EFFICIENT INERTIALLY AIDED VISUAL ODOMETRY TOWARDS MOBILE AUGMENTED REALITY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

YAĞIZ AKSOY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AUGUST 2013

Approval of the thesis:

EFFICIENT INERTIALLY AIDED VISUAL ODOMETRY TOWARDS MOBILE AUGMENTED REALITY

submitted by **YAĞIZ AKSOY** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. A. Aydın Alatan
Supervisor, **Electrical and Electronics Engineering Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering Dept., METU _____

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering Dept., METU _____

Assist. Prof. Dr. Umut Orguner
Electrical and Electronics Engineering Dept., METU _____

Assist. Prof. Dr. Sinan Kalkan
Computer Engineering Dept., METU _____

Dr. Zafer Arıcan
Argela _____

Date: **August 20, 2013**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: YAĞIZ AKSOY

Signature :

ABSTRACT

EFFICIENT INERTIALLY AIDED VISUAL ODOMETRY TOWARDS MOBILE AUGMENTED REALITY

Aksoy, Yağız

M.S., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. A. Aydın Alatan

August 2013, 83 pages

With the increase in the number and computational power of commercial mobile devices like smart phones and tablet computers, augmented reality applications are gaining more and more volume. In order to augment virtual objects effectively in real scenes, pose of the camera should be estimated with high precision and speed. Today, most of the mobile devices feature cameras and inertial measurement units which carry information on change in position and attitude of the camera. In this thesis, utilization of inertial sensors on mobile devices in aiding visual pose estimation is studied. Error characteristics of the inertial sensors on the utilized mobile device are analyzed. Gyroscope readings are utilized for aiding 2D feature tracking while accelerometer readings are used to help create a sparse 3D map of features later to be used for visual pose estimation. Metric velocity estimation is formulated using inertial readings and observations of a single 2D feature. Detailed formulations of uncertainties on all the estimated variables are provided. Finally, a novel, lightweight filter, which is capable of estimating the pose of the camera together with the metric scale, is proposed. The proposed filter runs without any heuristics needed for covariance propagation, which enables it to be used in different devices with different sensor characteristics without any modifications. Sensors with poor noise characteristics are successfully utilized to aid the visual pose estimation.

Keywords: Pose Estimation, Visual Odometry, Inertial Navigation, Probabilistic Robotics, Augmented Reality

ÖZ

GEZGİN ZENGİNLEŞTİRİLMİŞ GERÇEKLİĞE YÖNELİK EYLEMSİZLİK DESTEKLİ GÖRSEL ETKİN POZ TAKİBİ

Aksoy, Yağız

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. A. Aydın Alatan

Ağustos 2013 , 83 sayfa

Gündelik yaşamda kullanılan akıllı telefon, tablet bilgisayar gibi mobil cihazların sayısındaki ve işlemci gücündeki artışla birlikte, zenginleştirilmiş gerçeklik uygulamaları gittikçe daha çok hacim kazanmaktadır. Gerçek sahnelere etkili biçimde sanal nesne yerleştirilebilmesi için, kamera pozunun yüksek hassaslık ve hızla bulunması gerekir. Bu tezde eylemsizlik algılayıcılarının görsel poz kestirimine destek için kullanımı çalışılmıştır. ullanılan mobil cihaz üzerindeki eylemsizlik algