming to compare how they perform with respect to the criteria outlined above in underwater environments. A feature tracker

consists of a *detector* that will consistently identify points of interest, a *descriptor* that will assign a signature to each of those points, and a *matching* technique that will match points based on their signatures.

3.1. Feature detectors and descriptors

Our evaluation compares the following feature detectors: SURF, Shi-Tomasi, FAST, CenSurE, and the SURF descriptor.

The SURF [**12**] detector is designed to be scale and (partially) rotation invariant. It detects keypoints at local extrema of the determinant of the image's approximate Hessian. This filter response is evaluated at different scales so as to achieve scale invariance. The scale space is divided into overlapping *octaves*, each of which consists of filter responses at increasing scales. Rotation invariance is due to the assignment of a dominant orientation of Haar wavelet responses around the detected keypoint. Each keypoint is characterized by a SURF descriptor, which is a vector that consists of sums of Haar wavelet responses around its oriented neighborhood. Neither the detector nor the descriptor use any color information.

The Shi-Tomasi detector [**78**] is invariant under affine transformations, but not under scaling. Its keypoints are image locations where the 2^{nd} moment matrix has two large eigenvalues, indicating image intensity change in two directions, i.e. a corner.

The FAST (Features from Accelerated Segment Test) [**74**] detector, on the other hand, focuses on speed of keypoint detection rather than robustness to noise and invariance properties. It uses a decision tree to classify a pixel as a keypoint if a circle of pixels around it has arcs that are much brighter or darker than the center. The authors mention that while this classifier is not very robust to noise, scaling and illumination variations, it has high repeatability.

Finally, the CenSurE (Center Surround Extrema) [**1**] detector (also known as STAR) aims to identify keypoints at any scale, without resorting to image subsampling like SIFT [**57**], or filter upsampling like SURF. It does this by searching for extrema of the image's approximate Laplacian, using polygonal (e.g. octagonal)

center-surround Haar wavelet filters. By comparison, SURF uses box-shaped filters for its descriptor, resulting in less symmetry and partial rotation invariance. Finally, among the detected extrema, only the ones with high Harris response are kept, so as to eliminate keypoints detected along edges, which might be unstable for tracking.

3.1.1. Feature matching. The feature matching component of our evaluation is based on Muja and Lowe's work [**67**], the Fast Library for Approximate Nearest Neighbours (FLANN). More specifically, we match feature pairs based on Approximate Nearest Neighbor search among the respective SURF descriptors. We accept a match if the distance ratio of the nearest neighbour over the second nearest neighbour is below a certain threshold, and provided the feature correspondence is bi-directional. For the Approximate Nearest Neighbour (ANN) search we experimented with both hierarchical K-means and randomized kd-trees. For combinations that use FAST, CenSurE, or Shi-Tomasi keypoints, we use normalized cross-correlation (ZNCC) as a measure of similarity between image patches around said keypoints.

3.1.2. Outlier detection. In order to identify false positive matches between two consecutive frames C_i and C_{i+1} we use two techniques based on two criteria:

(i) The epipolar constraints imposed by the fundamental matrix.

(ii) A a χ^2 test from robust statistics that detects when the residual due to a terminated feature trail is too large (i.e. outlier).

For the first criterion we rely on two different methods of estimating the fundamental matrix: one is the RANSAC [**13**] method, which works well as long as there is a significant number of inliers among the matches. The other one is via direct computation [**34**], which is made possible due to the presence of the camera transformations in the state vector. Specifically, we can express the fundamental matrix $\mathbf{\Phi}$ in terms of the calibration matrix $\mathbf{K}$ of the camera, and the transformation between camera frames C_{i+1} and C_i:

$$\begin{aligned} \mathbf{\Phi} &= \mathbf{K}^{-T} \lfloor {}^{C_i}\hat{\mathbf{p}}_{C_{i+1}} \times \rfloor \mathbf{R}({}^{C_{i+1}}_{C_i}\hat{\mathbf{q}}) \mathbf{K}^{-1} \\ \mathbf{R}({}^{C_{i+1}}_{C_i}\hat{\mathbf{q}}) &= \mathbf{R}({}^{C_{i+1}}_{G}\hat{\mathbf{q}}) \mathbf{R}^T({}^{C_i}_{G}\hat{\mathbf{q}}) \\ {}^{C_i}\hat{\mathbf{p}}_{C_{i+1}} &= \mathbf{R}({}^{C_i}_{G}\hat{\mathbf{q}})({}^{G}\hat{\mathbf{p}}_{C_{i+1}} - {}^{G}\hat{\mathbf{p}}_{C_i}) \end{aligned} \tag{3.1}$$

We classify a pair of keypoints, $\mathbf{x}_1$ and $\mathbf{x}_2$ written in homogeneous pixel coordinates, as an outlier match if for either of the two estimates of the fundamental matrix:

$$\mathbf{x}_2^T \mathbf{\Phi} \mathbf{x}_1 > \tau \tag{3.2}$$

where τ is a threshold (in our experiments $\tau = 1.0$ pixels). We classify a feature trail as an outlier if any of its constituent matching pairs are outliers.

The second outlier detection technique computes the Mahalanobis norm of the residual due to a single feature f (see Eq. 2.65) that has been observed in M_f consecutive camera frames, but stopped being tracked:

$$d_f^2 = \mathbf{r}_f'^T (\mathbf{H}_f' \hat{\mathbf{P}} \mathbf{H}_f'^T + \sigma_{im}^2 \mathbf{I}_{2M_f-3})^{-1} \mathbf{r}_f' \tag{3.3}$$

Notice that $\mathbf{H}_f' \hat{\mathbf{P}} \mathbf{H}_f'^T + \sigma_{im}^2 \mathbf{I}_{2M_f-3}$ is the covariance matrix of the normally-distributed residual $\mathbf{r}_f'$, and for this norm to be meaningful it needs to be positive definite [1]. The norm measures how far the observed feature measurements were with respect to the expected feature measurement, given the uncertainty in the camera frames. It is a known result that the distribution of the squared Mahalanobis distance of a normally-distributed vector is chi-squared with as many degrees of freedom as the size of the vector. In other words, $d_f^2 \sim \chi^2(2M_f - 3)$. What this means in our case is that we can compare the particular instantiation of this squared distance with the cumulative distribution of $\chi^2(2M_f - 3)$, and if said distance is in the 'tail' of the chi-squared then we consider the feature an outlier because it is too surprising with respect to the expected measurements. More formally, if d_f^2 is within the 95^{th} percentile of

[1] See previous discussion on the covariance of the error quaternion following Eq. (2.27)

TABLE 3.1. Feature trackers included in the evaluation

Detector	Descriptor	Matching scheme [2]
SURF	SURF	ANN
SURF	Image patch	ZNCC
Shi-Tomasi	Image patch	ZNCC
FAST	Image patch	ZNCC
STAR	Image patch	ZNCC

$\chi^2(2M_f - 3)$ then we accept it as an inlier. Otherwise, we consider it an outlier and we ignore it during the update step.

In method (i) the direct computation method is useful as a complement to RANSAC when the presence of outlier matches is significant, however the use of these two methods incurs a significant computational cost in a feature tracking system that we want to run online. This is why in our system we mainly use (ii).

3.2. Experimental evaluation of feature trackers

Our experimental comparison included the following combinations of detectors, descriptors, and matching strategies:

For each of the experiments conducted we attempted to tune the parameters of individual feature trackers in such a way that each of them performs as well as possible. The outlier detection parameters were fixed for all experiments, so that all combinations are treated in the same way.

3.2.1. Underwater camera and IMU datasets. We rely on 3 different datasets of camera and IMU readings in order to conduct our evaluation. The datasets represent distinct underwater environments that are relatively feature rich, and have been recorded under varying illumination conditions. All datasets were recorded at approximately 30 feet depth. The frame rate of the downward-looking camera was set at 15Hz. The camera is a PointGrey Dragonfly Hi-Color 1024×768 pixels. The IMU sensor we used was a MicroStrain 3DM-GX1, and its sampling rate was set at

[2] ANN (Approximate Nearest Neighbour Search), ZNCC (Normalized Cross-Correlation)

50Hz. Both sensors are on board of one of the Aqua family of amphibious robots [**75**]. More specifically:

Dataset1 features a straight 30 meter-long trajectory, where the robot moves at approximately 0.2 meters/sec forward, while preserving its depth. The sea bottom is mostly flat, and the robot moves about 2 meters over it. A white 30 meter-long tape has been placed on the bottom, both to provide ground truth for distance travelled, and to facilitate the straight line trajectory.

Dataset2 is an L-shaped trajectory, where the robot goes straight for about 7 meters, turns left, and then continues straight for about 3 meters, while preserving depth. Again, the sea bottom is mostly flat, and the robot moves about 2 meters over it, at speed 0.3 meters/sec. Another way it differs from *Dataset1* is that it has a higher degree of motion blur.

Dataset3 features a remotely controlled trajectory over a coral reef, where the robot moves for 20 seconds at speed 0.2m/s. It is the most feature-rich and blur-free among the datasets we have used, it consists of 300 images and it differs from the previous two datasets in that the robot changes depth.

3.2.2. Feature track lengths. The distribution of feature track lengths is shown in Fig. 3.4, and it is a decaying curve for all combinations. The SURF detector seems to track most features for an average of 5 camera frames, and it is virtually unable to do it for more than 25 frames, using Approximate Nearest Neighbors as the matching method. From the datasets we observed that, given the almost constant forward velocity of the robot, each feature is visible in the camera's field of view for about 25 frames. This means that on average the combination mentioned above can track approximately $1/5^{th}$ of the real trajectory of a feature. Fig. 3.5 shows that this tracking is very accurate.

3.2.3. False positive matches. Figure 3.5 demonstrates the difference in accuracy between the combinations involving SURF, and all the rest. Matching FAST keypoints with the normalized cross-correlation similarity score results in as many as

9% false matches, while the same quantity appears to be 5% for SURF. This difference is pronounced when the image has motion blur, since the normalized cross-correlation similarity measure does not take into account image intensity gradients of any sort, so one should apply very aggressive outlier detection schemes when using it.

3.2.4. Running time. We compared the average time spent on keypoint extraction and matching, normalized by the total number of keypoints processed. The results were obtained on an Intel Pentium Core 2 Duo at 1.6GHz and 4GB of RAM. The OpenCV C++ library implementations were used for each detector, descriptor and matching scheme. The results in Fig. 3.6(a) clearly show that SURF and the Shi-Tomasi detector spend approximately 1 millisecond on each feature, while CenSurE spends 3 milliseconds. The latter is most likely due to the fact that CenSurE computes keypoints at all scales. Fig. 3.6(b) depicts a very significant difference between the running times of Approximate Nearest Neighbor matching through either randomized kd-trees or K-means, and normalized-cross correlation matching. This is not surprising, as the implementation of the latter has complexity $O(nm)$ where n, m are the number of features extracted from the previous and the current image, respectively. On the other hand, kd-trees are constructed in $O(nlogn)$ and queried in $O(logm)$. Therefore, if real-time processing of images is a requirement, limiting the number of features extracted by the detectors becomes necessary. Specifically, in all of our feature tracking experiments we have used no more than 800 features per image, but no less than 200, before the outlier detection phase.

3.2.5. Swapping feature trackers. Figure 3.7 presents two illustrative examples of the state estimation algorithm applied to *Dataset3*, in one case using the SURF detector adjusted so that it detects approximately 500-600 features per image, and in the other case using the the Shi-Tomasi detector, tuned so that it detects approximately 200-300 features. Less than half of those features were matched from frame to frame, yet the robot's trajectory was reproduced at 1Hz camera frame rate for SURF and 3Hz frame rate for Shi-Tomasi. The estimated trajectory and 3D

structure of the coral is shown in Fig. 3.7. It is worth mentioning that this experiment did not include ground truth validation, unlike the experiments that are going to be presented in the next chapter. The purpose of this swapping was simply to illustrate that different feature trackers can be tuned to have comparable accuracy, and to highlight the existence of an accuracy-speed trade-off among them.

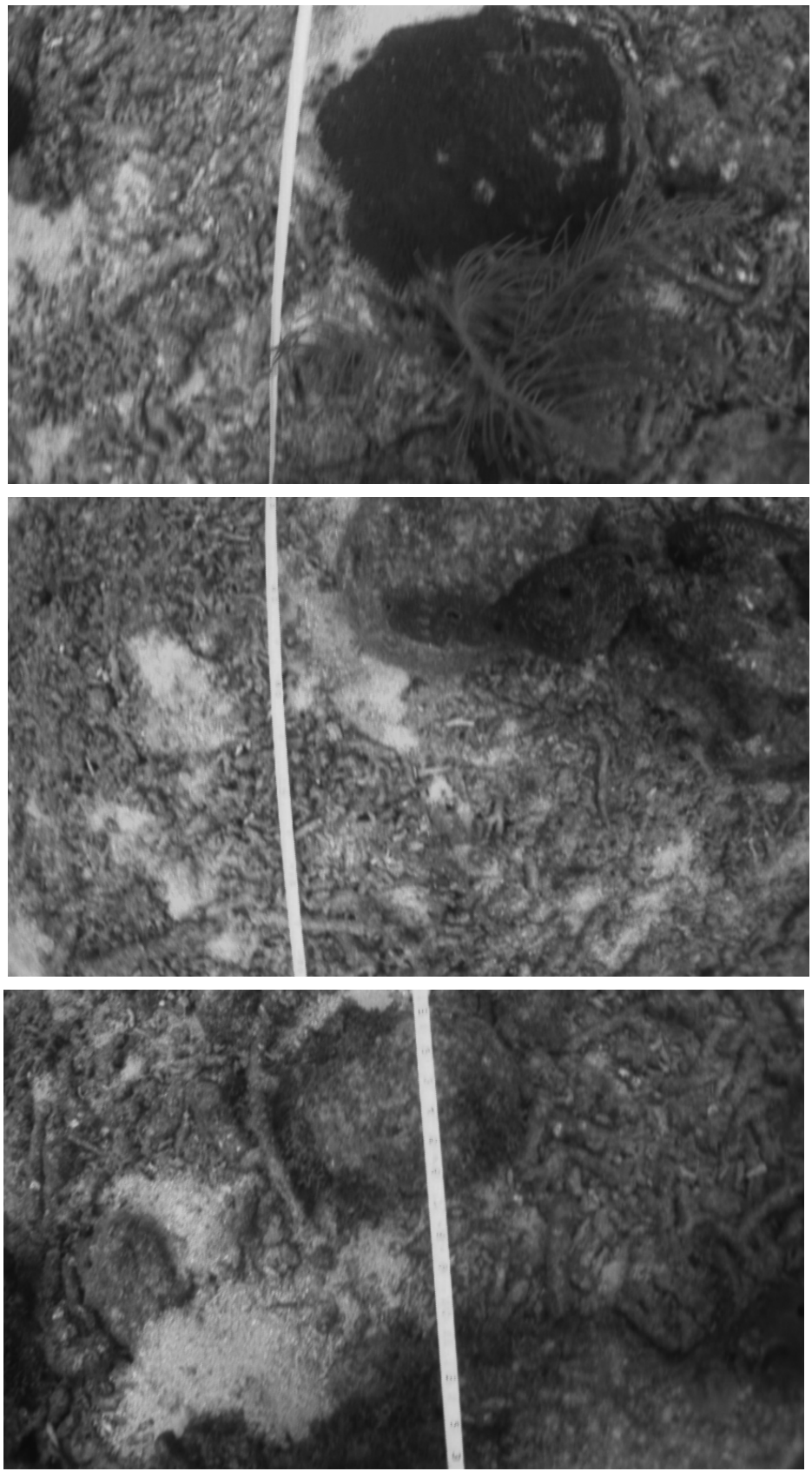


FIGURE 3.1. Underwater dataset 1. The white line is a 30m long measuring tape carefully laid on the seafloor for approximate ground truth.

FIGURE 3.2. Underwater dataset 2 with more motion blur than the other datasets.

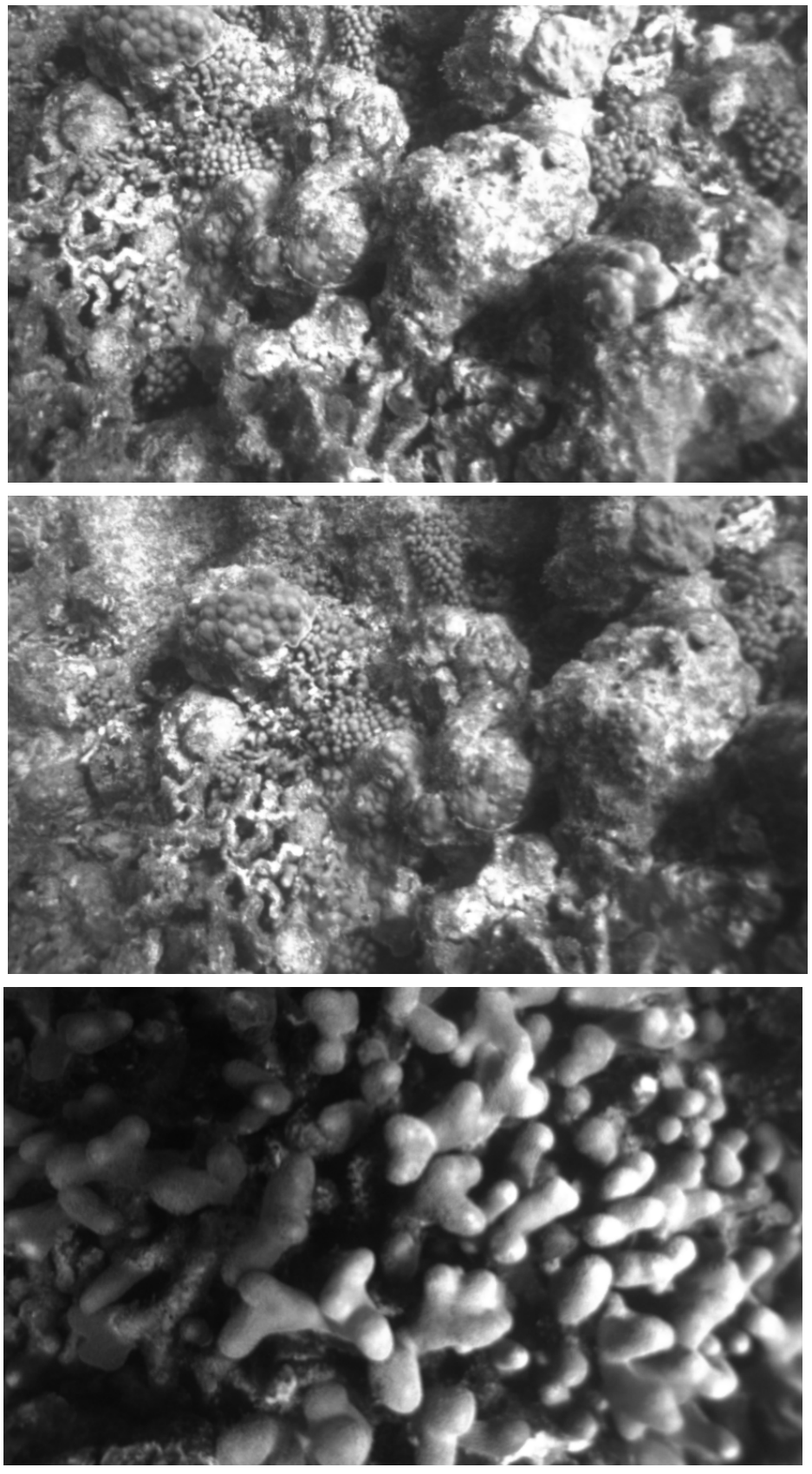


FIGURE 3.3. Underwater dataset 3

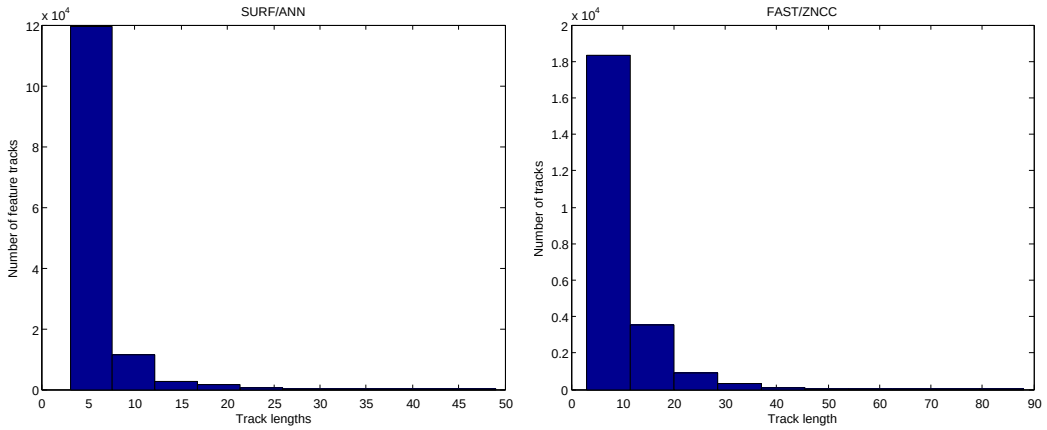


FIGURE 3.4. Distribution of feature track lengths for two representative combinations. The figure on the left shows the length distribution for the SURF detector, and Approximate Nearest Neighbor matching. The one on the right shows it for the FAST detector, matched by the normalized cross-correlation score.

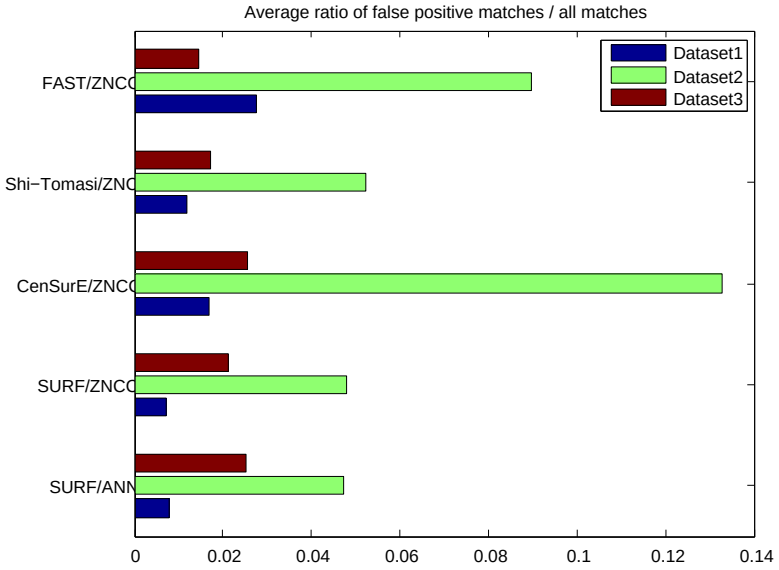
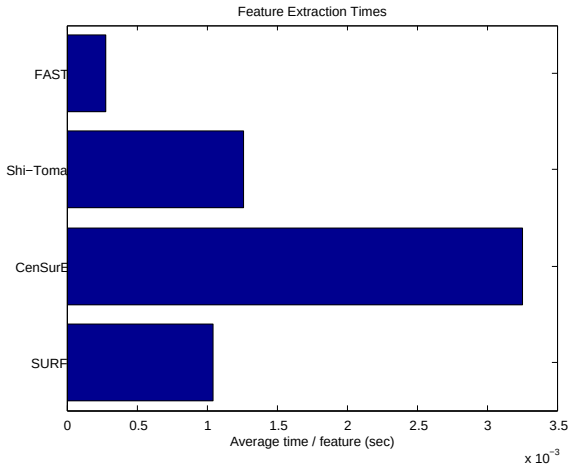
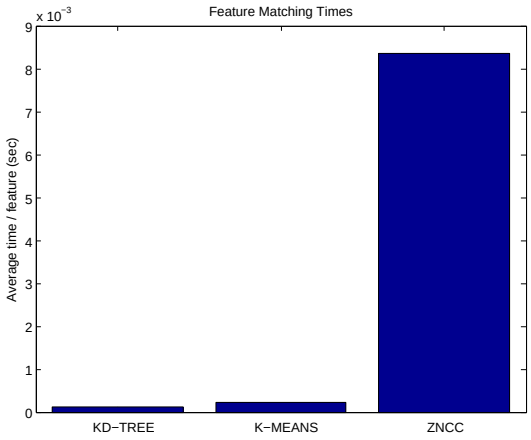


FIGURE 3.5. Average ratio of false positive matches / all matches. High value implies matching that is prone to outliers. SURF feature matching, and Shi-Tomasi features matched with ZNCC appear to be overall the most robust. All detectors did well on datasets 2 and 3, which were not blurry. FAST and CenSurE were more prone to outliers on Dataset2, which had the highest motion blur.



(a)



(b)

FIGURE 3.6. (a) Average feature extraction times per feature. CenSurE is the slowest among the detectors that we evaluated. (b) Average feature matching times per feature. The ZNCC score is computed by a brute force comparison of all possible feature pairs, hence it requires more computation time than ANN. The time spent on computing the fundamental matrix is not included in these matching times.

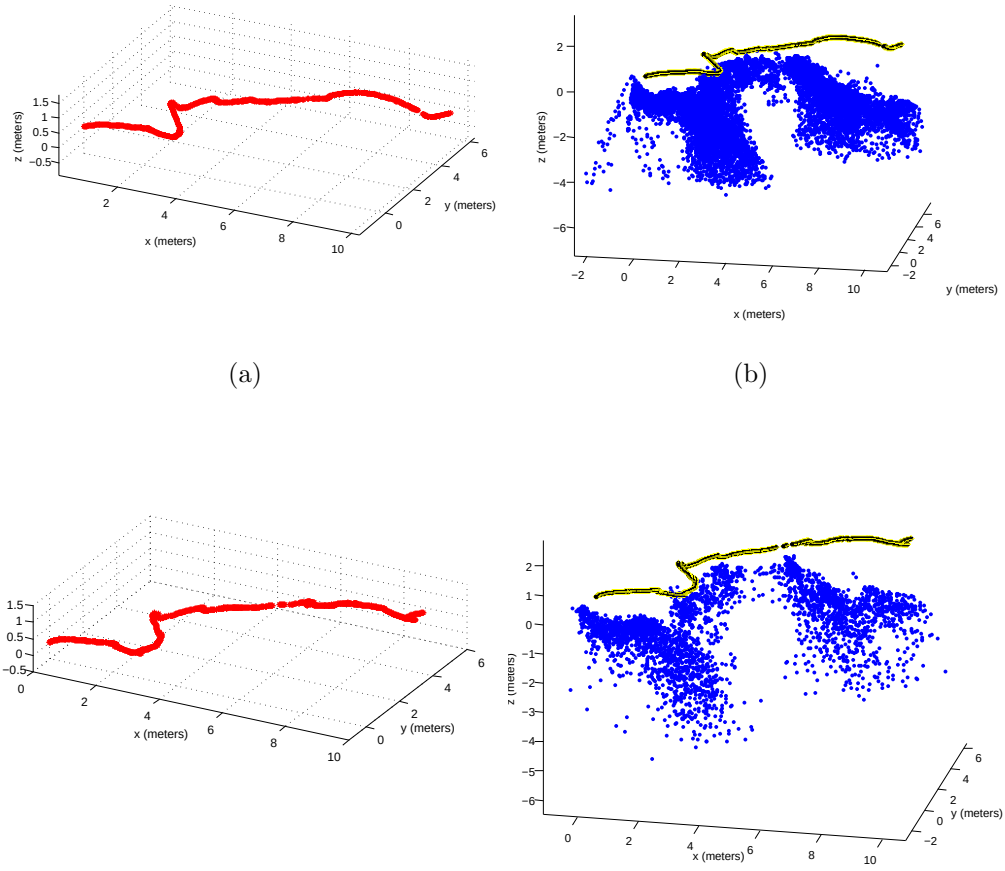


FIGURE 3.7. (a) Estimated trajectory for Dataset3, using SURF and Approximate Nearest Neighbor matching. (b) 3D coral structure, estimated via Eq. (2.58) along with the trajectory. (c) Estimated trajectory for Dataset3, using Shi-Tomasi features and ZNCC matching. (d) 3D coral structure

CHAPTER 4

EXPERIMENTAL EVALUATION

In this chapter we are going to validate the system that has been presented so far in its entirety, and present results that show correct implementation of the algorithm. Thus, this will open up the possibility of future work that enhances different components of the system so as to achieve the goal of real-time operation at 5-10 camera frames per second on the computing hardware of the robot. We present experimental results and the parameters that enabled them, both for a simulated world and for two real-world underwater datasets.

4.1. Simulation

In the simulation scenario we compose synthetic data and provide it as an input to the algorithm. The trajectory of the robot is executed along a quarter-circular arc, 2 meters above the seabed, which is simulated as a flat surface, as shown in Fig. 4.1(a). The field of view of the simulated camera is about 35°. The depth-based update is not applied in these experiments, as they are meant mostly to explore the performance of the algorithm as the noise of the IMU and camera data increases. The exact trajectory is generated by:

$$\mathbf{p}(t) = [R\cos(\omega t) \quad R\sin(\omega t) \quad Z] \tag{4.1}$$

$$\mathbf{q}(t) = [0 \quad 0 \quad \sin(\omega t) \quad \cos(\omega t)] \tag{4.2}$$

where $R = 10m$, $Z = 2m$, $\omega = \frac{\pi}{60} rad/s$ and $t \in [0, 30]$, which means that we can differentiate to get the exact velocity as well. In the case where the sensory data provided to the algorithm is noise-free, it is able to reconstruct the trajectory without errors, as shown in Fig. 4.1(c). We want to see how the estimation error

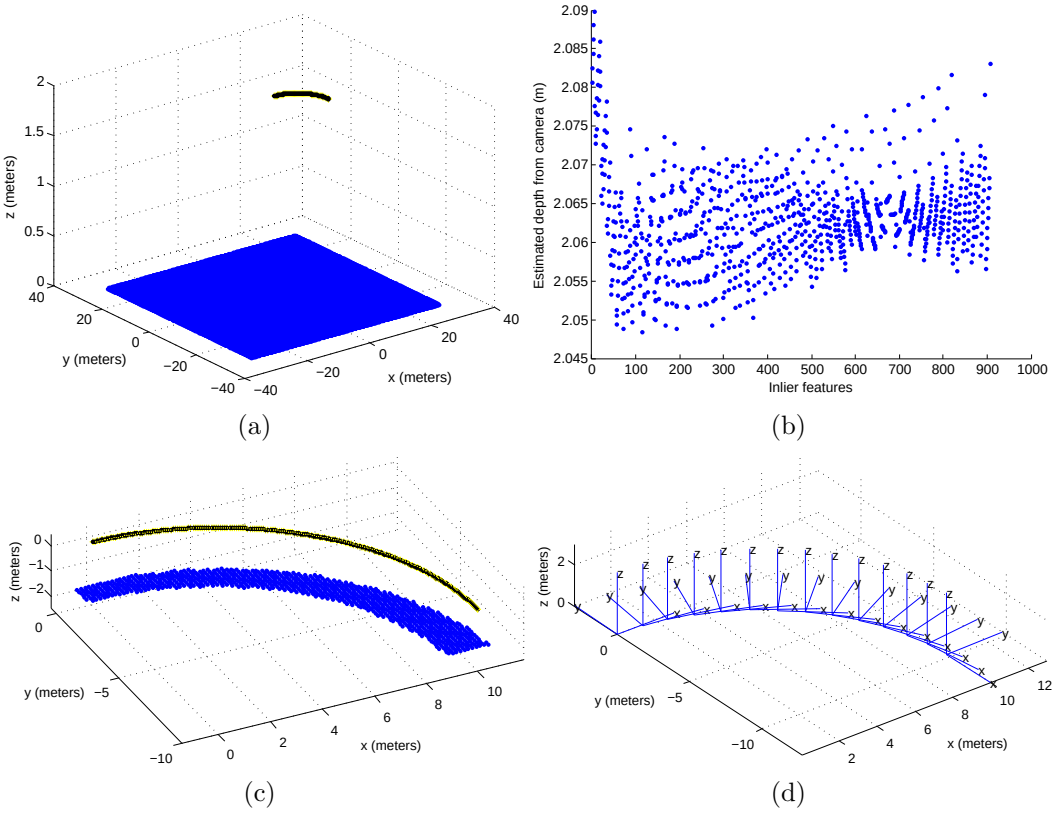


FIGURE 4.1. The output of the algorithm with noise-free measurements. (a) Simulated noise-free trajectory and world (b) Estimated depth of features from the camera. Real depth: 2.092m (c) Estimated trajectory and 3D structure (d) Estimated orientation of the IMU frame

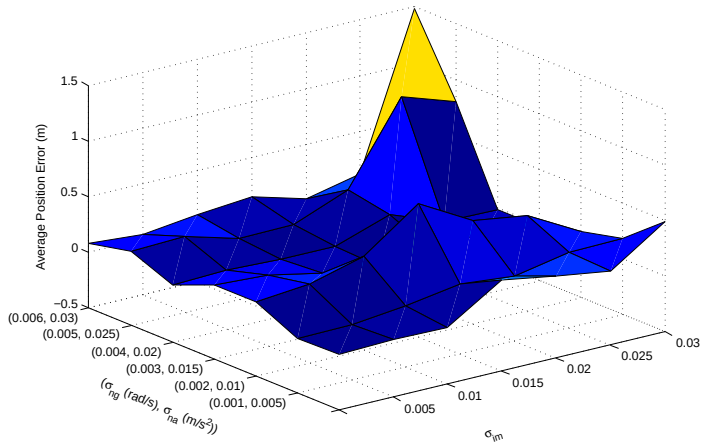
of the algorithm grows as a function of increasing IMU and camera noise. To this end we ran multiple simulations with different noise parameters to get an empirical evaluation of the estimation error. In particular, for position and velocity we use the Euclidean norm, while for quaternion differences, we use the angle in Eq. (2.26).

Figure 4.2 illustrates that when both the IMU and the camera noise are low the estimation error is low for all the quantities of interest. In the presence of IMU noise

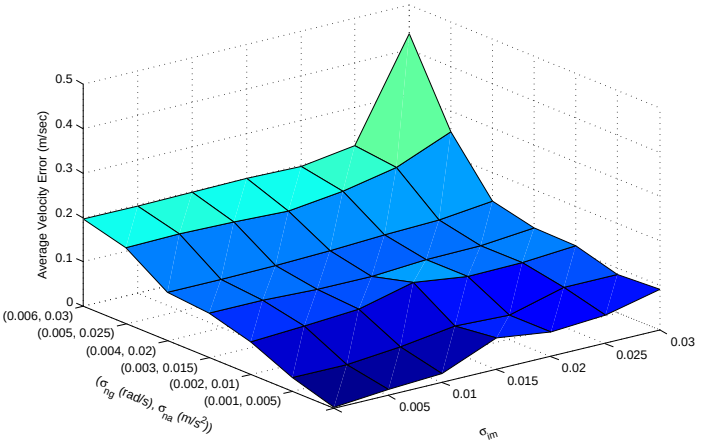
only, the velocity estimate seems to be affected in the same way as the position estimate (similar error gradient), which suggests that near-perfect camera measurements are able to correct the position as well as the velocity. In the presence of camera noise only, the position estimate degrades faster than the velocity estimate. When both IMU and camera noise are present, the rate of velocity degradation is again lower than that of the position.

In the real system the measured gyroscope noise has standard deviation 0.006 rad/sec, the measured accelerometer noise has standard deviation $0.06m/s^2$, and we expect an error of around 3 pixels over 900 pixels of the total field of view of the camera, due to calibration and keypoint localization errors. This simulation shows that if we assume the camera has been calibrated well, the camera to IMU transformation is accurate, and feature matching is perfect, then in this sort of trajectory we should expect position errors of at least 1.5 meters (over 15 meters in total, i.e. approximately 10%), velocity errors around 0.5 meters/sec, and orientation errors of about 2 degrees.

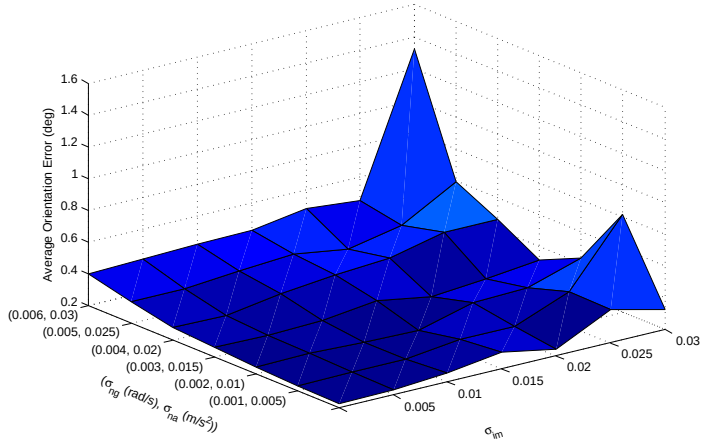
Aside from the expected error, another lesson learned from the simulations when the camera and IMU noise levels are set to realistic levels is that the filter covariance may underestimate the errors. This is shown in Fig. 4.3 which indicates that the Extended Kalman Filter presented here is an inconsistent state estimator. Measures for dealing with this inconsistency have been presented in Huang, Mourikis and Roumeliotis [**37**], Julier and Uhlmann [**44**], and the issue has been studied in other works, such as [**39, 41, 38**]. For the purposes of the present system it is left as future work.



(a) Average position error



(b) Average velocity error



(c) Average orientation error

FIGURE 4.2. The sensitivity of the algorithm to sensor noise. Camera noise with standard deviation σ_{im} is injected in the perspective projection to account for the innacuracies of the pinhole camera model. IMU noise is injected in the acceleration and angular velocity readings of the simulated IMU, with standard deviations σ_{na} and σ_{ng} respectively.

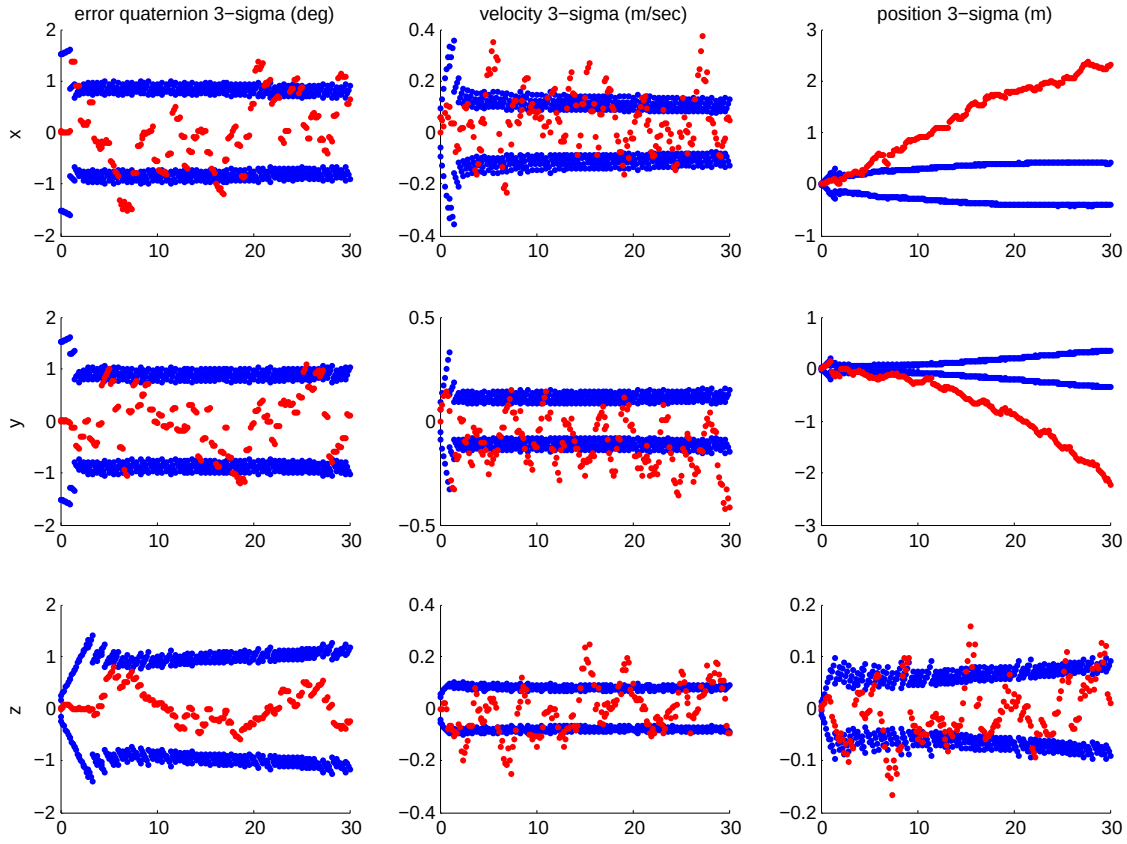


FIGURE 4.3. 3-sigma covariance bounds for the orientation, velocity and position of the simulated trajectory. In this experiment, $\sigma_{ng} = 0.006$, $\sigma_{na} = 0.03$ and $\sigma_{im} = 0.03$. The position covariance particularly underestimates the error.

4.1.1. Error propagation from the 3D feature estimation to the velocity estimate. We wanted to get a picture of how a single "pulse" of error injected in the camera measurements, would affect the estimates of the velocity. In other words, assuming that before and after the injection of this pulse of noise, the IMU and camera data were noise-free, would the velocity estimates converge to the correct values? Fig. 4.4 shows that the answer is no; the velocity estimates converge, but not necessarily to the correct values. In this simulation the pulse of camera noise with standard deviation $\sigma_{im} = 0.03$ is applied during $t \in [2, 5]$ seconds, which is why the velocity errors shown are zero in the first 2 seconds.

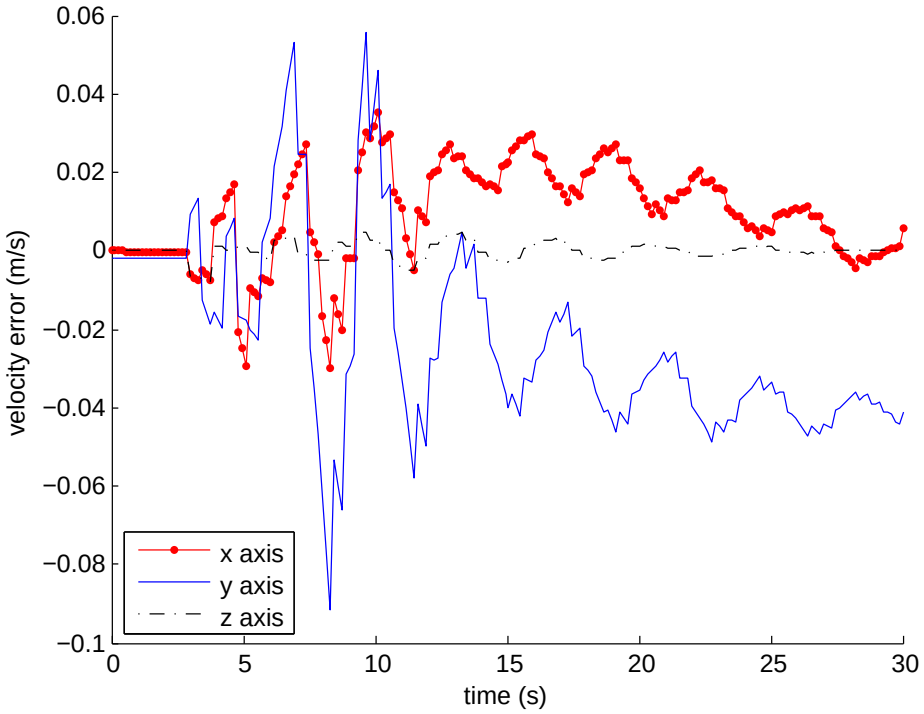


FIGURE 4.4. Velocity errors of an almost noise-free trajectory, with the exception of camera noise during $t \in [2, 5]$ seconds. Notice that the velocity error in y does not converge to 0, despite the noise-free measurements following the error pulse.

In the same vein, we wanted to see whether the velocity estimates are susceptible to feedback of error, in other words, whether a constant level of error on the velocity

gives rise, through the 3D feature estimation procedure, to increasing velocity errors. Fig. 4.5 shows that this does indeed happen, and with time, the error on the velocity accumulates in such a way that even perfect sensory inputs are not enough to keep it constant. For this simulation we added random noise with standard deviation of 0.02m/s to the velocity estimate, during $t \in [2, 30]$, while the IMU and the camera noise were set to zero.

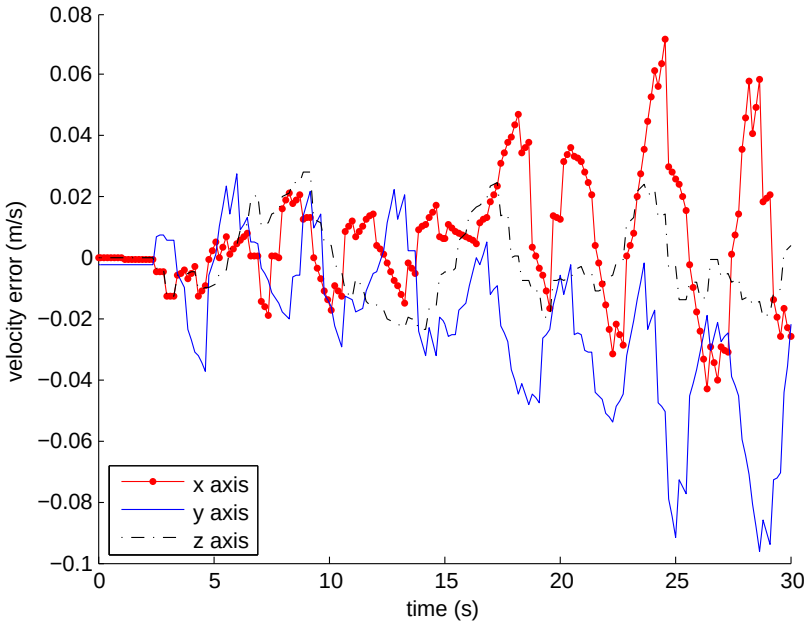


FIGURE 4.5. Velocity errors feed back via the 3D feature position estimation procedure. Camera noise was added during $t \in [2, 30]$ seconds.

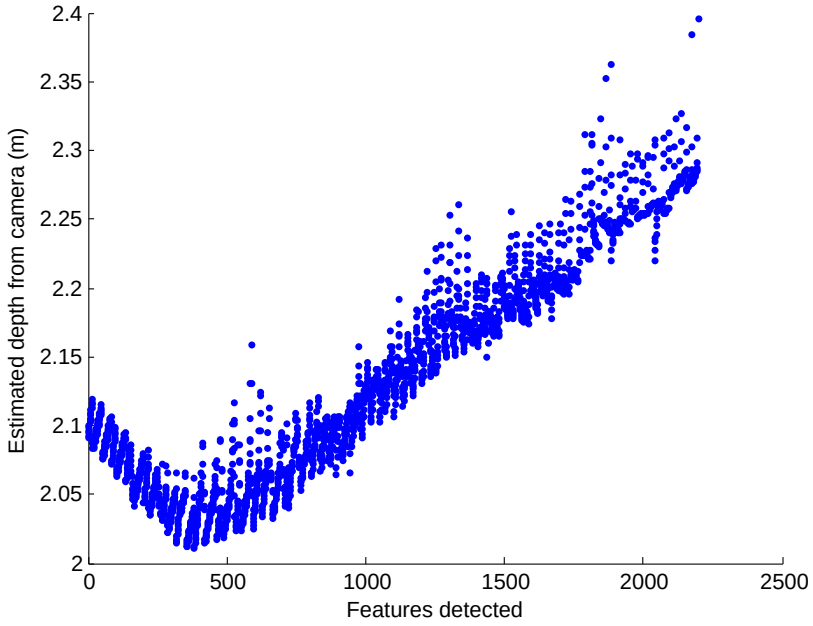


FIGURE 4.6. Misestimation of the depth of features from the camera. Increasing velocity gives rise to features that are deemed to be farther away.

This suggests that in the case of a velocity increase that is unnatural given the physical constraints of the robot (e.g. forward speed more than 1m/s) to mitigate the effect of this accumulation and force a speed reduction without directly modifying the state vector of the EKF, one could intervene at the 3D feature position estimation procedure to bring the features closer to the camera, i.e. reduce their depth by an appropriate percentage, which would lower the estimated speed of the robot.

4.2. Underwater Robot

The target underwater vehicle for this system is the Aqua2 family of underwater robots (see Fig. 4.7). These hexapod robots have both walking and swimming capabilities and are able to execute complex trajectories. They are capable of non-holonomic 6-degree-of-freedom motion and generally very agile and lightweight. More specifically, they include two embedded computers, one for vision processing (vision stack) and one for low-level control operations (control stack), both conforming to

the PC/104 PLUS form factor. The vision stack features a Pentium-M processor at 1.6GHz, with 1GB RAM, running a minimalistic version of the Ubuntu operating system and ROS (Robot Operating System) [**29**] as the sensor abstraction and application layer. Three IEEE-1394 IIDC Firewire cameras are connected to the vision stack, two high-resolution camera (1024×768), and one low-resolution (640×480). The control stack on the other hand is computationally more limited. It features a Geode processor running at 300MHz, with 256MB of RAM. The QNX real-time operating system is running on the control stack, and on top of that there is the RoboDevel library, which provides a hardware abstraction layer, motion primitives for the swimming, walking and other operation modes of the robot, as well as more complex walking gaits. A pressure sensor and a MicroStrain 3DM-GX1 IMU are connected to the control stack because their data are used for roll and depth stabilization. That said, the two stacks are able to exchange data or commands with each other, so from the vision stack we can programmatically direct the robot via a simple API. It is worth mentioning that the downward looking camera (enabled by the mirror setup presented in Fig. 2.4) is at the back of the robot, about $50cm$ away from the IMU, which is closer to the front. So, the translation between the camera frame and the IMU frame is significant.

The state estimation system described in this thesis runs on the vision stack of the robot, and is implemented in C++ as collection of ROS nodes with Eigen3 (and occasional patches of LAPACK) as the underlying numerical computation library. The robot can operate in tethered (connected via fiber-optic cable to a laptop on the surface) or tetherless mode (divers program the robot underwater via tags), and it has Lithium-Ion batteries capable of delivering more than 5 hours of operating time underwater. The mass of the robot is around 17kg and its body is neutrally buoyant. The maximum forward speed of the robot is around $1m/s$.

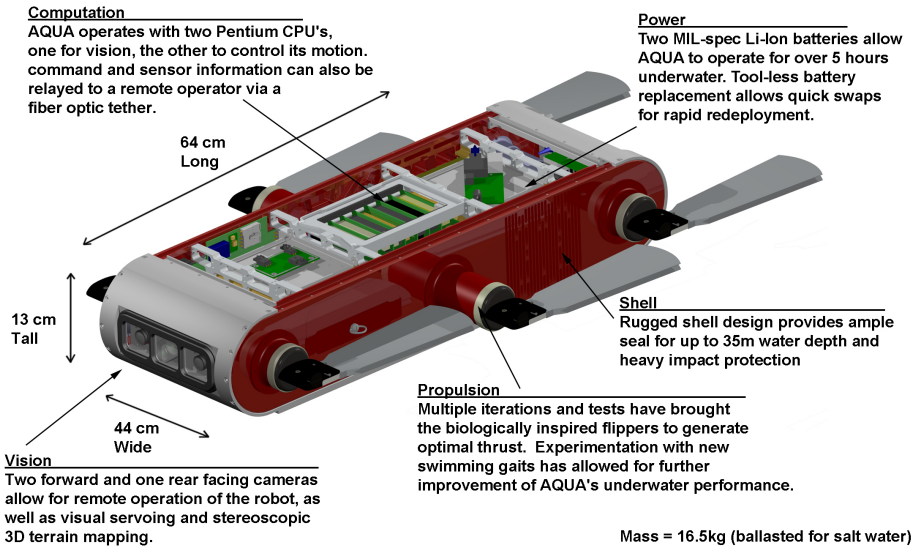


FIGURE 4.7. A view of the design of the Aqua2 family of underwater robots.

4.3. Experiments on underwater datasets

To test the validity of our adaptation of the algorithm in real-world data, we collected underwater datasets with ground truth from the waters of the island of Barbados, and we performed offline experiments to test our implementation. Two of these datasets are going to be presented below together with the state estimates. Images on both datasets were recorded from the back camera at 15Hz, and a 640×480 subregion of the image was used in the experiments. The IMU data is coming from a MicroStrain 3DM-GX1 MEMS unit, sampled at 50Hz. The first dataset, depicted in Fig. 4.8, features a straight 30 meter-long trajectory, where the robot moves at approximately 0.2 meters/sec forward, while preserving its depth. The sea bottom is mostly flat, and the robot moves about 2 meters over it. A white 30 meter-long tape has been placed on the bottom, both to provide ground truth for the distance traveled, and to facilitate the robot's guided straight line trajectory.

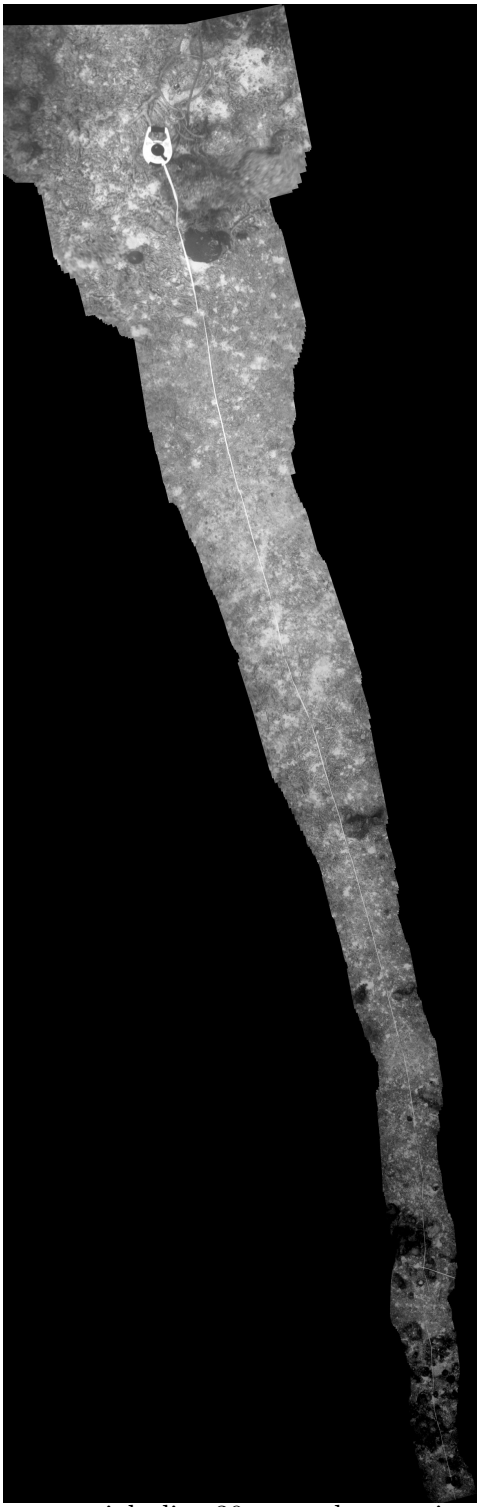


FIGURE 4.8. The near-straight line 30 meter-long trajectory executed in the first experiment. The increase of the field of view of the robot is an artifact of the image stitching software.

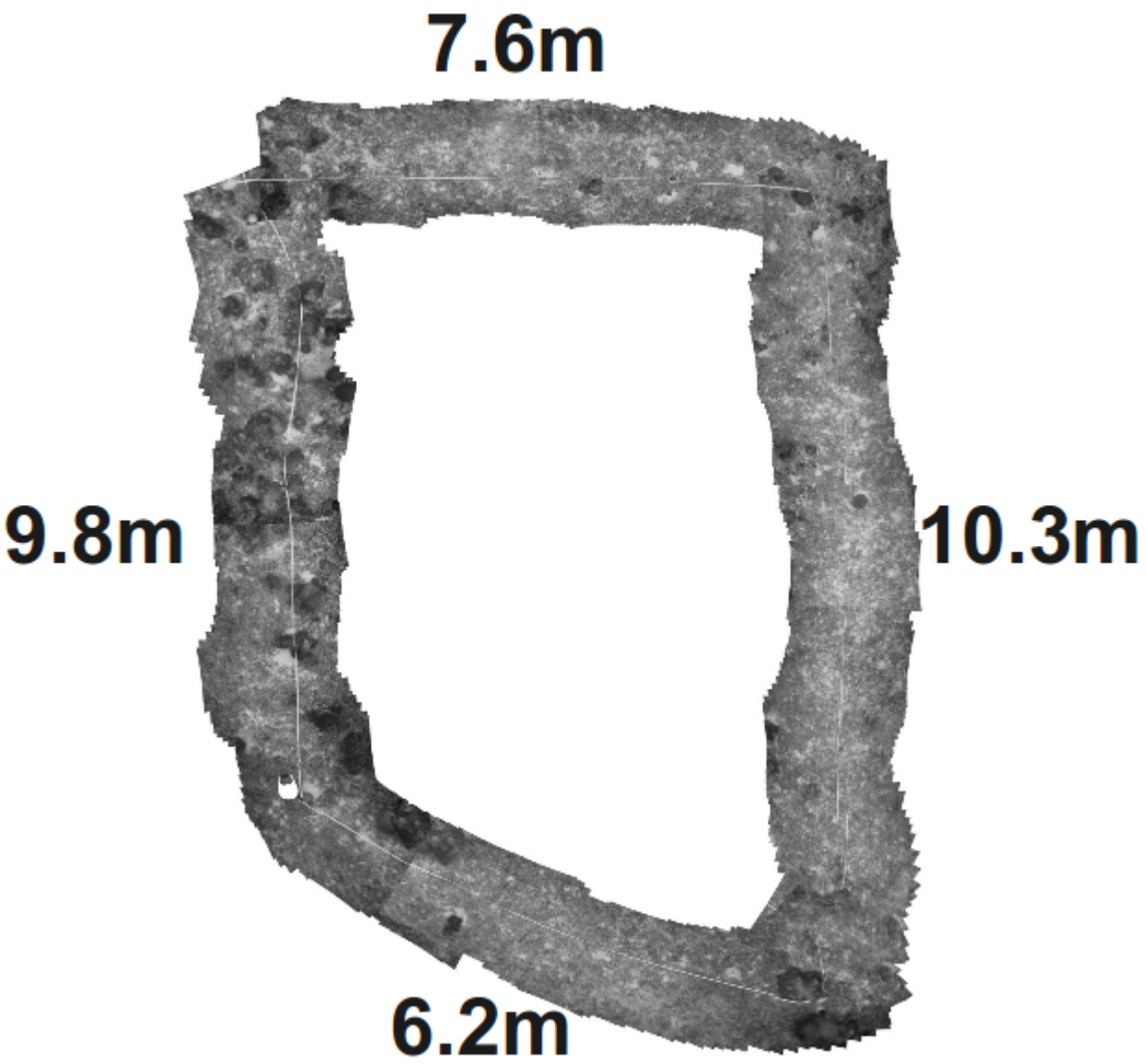


FIGURE 4.9. The second experiment, where the robot is guided to perform a loop closure while recording IMU data and images. In this case images have been stitched using image stitching software operating in a semi-autonomous fashion, where the user manually rejects poor stitches.

The second dataset corresponds to an experiment that took place over the same site, and under the same conditions mentioned above, however the shape of the trajectory was a closed loop, as depicted in Fig. 4.9. The total length was approximately

33 meters, and the side lengths are shown on the figure. In both cases the robot only performed data collection, and the datasets were post-processed offline.

4.4. Discussion of results

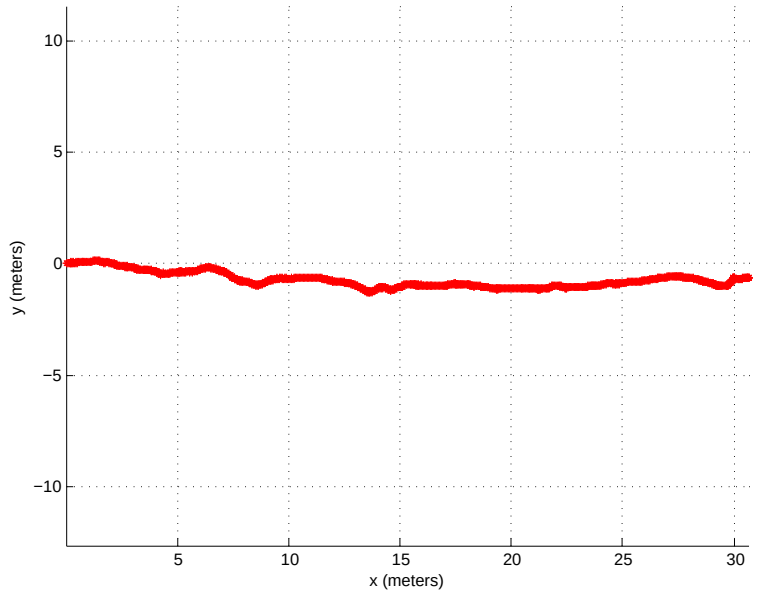
As seen in the top view of the estimated trajectories in Fig. 4.10 the length of the straight line was estimated correctly, with a 1 meter error, while the loop trajectory accumulated more error. In particular, the horizontal segments of the loop are estimated correctly with error below 0.5 meter, while the vertical segments have been overshot by 2 meters. The fact that the straight line was estimated correctly, while the loop segment lengths are overshot, indicates one or more of the following, ordered from most to least likely:

(a) The 3 turns on the loop trajectory are too sharp, or are such that there is only rotation in place without much translation. This means that the 3D estimates of the tracked features might contain larger errors than usual, because the baseline is too small. This is actually confirmed by Fig. 4.13, specifically, by the presence of the 3 spikes in the estimated depths of features from the camera. The spikes are unnatural in the sense that there cannot be a sudden change of depth in the environment, which is assumed to be more or less continuous. This issue might be fixable if handled as a special case, where, if the baseline of consecutive occurrences of a features is too small, then the feature should not be taken into account. Currently, this is not handled in the code.

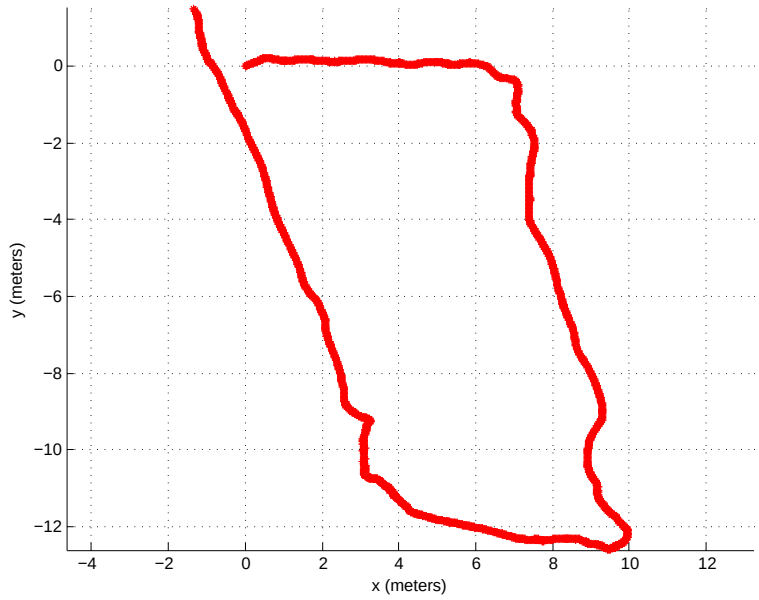
(b) Inaccurate modeling of the gyroscope noise and bias. The standard deviation of the gyroscope noise was measured when the robot was still and the bias is too small a quantity to cause the length errors we have observed.

(c) Camera calibration errors, which affect the undistortion process.

(d) Inaccurate initial conditions for the state and the covariance. This is less likely due to the correct estimation of the first segment.



(a)



(b)

FIGURE 4.10. (a) Top view of the estimated trajectory for the straight line. (b) Top view of the estimated trajectory for the loop.

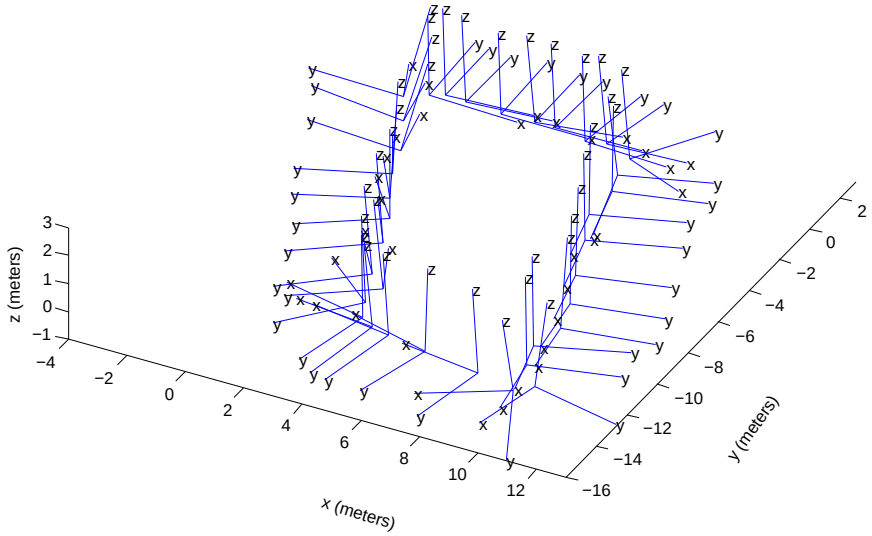
Reason (a) is most likely because it explains another issue: in the feature depth estimates for the loop (Fig. 4.13(a)) we would expect that if the depth sensor indicates

no change of depth, then the depth of features seen in the beginning of the loop trajectory should be the same as that seen at the end. This is obviously not the case at the moment. If (a) is indeed the root of the problem, then it would cause an error in velocity after each of the turns, which would in turn cause an error in the baselines of consecutive camera frames, and thus in the estimated feature depths.

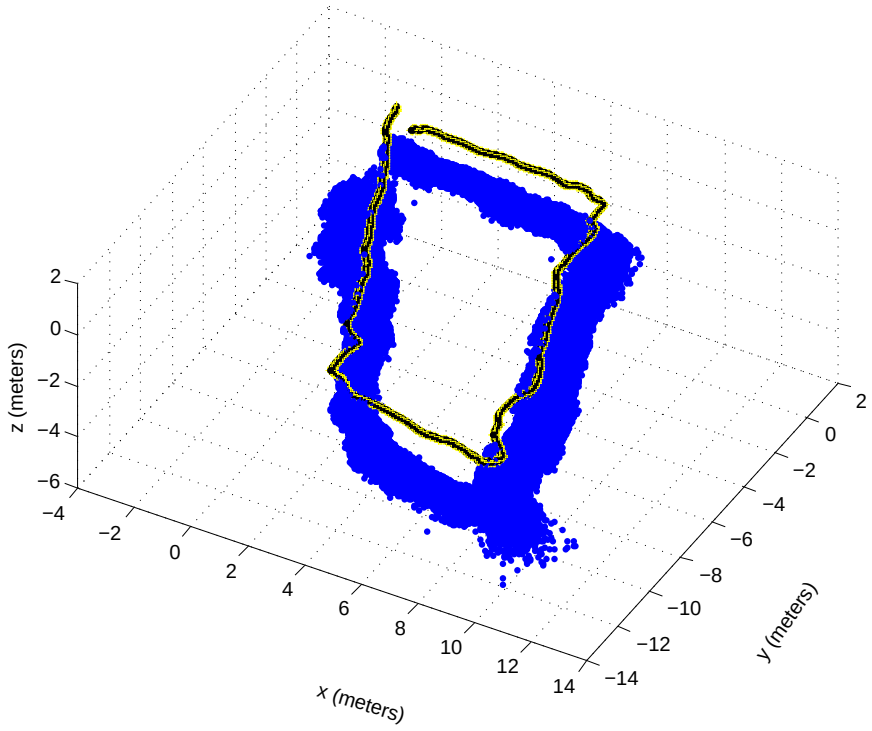
The orientation of the IMU frame is shown in Fig. 4.11(a) and 4.12(a), and it appears to be estimated correctly, thought we have no effective way to compare it against ground truth.

The parameters used for the loop and straight line experiments are shown in Table A.1 in the Appendix.

To summarize, the experiments have shown that the algorithm is accurate on straight lines, and generally accurate on trajectories with turns, but faces problems on sharp turns, especially when they are performed in place and with reduced speed, which causes the baselines of the camera frames from which the features are seen to be very small. We also noticed that the EKF tends to underestimate the errors of the state, particularly for the position and less so for the velocity and the orientation. We also demonstrated through simulation that velocity errors feed accumulate and feed back via the 3D feature position estimation procedure, which is a potential cause of divergence if the errors are high.

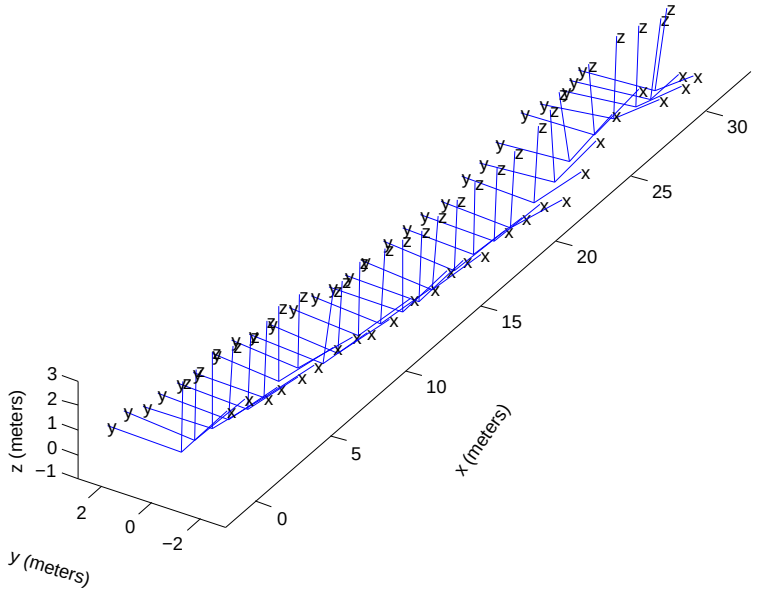


(a)

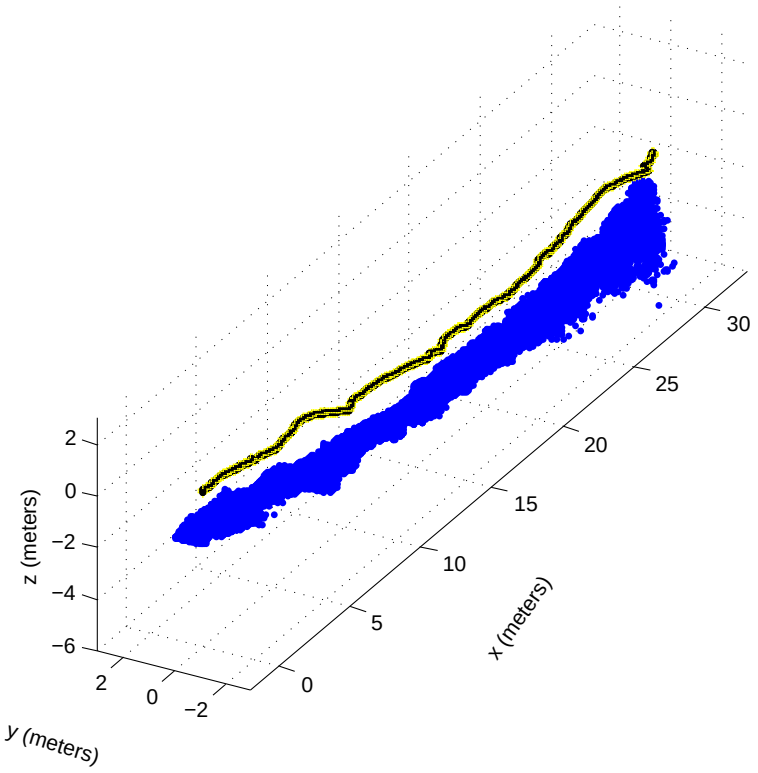


(b)

FIGURE 4.11. (a) Estimated IMU frame trajectory for the loop (b) Estimated trajectory and 3D structure for the loop

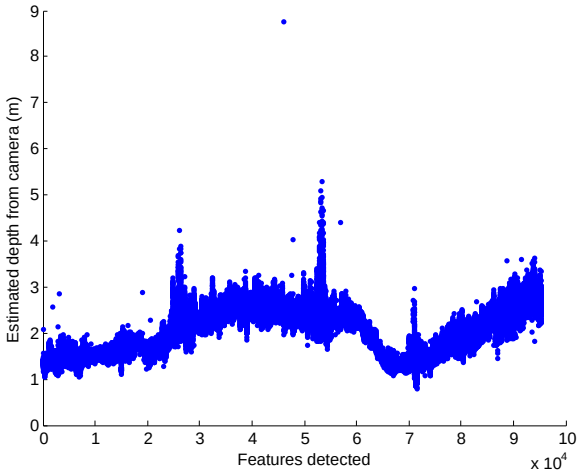


(a)

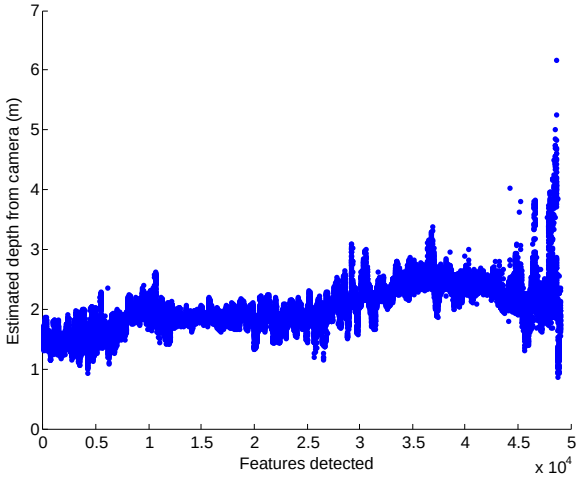


(b)

FIGURE 4.12. (a) Estimated IMU frame trajectory for the straight line (b) Estimated trajectory and 3D structure for the straight line

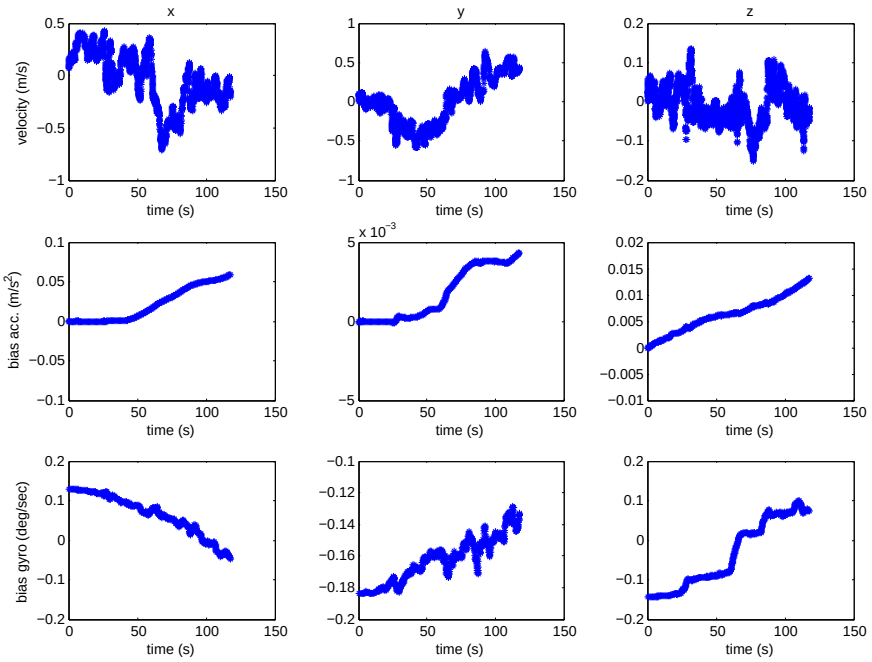


(a)

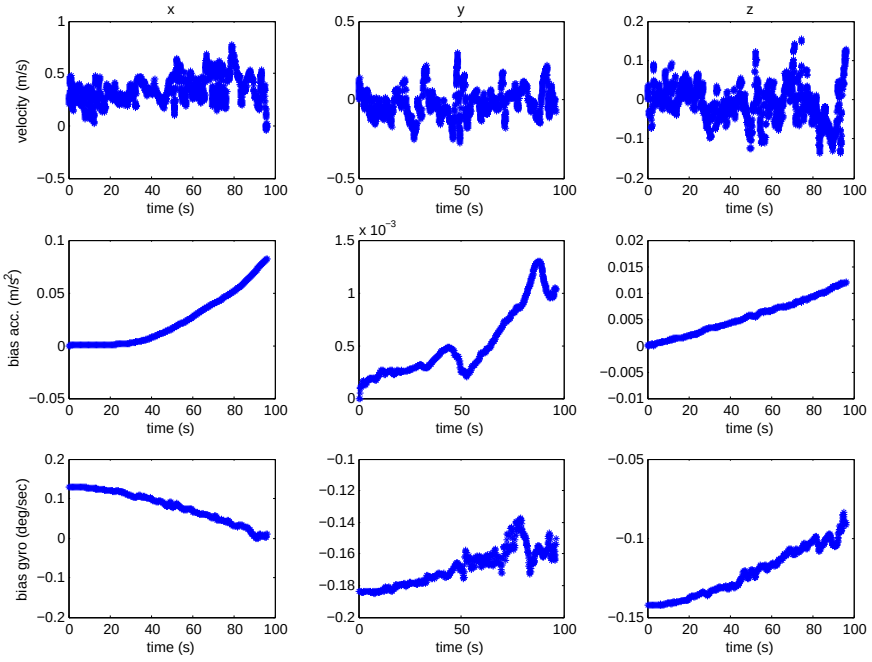


(b)

FIGURE 4.13. (a) Estimated depth of features for the loop (b) Estimated depth of features for the straight line

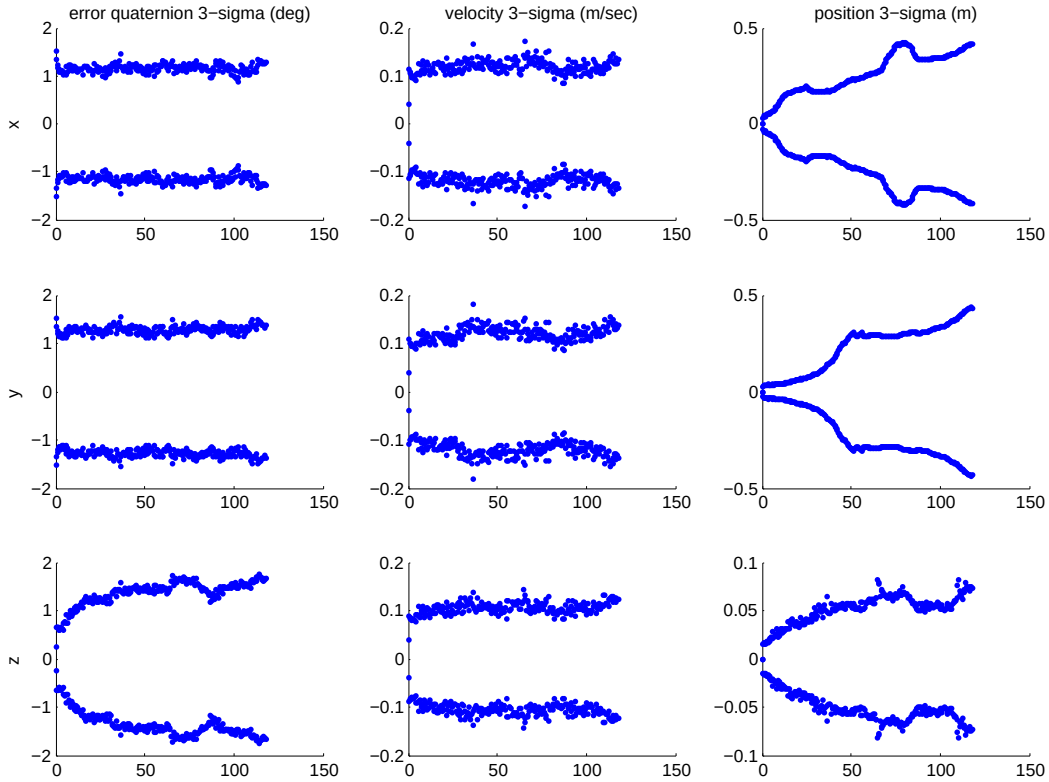


(a)

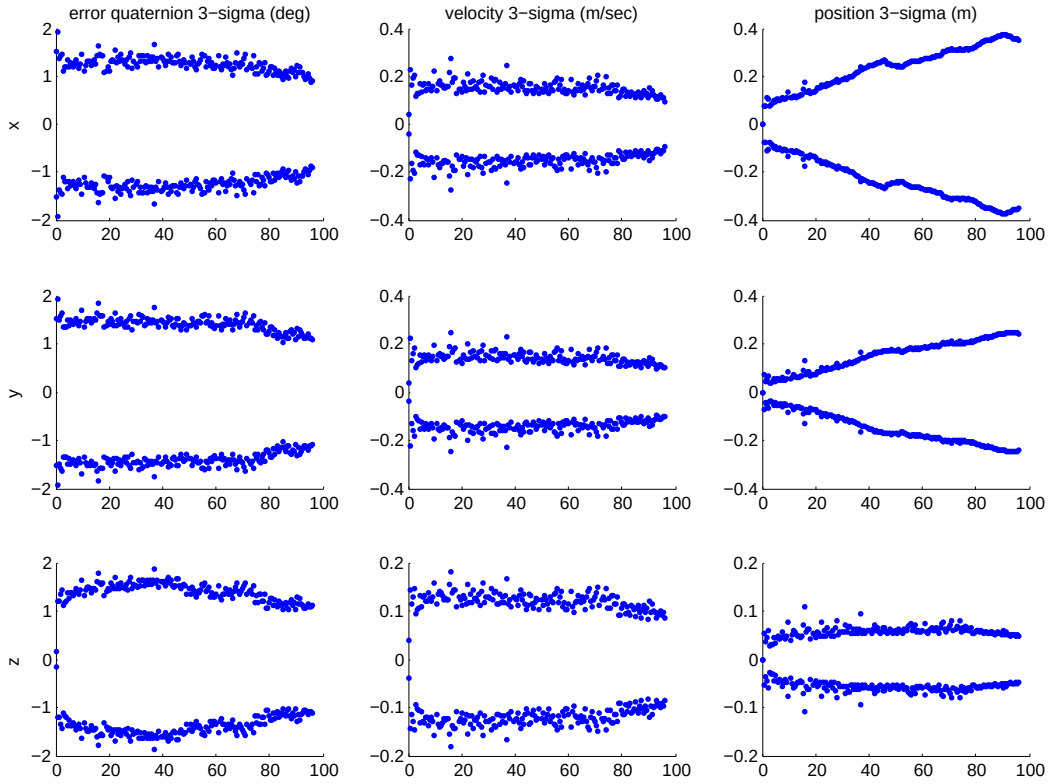


(b)

FIGURE 4.14. (a) Estimated velocity and biases for the loop (b) Estimated velocity and biases for the straight line. Underwater experiments performed both in the pool and in the ocean suggest that the maximum forward speed of the robot is about 1m/s.



(a)



(b)

FIGURE 4.15. (a) Estimated 3-sigma error bounds for the loop (b) Estimated 3-sigma error bounds for the straight line

4.5. Time synchronization between camera and IMU

As mentioned previously, camera measurements are made from the vision CPU of the robot, while IMU measurements are made from the control stack. This allows for errors due to lack of exact time-synchronization between the camera and the IMU. That said, even if the two sensors were to be connected to the same computer, the issue would still be there, because the camera has a higher delay in preparing the entire message compared to the IMU. In order to get a rough estimate of this time delay between the camera and the IMU messages, we performed the following experiment in the lab, while the robot was completely still: we gently hit the part of the robot containing the IMU, while the camera was recording the collision. We performed 13 such hits, and we recorded the maxima of the IMU accelerometer readings, and also the collision times as seen in the corresponding camera frames. The results, shown in figure 4.16 indicate that on average the camera is 45 milliseconds "slower" than the IMU, with standard deviation 0.39 milliseconds. The reason why this experiment is only going to provide a rough estimate is that the camera is recording at 15Hz, or one frame every 66 milliseconds. In other similar algorithms, such as Jones and Soatto [**42**], the time delay between their IMU and camera was reported to be 50 milliseconds. In other words, the standard deviation of these time delays is too big, and would become smaller if our camera was recording at 30Hz. Nevertheless, this simple experiment illustrates that a time delay between the camera and the IMU does exist, however a more precise experiment is needed.

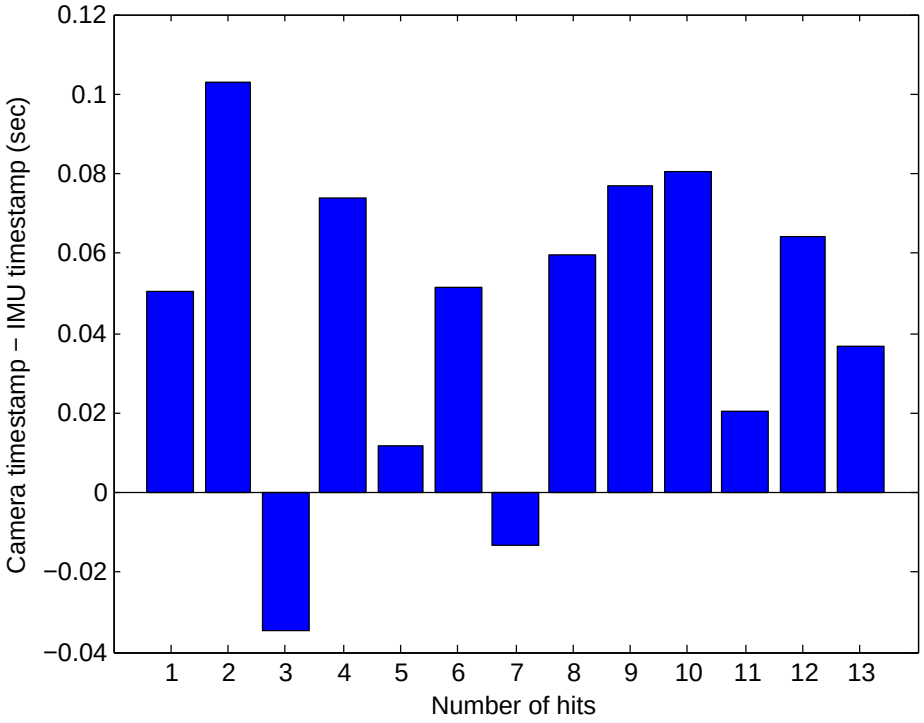


FIGURE 4.16. Time delay between camera and IMU timestamps

Dealing with these time delays between sensors usually involves common hardware triggers, which poll the IMU and the camera at regular intervals. If that is not possible, then a common heuristic is to simply subtract the average time delay from the camera timestamps, but that does not provide exact synchronization.

CHAPTER 5

CONCLUSIONS

5.1. Summary

In this thesis we presented in full detail the adaptation of an existing pose estimation algorithm for use by an underwater robot in shallow, feature-rich environments. An Extended Kalman Filter that incorporates measurements from a camera, an IMU, and a depth sensor was presented and validated in simulated and real-world environments. While real-time operation of pose estimation algorithms is usually a requirement, in our current implementation we have opted to be concerned more with accuracy than speed of operation. Our results show that visual and inertial measurements are a viable option for pose estimation techniques, as long as careful calibration and synchronization between the two sensors is performed. The algorithm examined in this thesis is extendable in the sense that it can easily incorporate input from two or more cameras, or some stable features in the state vector.

5.2. Critique And Shortcomings

One of the main limitations of the algorithm is that the initial velocity of the robot has to be zero or otherwise known if the algorithm is started while the robot is in motion. Also, the initial acceleration has to be zero, so that the estimate of the initial orientation is done correctly. Unfortunately, in practice, these two conditions are not easy to guarantee, especially in the underwater scenario. That said, recent

results have addressed both of these issues [**60, 61**], in an effort to estimate the initial conditions of the trajectory.

Another limitation of this algorithm, and others like it, is the sheer number of parameters that need to be tuned to the particularities of the dataset on which estimation is performed. Some parameters, like the IMU noise characteristics, are considered to be constant across different environments and datasets, but others, like the number of features detected on each image, the initial covariance settings, and the IMU bias modeling might be different depending on the environment, the lighting conditions and the particularities of the dataset, such as the frame rate of the camera.

Finally, the underestimation of the error by the update of the covariance matrix is also a limitation, and it reduces our confidence in the provided estimates.

5.3. Potential Pitfalls

Vision-aided pose estimation algorithms that incorporate inertial measurements are prone to a number of sources of error, which might initially seem as unimportant implementation details, but they nevertheless have a significant impact on performance. Assuming a correct implementation, the following are some reasons behind the potential limited accuracy of this class of algorithms:

- Recorded images have motion blur.
- The environment does not have many salient features. For example, when the robot is moving over sandy regions of the seabed.
- The robot is moving very fast (there is little overlap between consecutive images).
- Gravity is not estimated accurately for the place where the experiments are going to take place.
- The fixed camera-to-IMU transformation is not accurate.
- Camera calibration, and joint camera-IMU calibration is not performed underwater. Accurate camera calibration underwater is of paramount importance to the algorithm's accuracy. This is because aside from the camera

lens, the glass of the camera's underwater housing, the water medium itself, and in our particular case, the mirror, cause distortions that facilitate calibration errors that are hard to model.

- Outlier rejection for both camera and IMU data is not done carefully.
- IMU and camera measurements are not time-synchronized.
- IMU measurements are very noisy (indicative of a low-end sensor, or a magnetically noisy environment).
- IMU and camera noise characteristics (covariance matrices) are not modeled correctly.

5.4. Opportunities For Future Work

There are many directions in which this work can be extended or improved. Some of them are related to software implementation issues, others propose significant modifications to the algorithm, and a couple highlight possibilities for different algorithms altogether.

5.4.1. Possibilities for speed-ups. One of the decisive issues in allowing this algorithm, and more generally vision-aided pose estimation algorithms, to run in real time is the speed at which feature tracking is performed from image to image. As mentioned previously, our current implementation relies on SURF keypoints matched by Approximate Nearest Neighbour matching, which is able to operate on images at a rate of 1-3Hz on the robot's vision CPU. The current implementation is therefore not considered real-time if our goal is to process images at 10-15Hz. At this point there are two different ideas that can be explored, one hardware-related, while the other already exists in software.

The first idea involves designing an FPGA (Field Programmable Gate Array) whose sole function will be the computation of SURF keypoints and descriptors given an input image. Barfoot [**9**] followed this approach for the extraction of SIFT keypoints and reported a 7-fold speed-up in the feature detection process (from 1 second

on a Pentium-M to 0.15 seconds on the FPGA). Of course, this opens up a new challenge: correctly implementing SURF on a hardware description language. Recently, progress has been made in this direction via FPGA-SURF [**90**], an open-source implementation of the SURF detector and descriptor on a Xilinx FPGA. The authors report processing rate of 10 frames per second for a 1024×768 camera, with 10W power consumption. This would definitely open up the road for real-time operation of the algorithm on the PC104 vision stack of the robot.

The second idea involves software and suggests tracking features from the FAST detector that have high Shi-Tomasi filter response, in order to obtain relatively stable features at high speed. This approach has already been implemented in PTAM (Parallel Tracking And Mapping) [**48**], and the processing rate of its feature tracking module was reported to be 50 frames per second. Although PTAM was designed with small, augmented reality workspaces in mind, careful outlier rejection can make its feature detection module very fast and robust.

5.4.2. Bundle adjustment. Unlike the filtering paradigm, where the Markovian assumption allows errors to be propagated from the previous to the current state, the principle behind bundle adjustment [**93**] is that the estimated state best explains the measurements that were obtained during a portion of the trajectory. More formally, bundle adjustment is the process of estimating both the camera trajectory and the 3D structure of the observed world, based on the feature trails that were seen. This involves solving an optimization problem where the objective function is a user-defined reprojection error, between the actual measurements and the expected measurements based on the variable, hypothetical camera trajectory. In fact, since this objective function involves quaternions for the representation of rotations, specialized numerical solvers have been developed to handle this minimization process, such as g^2o [**50**]. Numerical solutions of this problem require an initial estimate of the solution, which in our case can be provided by the propagation step of the IMU.

Performing bundle adjustment globally, on the entire trajectory, is an expensive operation, so in most cases bundle adjustment is done locally, on small portions of

the trajectory, without aiming for a globally consistent trajectory and 3D structure estimate. PTAM itself follows this approach, and other work too (e.g. Sibley [**83**], Kaess [**45**]). A possible advantage of bundle adjustment methods over the proposed algorithm would be that they do not necessarily take into account the velocity of the vehicle, and thus the possibility of a velocity error propagating forward in time and causing the estimate to diverge is reduced.

Therefore, a potential direction for future work would be to integrate bundle adjustment into the filter presented in this thesis, particularly on segments of the robot's trajectory that give rise to small camera frame baselines.

5.4.3. Unscented Kalman Filter. Another direction of work which we have not explored but seems very promising is the replacement of the EKF framework by an Unscented Kalman Filter (UKF). As mentioned in previous chapters, the UKF does not perform linearization of the nonlinear state transition and sensor models, which make it generally better in terms of accuracy compared to the EKF [**96**]. An interesting possibility would be to compare the UKF and the EKF in the same simulation scenarios that we considered here to get a measure of the difference in accuracy.

5.5. Final remarks

Performing underwater localization is a hard problem, however it is a challenge that is worthwhile to take up not only because of the technical difficulties, but also because of the potential effects it will have in the collection of data for marine biology. Underwater robot localization will be an important enabler of the automation of the exploration and monitoring of subsea ecosystems, processes that are currently performed by individual marine biologists, after a few hours of diving every day. These processes are tedious, very long, and give scientists what can at best be regarded as a very isolated view of the health of the ocean. On land, we have realized the value of collecting streams of data from our cities and environments a long time ago; it is time to start exploring that direction for the underwater domain as well.

REFERENCES

[1] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. CenSurE: center surround extremas for realtime feature detection and matching. In *ECCV (4)*, pages 102–115, 2008.

[2] Guillem Alenya, Elisa Martnez, and Carme Torras. Fusing visual and inertial sensing to recover robot ego-motion. *Journal of Robotic Systems*, 21(1):2332, 2004.

[3] K.J. Astrom. *Introduction to Stochastic Control Theory*. Dover Books on Engineering. Dover Publications, 2006.

[4] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Pittsburgh, PA, July 2002.

[5] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Pittsburgh, PA, July 2002.

[6] Itzhack Bar-itzhack. Optimum normalization of a computed quaternion of rotation. *IEEE Transactions on Aerospace and Electronic Systems*, AES-7:401–402, 1971.

[7] I.Y. Bar-Itzhack and Y. Oshman. Attitude determination from vector observations: Quaternion estimation. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-21(1):128 –136, jan. 1985.

[8] I.Y. Bar-Itzhack and J.K. Thienel. On the singularity in the estimation of the quaternion-of-rotation. In *Guidance, Navigation, and Control Conference and Exhibit, AIAA*, Monterrey, CA, August 2002.

[9] T. D Barfoot. Online visual motion estimation using FastSLAM with SIFT features. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 579 – 585, August 2005.

[10] Timothy Barfoot, James R. Forbes, and Paul T. Furgale. Pose estimation using linearized rotations and quaternion algebra. *Acta Astronautica*, 2010.

[11] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, page 404417, 2006.

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, page 404417, 2006.

[13] R. C. Bolles and M. A. Fischler. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM,*, 24:381–395, 1981.

[14] A. Censi, L. Iocchi, and G. Grisetti. Scan matching in the hough domain. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2739 – 2744, April 2005.

[15] A.B. Chatfield. *Fundamentals of high accuracy inertial navigation*. AIAA, 2005.

[16] Olivia Min Yee Chiu. A stability and control system for a hexapod underwater robot. Master's thesis, Department of Mechanical Engineering, McGill University, Montreal, Canada, 2008.

[17] Peter Corke. An inertial and visual sensing system for a small autonomous helicopter. *Journal of Robotic Systems*, 21(2):4351, 2004.

[18] Mark Cummins and Paul Newman. Highly scalable Appearance-Only SLAM FAB-MAP 2.0. In *Robotics Science and Systems (RSS)*, Seattle, USA, June 2009.

[19] Andrew Davison, Ian Reid, N. Molton, and O. Stasse. MonoSLAM: Real-Time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):10521067, June 2007.

[20] Bar-Itzhack I. Deutschmann, J. and K. Galal. Quaternion normalization in spacecraft attitude determination. In *AIAA/AAS Astrodynamics Specialist Conference*, 1992.

[21] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors, 2006.

[22] David D. Diel, Paul DeBitetto, and Seth Teller. Epipolar constraints for Vision-Aided inertial navigation. *Proc. IEEE Motion and Video Computing*, pages 221–228, 2005.

[23] Gamini Dissanayake, Hugh F. Durrant-whyte, and Tim Bailey. A computationally efficient solution to the simultaneous localisation and map building (slam) problem. In *International Conference on Robotics and Automation*, pages 1009–1014, 2000.

[24] Gregory Dudek, Philippe Giguere, Chris Prahacs, Shane Saunderson, Junaed Sattar, Luz-Abril Torres-Mendez, Michael Jenkin, Andrew German, Andrew Hogue, Arlene Ripsman, Jim Zacher, Evangelos Milios, Hui Liu, Pifu Zhang, Martin Buehler, and Christina Georgiades. Aqua: An amphibious autonomous robot. *IEEE Computer Magazine*, 40(1):46–53, January 2007.

[25] Gregory Dudek and Michael R. M. Jenkin. *Computational principles of mobile robotics*. Cambridge University Press, 2010.

[26] D. Eberly. Rotation representations and performance issues, 2008.

[27] Ryan Eustice, Hanumant Singh, John Leonard, Matthew Walter, and Robert Ballard. Visually navigating the RMS titanic with SLAM information filters. In *Robotics Science and Systems*, June 2005.

[28] B. Ferris, D. Haehnel, and D. Fox. Gaussian processes for signal Strength-Based location estimation. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[29] Willow Garage. Robot operating system. http://www.ros.org.

[30] P. Gemeiner, P. Einramhof, and M. Vincze. Simultaneous motion and structure estimation by fusion of inertial and vision data. *The International Journal of Robotics Research*, 26(6):591, 2007.

[31] Christina Georgiades. *Simulation and control of an underwater hexapod robot*. PhD thesis, McGill University, Montreal, Canada, 2005.

[32] F. Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3:29–48, 1998.

[33] Marco Grimm and Rolf-Rainer Grigat. Real-Time hybrid pose estimation from vision and inertial data. *Computer and Robot Vision, Canadian Conference*, 0:480486, 2004.

[34] R. I Hartley and A. Zisserman. Multiple view geometry in computer vision. pages 245 – 246. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[35] C. Hertzberg, R. Wagner, O. Birbach, T. Hammer, and U. Frese. Experiences in building a visual slam system from open source components. In *IEEE International Conference on Robotics and Automation*, Shanghai, China, April 2011.

[36] H. Hopf. Systeme symmetrischer bilinearformen und euklidische modelle der projektiven raüme. *Vjschr. naturf. Ges. Zurich*, 85:165–177, 1940.

[37] G. Huang, A. I. Mourikis, and S. I. Roumeliotis. Observability-based rules for designing consistent EKF SLAM estimators. *International Journal of Robotics Research*, 29(5):502–528, April 2010.

[38] G.P. Huang, A.I. Mourikis, and S.I. Roumeliotis. Genearalized analysis and improvement of the consistency for ekf-based slam. 2008.

[39] S. Huang and G. Dissanayake. Convergence and consistency analysis for extended kalman filter based slam. *Robotics, IEEE Transactions on*, 23(5):1036–1049, 2007.

[40] Andreas Huster, Stephen M. Rock, Stephen P. Boyd, and Oussama Khatib. *Relative Position Sensing by Fusing Monocular Vision and Inertial Rate Sensors*. 2003.

[41] J. Neira J. A. Castellanos and J. D. Tardós. In *5th IFAC Symp. on Intelligent Autonomous Vehicles, IAV'04*, Lisbon, Portugal, Lisbon, Portugal 2004.

[42] E. Jones and S. Soatto. Visual-Inertial navigation, mapping and localization: A scalable Real-Time causal approach. *International Journal of Robotics Research*, October 2010.

[43] S.J. Julier and J.J. LaViola. On kalman filtering with nonlinear equality constraints. *Signal Processing, IEEE Transactions on*, 55(6):2774 –2784, june 2007.

[44] S.J. Julier and J.K. Uhlmann. Simultaneous localisation and map building using split covariance intersection. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 3, pages 1257–1262. IEEE, 2001.

[45] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365–1378, 2008.

[46] Jonathan Kelly and Gaurav S. Sukhatme. Visual-Inertial sensor fusion: Localization, mapping and Sensor-to-Sensor Self-Calibration. *International Journal of Robotics Research*, 30(1):5679, 2011.

[47] A. Kim and M.F. Golnaraghi. Initial calibration of an inertial measurement unit using an optical position tracking system. In *Position Location and Navigation Symposium, 2004. PLANS 2004*, pages 96–101, april 2004.

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

[49] Kurt Konolige, Motilal Agrawal, and Joan Sol. Large scale visual odometry for rough terrain. In *Proc. International Symposium on Research in Robotics (ISRR)*, November 2007.

[50] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation*, Shanghai, China, April 2011.

[51] A. M. Ladd, Kostas E. Bekris, G. Marceau, A. Rudys, L. E. Kavraki, and D. S. Wallach. Using wireless ethernet for localization. In *Proceedings of the 2002 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS 2002)*, volume 1, pages 402–408, Lausanne, Switzerland, 30 Sept. - 5 Oct 2002. IEEE Press, IEEE Press.

[52] Michael F. Land. Optics of the eyes of marine animals. In P. J. Herring, A. K. Campbell, M. Whitfield, and L. Maddock, editors, *Light and life in the sea*, page 149166. Cambridge University Press, Cambridge, UK, 1990.

[53] Jack Langelaan and Steve Rock. Navigation of small UAVs operating in forests. In *in Proc. AIAA Guidance, Navigation, and Control Conf*, page 468473, 2004.

[54] E. J. Lefferts, F. L. Markley, and M. D. Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance Control and Dynamics*, 5:417–429, 1982.

[55] Jorge Lobo and Jorge Dias. Inervis calibration instructions. `http://www2.deec.uc.pt/~jlobo/InerVis_WebIndex/InerVis_How_To/InerVis_Toolbox_example1.html`.

[56] Jorge Lobo and Jorge Dias. Relative pose calibration between visual and inertial sensors. *International Journal of Robotics Research*, 26(6), 2007.

[57] David G Lowe. Distinctive image features from Scale-Invariant keypoints. *Int. J. Comput. Vision*, 60:91110, November 2004.

[58] Feng Lu and E.E. Milios. Optimal global pose estimation for consistent sensor data registration. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 93 –100 vol.1, May 1995.

[59] F. Landis Markley. Multiplicative vs. additive filtering for spacecraft attitude determination. In *6th Cranfield Conference on Dynamics and Control of Systems and Structures in Space*, pages 467–474, 2004.

[60] Agostino Martinelli. Closed-form solution for attitude and speed determination by fusing monocular vision and inertial sensor measurements. In *IEEE International Conference on Robotics and Automation*, Shanghai, China, April 2011.

[61] Agostino Martinelli, Laurent Kneip, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence. In *IEEE International Conference on Robotics and Automation*, Shanghai, China, April 2011.

[62] Malika Meghjani and Gregory Dudek. Combining multi-robot exploration and rendezvous. *Computer and Robot Vision, Canadian Conference*, 0:80–85, 2011.

[63] F.M. Mirzaei and S.I. Roumeliotis. A kalman filter-based algorithm for IMU-Camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics*, 24(5):1143–1156, October 2008.

[64] L. Montesano, J. Minguez, and L. Montano. Probabilistic scan matching for motion estimation in unstructured environments. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3499 – 3504, August 2005.

[65] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3565–3572, Rome, Italy, April 2007.

[66] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3565–3572, Rome, Italy, April 2007.

[67] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[68] Richard A. Newcombe and Andrew J. Davison. Live dense reconstruction with a single moving camera. In *CVPR*, pages 1498–1505, 2010.

[69] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23:2006, 2006.

[70] E. B Olson. Real-time correlative scan matching. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4387 –4393, May 2009.

[71] B. W Parkinson and S. W Gilbert. NAVSTAR: global positioning system, ten years later. *Proceedings of the IEEE*, 71(10):1177 – 1186, October 1983.

[72] Nicolas Plamondon. *Modeling and Control of a Biomimetic Underwater Vehicle*. PhD thesis, Department of Mechanical Engineering, McGill University, Montreal, Canada, 2010.

[73] H. Rehbinder and B. K Ghosh. Multi-rate fusion of visual and inertial data. In *Multisensor Fusion and Integration for Intelligent Systems, 2001. MFI 2001. International Conference on*, pages 97 – 102, 2001.

[74] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, page 15081511, October 2005.

[75] Junaed Sattar, Gregory Dudek, Olivia Chiu, Ioannis Rekleitis, Alec Mills, Philippe Giguere, Nicolas Plamondon, Chris Prahacs, Yogesh Girdhar, Meyer Nahon, and John-Paul Lobos. Enabling autonomous capabilities in underwater robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3628–3634, Nice, France, September 2008.

[76] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. AIAA Education Series, Reston, VA, October 2003.

[77] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[78] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[79] M. D. Shuster. A survey of attitude representations. *The Journal of the Astronautical Sciences*, 41(4):439–517, 1993.

[80] M. D. Shuster. Constraint in attitude estimation part i: Constrained estimation. *The Journal of the Astronautical Sciences*, 51(1):51–74, 2003.

[81] M. D. Shuster. Constraint in attitude estimation part ii: Unconstrained estimation. *The Journal of the Astronautical Sciences*, 51(1):75–101, 2003.

[82] M.D. Shuster. The quaternion in kalman filtering. In *AAS/AIAA Astrodynamics Conference*, Victoria, BC, Canada, August 1993.

[83] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[84] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[85] Sajid Siddiqi, Gaurav S. Sukhatme, and Andrew Howard. Experiments in Monte-Carlo localization using WiFi signal strength. In *Proceedings of the International Conference on Advanced Robotics*, Coimbra, Portugal, July 2003.

[86] D. Simon and Tien Li Chia. Kalman filtering with state equality constraints. *Aerospace and Electronic Systems, IEEE Transactions on*, 38(1):128 –136, jan 2002.

[87] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Real-time monocular SLAM: why filter? In *ICRA*, pages 2657–2664, 2010.

[88] D. Strelow and S. Singh. Optimal motion estimation from visual and inertial measurements. In *Applications of Computer Vision, 2002. (WACV 2002). Proceedings. Sixth IEEE Workshop on*, pages 314 – 319, 2002.

[89] J. Stuelpnagel. On the parametrization of the three-dimensional rotation group. *SIAM review*, 6(4):422–430, 1964.

[90] J. Svab, T. Krajnik, J.Faigl, and L. Preucil. FPGA-based speeded up robust features. In *2009 IEEE International Conference on Technologies for Practical Robot Applications 2009*, November 2009.

[91] N. Trawny and S. I. Roumeliotis. Indirect kalman filter for 3D attitude estimation. Technical Report 2005-002, University of Minnesota, Dept. of Comp. Sci. and Eng.,, March 2005.

[92] N. Trawny and S. I. Roumeliotis. Indirect kalman filter for 3D attitude estimation. Technical Report 2005-002, University of Minnesota, Dept. of Comp. Sci. and Eng.,, March 2005.

[93] Bill Triggs, Philip Mclauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, page 298375. Springer Verlag, 2000.

[94] A. Vedaldi, Hailin Jin, P. Favaro, and S. Soatto. KALMANSAC: robust filtering by consensus. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 633 – 640 Vol. 1, October 2005.

[95] K. Vickery. Acoustic positioning systems. a practical overview of current systems. In *Autonomous Underwater Vehicles, 1998. AUV'98. Proceedings Of The 1998 Workshop on*, page 517, 1998.

[96] Eric A. Wan and Rudolph Van Der Merwe. The unscented kalman filter. In *Kalman Filtering and Neural Networks*, page 221280. Wiley, 2001.

[97] B. P. Williams, N. H. Hudson, B. E. Tweddle, R. Brockers, and L. Matthies. Feature and pose constrained visual aided inertial navigation for computationally constrained aerial vehicles. In *IEEE International Conference on Robotics and Automation*, Shanghai, China, April 2011.

[98] B. Woniak and J. Dera. *Light absorption in sea water*. Springer Verlag, 2007.

[99] Renato Zanetti, Manoranjan Majji, Robert H. Bishop, and Daniele Mortari. Norm-constrained kalman filtering. *Journal of Guidance Control and Dynamics*, 32:1458–1465, 2009.

APPENDIX A

A.1. Classical Rodriguez Parameters and Quaternions

$$crp2quat(s_1, s_2, s_3) = \frac{1}{\sqrt{1+\mathbf{s}\mathbf{s}^T}} \begin{bmatrix} s_1 & s_2 & s_3 & 1 \end{bmatrix} \tag{A.1}$$

$$quat2crp(q_1, q_2, q_3, q_4) = \frac{1}{q_4} \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \tag{A.2}$$

A.2. Modified Rodriguez Parameters and Quaternions

$$mrp2quat(m_1, m_2, m_3) = \frac{1}{1+\mathbf{m}\mathbf{m}^T} \begin{bmatrix} 2m_1 & 2m_2 & 2m_3 & 1-\mathbf{m}\mathbf{m}^T \end{bmatrix} \tag{A.3}$$

$$quat2mrp(q_1, q_2, q_3, q_4) = \frac{1}{1+q_4} \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \tag{A.4}$$

A.3. Derivative of the IMU state error

In the following proofs we are going to make frequent use of the approximation $\mathbf{R}(\tilde{\mathbf{q}}) \simeq \mathbf{I} - \lfloor \tilde{\boldsymbol{\theta}} \times \rfloor$ and $\lfloor a \times \rfloor^T = -\lfloor a \times \rfloor$

Lemma 1.

$$\overset{I}{\underset{G}{\dot{\tilde{\boldsymbol{\theta}}}}} \simeq -\lfloor \hat{\boldsymbol{\omega}} \times \rfloor \tilde{\boldsymbol{\theta}} - \tilde{\boldsymbol{b}}_g - \boldsymbol{n}_g \tag{A.5}$$

Proof. Can be found in [**91**]. □

LEMMA 2.

$$ {}^{G}\dot{\tilde{\boldsymbol{v}}}_I \simeq -\boldsymbol{R}^T({}^{I}_{G}\hat{\boldsymbol{q}})(\lfloor\hat{\boldsymbol{\alpha}}\times\rfloor\tilde{\boldsymbol{\theta}} + \tilde{\boldsymbol{b}}_a + \boldsymbol{n}_a) \tag{A.6} $$

PROOF.

$$
\begin{aligned}
{}^{G}\dot{\tilde{\mathbf{v}}} &= {}^{G}\dot{\mathbf{v}} - {}^{G}\dot{\hat{\mathbf{v}}} \\
&= R^T({}^{I}_{G}\mathbf{q})\boldsymbol{\alpha}_m - R^T({}^{I}_{G}\mathbf{q})\mathbf{b}_a - R^T({}^{I}_{G}\mathbf{q})\mathbf{n}_a - R^T({}^{I}_{G}\hat{\mathbf{q}})\hat{\boldsymbol{\alpha}} \\
&= R^T({}^{I}_{G}\hat{\mathbf{q}})R^T({}^{I}_{G}\tilde{\mathbf{q}})\boldsymbol{\alpha}_m - R^T({}^{I}_{G}\hat{\mathbf{q}})R^T({}^{I}_{G}\tilde{\mathbf{q}})\mathbf{b}_a - R^T({}^{I}_{G}\hat{\mathbf{q}})R^T({}^{I}_{G}\tilde{\mathbf{q}})\mathbf{n}_a - R^T({}^{I}_{G}\hat{\mathbf{q}})\hat{\boldsymbol{\alpha}} \\
&= R^T({}^{I}_{G}\hat{\mathbf{q}})(R^T({}^{I}_{G}\tilde{\mathbf{q}}) - \mathbf{I})\hat{\boldsymbol{\alpha}} - R^T({}^{I}_{G}\hat{\mathbf{q}})R^T({}^{I}_{G}\tilde{\mathbf{q}})\tilde{\mathbf{b}}_a - R^T({}^{I}_{G}\hat{\mathbf{q}})R^T({}^{I}_{G}\tilde{\mathbf{q}})\mathbf{n}_a \\
&\simeq R^T({}^{I}_{G}\hat{\mathbf{q}})\lfloor\tilde{\boldsymbol{\theta}}\times\rfloor\hat{\boldsymbol{\alpha}} - R^T({}^{I}_{G}\hat{\mathbf{q}})(\mathbf{I} + \lfloor\tilde{\boldsymbol{\theta}}\times\rfloor)\tilde{\mathbf{b}}_a - R^T({}^{I}_{G}\hat{\mathbf{q}})(\mathbf{I} + \lfloor\tilde{\boldsymbol{\theta}}\times\rfloor)\mathbf{n}_a \\
&\simeq -R^T({}^{I}_{G}\hat{\mathbf{q}})\lfloor\hat{\boldsymbol{\alpha}}\times\rfloor\tilde{\boldsymbol{\theta}} - R^T({}^{I}_{G}\hat{\mathbf{q}})\tilde{\mathbf{b}}_a - R^T({}^{I}_{G}\hat{\mathbf{q}})\mathbf{n}_a
\end{aligned}
$$

where we neglected the second-order error terms. □

A.4. Derivative of the continuous-time covariance matrix

$\dot{\tilde{\mathbf{X}}}_{IMU} = \mathbf{F}\tilde{\mathbf{X}}_{IMU} + \mathbf{G}\mathbf{n}_{IMU}$ is a stochastic differential equation. Assuming that $\tilde{\mathbf{X}}_{IMU}$ has zero mean

$$
\begin{aligned}
\hat{\mathbf{P}}(t+h) - \hat{\mathbf{P}}(t) &= E\left[\tilde{\mathbf{X}}(t+h)\tilde{\mathbf{X}}(t+h)^T\right] - E\left[\tilde{\mathbf{X}}(t)\tilde{\mathbf{X}}(t)^T\right] \\
&= E\Big[\{\mathbf{F}\tilde{\mathbf{X}}h + \mathbf{G}\Delta\mathbf{n}\}\{\mathbf{F}\tilde{\mathbf{X}}h + \mathbf{G}\Delta\mathbf{n}\}^T + \tilde{\mathbf{X}}\{\mathbf{F}\tilde{\mathbf{X}}h + \mathbf{G}\Delta\mathbf{n}\}^T + \\
&\quad \{\mathbf{F}\tilde{\mathbf{X}}h + \mathbf{G}\Delta\mathbf{n}\}\tilde{\mathbf{X}}^T\Big] \\
&= o(h) + \mathbf{G}E\left[\Delta\mathbf{n}\Delta\mathbf{n}^T\right]\mathbf{G}^T h + \mathbf{F}E\left[\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\right]h + E\left[\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\right]\mathbf{F}^T h
\end{aligned}
$$

There are a couple of points worth emphasizing here: first, we have assumed that $\tilde{\mathbf{X}}$ is independent of $\mathbf{n}$; second, notice this is a *stochastic* differential equation, so $\tilde{\mathbf{X}}(t+h) = \tilde{\mathbf{X}}(t) + (\mathbf{F}\tilde{\mathbf{X}}(t) + \mathbf{G}\mathbf{n})dt$ from ordinary calculus is not true. Instead, results from the Ito calculus, or calculus of stochastic variables, are used here. In

particular, $E\left[\Delta\mathbf{n}\Delta\mathbf{n}^T\right] = E\left[\mathbf{n}\mathbf{n}^T\right]h + o(h)$ is the one that enables this proof, and can be found in more detail in [**3**].

A.5. IMU-Camera covariance propagation

The solution of the stochastic differential equation $\dot{\tilde{\mathbf{X}}}_{IMU} = \mathbf{F}\tilde{\mathbf{X}}_{IMU} + \mathbf{G}\mathbf{n}_{IMU}$ in the interval $[t_k, t_{k+1}]$ is given by:

$$\begin{aligned}\tilde{\mathbf{X}}_{IMU}(t_{k+1}) &= \mathbf{\Phi}(t_{k+1};t_k)\tilde{\mathbf{X}}_{IMU}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{\Phi}(t_{k+1};s)\mathbf{G}\mathbf{n}(s)ds \\ \frac{d}{dt}\mathbf{\Phi}(t;t_k) &= \mathbf{F}\mathbf{\Phi}(t;t_k) \quad \text{with} \quad \mathbf{\Phi}(t_k;t_k) = \mathbf{I}\end{aligned}$$

Therefore

$$\begin{aligned}\hat{\mathbf{P}}_{IC_{k+1|k}} &= cov(\tilde{\mathbf{X}}_{IMU}(t_{k+1}), \tilde{\mathbf{X}}_{CAM}(t_k)) \\ &= \mathbf{\Phi}(t_{k+1};t_k)cov(\tilde{\mathbf{X}}_{IMU}(t_k), \tilde{\mathbf{X}}_{CAM}(t_k)) \\ &= \mathbf{\Phi}(t_{k+1};t_k)\hat{\mathbf{P}}_{IC_{k|k}} \\ &= \exp(\mathbf{F}(t_{k+1}-t_k))\hat{\mathbf{P}}_{IC_{k|k}}\end{aligned}$$

A.6. Covariance augmentation

We have

$$\begin{aligned}{}^{C_{N+1}}_{G}\mathbf{q} &= {}^{C}_{I}\mathbf{q} \otimes {}^{I}_{G}\mathbf{q} \\ {}^{C_{N+1}}_{G}\hat{\mathbf{q}} &= {}^{C}_{I}\mathbf{q} \otimes {}^{I}_{G}\hat{\mathbf{q}}\end{aligned}$$

So

$$\begin{aligned}{}^{C_{N+1}}_{G}\tilde{\mathbf{q}} &= {}^{C_{N+1}}_{G}\mathbf{q} \otimes {}^{C_{N+1}}_{G}\hat{\mathbf{q}}^{-1} \\ &= {}^{C}_{I}\mathbf{q} \otimes {}^{I}_{G}\mathbf{q} \otimes {}^{I}_{G}\hat{\mathbf{q}}^{-1} \otimes {}^{C}_{I}\mathbf{q}^{-1} \\ &= {}^{C}_{I}\mathbf{q} \otimes {}^{I}_{G}\tilde{\mathbf{q}} \otimes {}^{C}_{I}\mathbf{q}^{-1}\end{aligned}$$

Using the small-angle approximation for the error quaternions we have

$$\begin{bmatrix} \frac{1}{2}{}^{C_{N+1}}_{G}\tilde{\boldsymbol{\theta}} \\ 1 \end{bmatrix} \simeq {}^{C}_{I}\mathbf{q} \otimes \begin{bmatrix} \frac{1}{2}{}^{I}_{G}\tilde{\boldsymbol{\theta}} \\ 1 \end{bmatrix} \otimes {}^{C}_{I}\mathbf{q}^{-1}$$

And, via the algebraic properties of quaternion multiplication (see [**91**]), we get:

$$\begin{bmatrix} \frac{1}{2}{}^{C_{N+1}}_{G}\tilde{\boldsymbol{\theta}} \\ 1 \end{bmatrix} \simeq \begin{bmatrix} \mathbf{R}({}^{C}_{I}\mathbf{q}) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2}{}^{I}_{G}\tilde{\boldsymbol{\theta}} \\ 1 \end{bmatrix}$$

which implies that $cov({}^{C_{N+1}}_{G}\tilde{\boldsymbol{\theta}}) = \mathbf{R}({}^{C}_{I}\mathbf{q})cov({}^{I}_{G}\tilde{\boldsymbol{\theta}})\mathbf{R}^{T}({}^{C}_{I}\mathbf{q})$. For the error in the camera's position we have the following

$$\begin{aligned} {}^{G}\mathbf{p}_{C_{N+1}} &= {}^{G}\mathbf{p}_{I} + \mathbf{R}^{T}({}^{I}_{G}\mathbf{q}){}^{I}\mathbf{p}_{C} \\ {}^{G}\hat{\mathbf{p}}_{C_{N+1}} &= {}^{G}\hat{\mathbf{p}}_{I} + \mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}}){}^{I}\mathbf{p}_{C} \end{aligned}$$

so the errors will satisfy

$$\begin{aligned} {}^{G}\tilde{\mathbf{p}}_{C_{N+1}} &= {}^{G}\tilde{\mathbf{p}}_{I} + (\mathbf{R}^{T}({}^{I}_{G}\mathbf{q}) - \mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})){}^{I}\mathbf{p}_{C} \\ &= {}^{G}\tilde{\mathbf{p}}_{I} + (\mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})\mathbf{R}^{T}({}^{I}_{G}\tilde{\mathbf{q}}) - \mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})){}^{I}\mathbf{p}_{C} \\ &= {}^{G}\tilde{\mathbf{p}}_{I} + \mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})(\mathbf{R}^{T}({}^{I}_{G}\tilde{\mathbf{q}}) - \mathbf{I}){}^{I}\mathbf{p}_{C} \\ &\simeq {}^{G}\tilde{\mathbf{p}}_{I} + \mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})\lfloor {}^{I}_{G}\tilde{\boldsymbol{\theta}} \times \rfloor {}^{I}\mathbf{p}_{C} \\ &\simeq {}^{G}\tilde{\mathbf{p}}_{I} - \mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})\lfloor {}^{I}\mathbf{p}_{C} \times \rfloor {}^{I}_{G}\tilde{\boldsymbol{\theta}} \end{aligned}$$

so we have

$$\begin{aligned} cov({}^{G}\tilde{\mathbf{p}}_{C_{N+1}}) \simeq\ & cov({}^{G}\tilde{\mathbf{p}}_{I}) - \mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})\lfloor {}^{I}\mathbf{p}_{C} \times \rfloor cov({}^{I}_{G}\tilde{\boldsymbol{\theta}}, {}^{G}\tilde{\mathbf{p}}_{I}) \\ & -cov({}^{G}\tilde{\mathbf{p}}_{I}, {}^{I}_{G}\tilde{\boldsymbol{\theta}})(\mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})\lfloor {}^{I}\mathbf{p}_{C} \times \rfloor)^{T} + \\ & \mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})\lfloor {}^{I}\mathbf{p}_{C} \times \rfloor cov({}^{I}_{G}\tilde{\boldsymbol{\theta}})(\mathbf{R}^{T}({}^{I}_{G}\hat{\mathbf{q}})\lfloor {}^{I}\mathbf{p}_{C} \times \rfloor)^{T} \end{aligned}$$

A.7. Linearization of vision-based residual

$$
\begin{aligned}
\mathbf{H}_f^{C_i} &= \Big[\; \mathbf{0}_{2\times 15} \quad \mathbf{0}_{2\times 6} \quad \dots \quad \mathbf{0}_{2\times 6} \quad \underbrace{\mathbf{J}_f^{C_i}\lfloor {}^{C_i}\hat{\mathbf{p}}_f \times \rfloor \quad -\mathbf{J}_f^{C_i}\mathbf{R}({}^{C_i}_{G}\hat{\mathbf{q}})}_{\text{Jacobian wrt camera frame i}} \quad \mathbf{0}_{2\times 6} \quad \dots \;\Big]_{2\times(15+6N)} \\
\mathbf{U}_f^{C_i} &= \mathbf{J}_f^{C_i}\mathbf{R}({}^{C_i}_{G}\hat{\mathbf{q}}) \\
\mathbf{J}_f^{C_i} &= \nabla_{{}^{C_i}\hat{\mathbf{p}}_f}\mathbf{z}_f^{C_i} = \frac{1}{{}^{C_i}\hat{Z}_f}\begin{bmatrix} 1 & 0 & -\frac{{}^{C_i}\hat{X}_f}{{}^{C_i}\hat{Z}_f} \\ 0 & 1 & -\frac{{}^{C_i}\hat{Y}_f}{{}^{C_i}\hat{Z}_f} \end{bmatrix}
\end{aligned}
$$

TABLE A.1. Parameters used in the experimental section

Parameters	Definition	Value
σ_{im}	See Eq. (2.63)	0.003
σ_{depth}	See Eq. (2.73)	0.2 m
σ_{na}	See Eq. (2.38)	0.06 m/s^2
σ_{ng}	See Eq. (2.38)	0.009 rad/s
σ_{ba}	See Eq. (2.38)	0.0001 m/s^2
σ_{bg}	See Eq. (2.38)	0.0001 rad/s
Initial uncertainty for ${}^{I}_{G}\mathbf{q}$		2 degrees for pitch and roll 0 degrees for yaw
Initial uncertainty for $\mathbf{b}_g$		0.0001 rad/s
Initial uncertainty for ${}^{G}\mathbf{v}_I$		0.01 m/s
Initial uncertainty for $\mathbf{b}_a$		0.003 m/s^2
Initial uncertainty for ${}^{G}\mathbf{p}_I$		0 m
N_{max}	Max camera frames in the state	30
Feature detector		SURF
Feature descriptor		SURF
Feature matching		ANN, with distance ratio 0.5
Features per image		1100
Outlier detection		χ^2 criterion, 95%
Gravity magnitude		9.75 m/s^2

Document Log:

Manuscript Version 1.0

Typeset by $\mathcal{AMS}$-LaTeX — 30 November 2015

Florian Shkurti

McGill University, 3480 University St., Montréal (Québec) H3A 2A7, Canada,
Tel. : (514) 398-7071

E-mail address: florian@cim.mcgill.ca

Typeset by $\mathcal{AMS}$-LaTeX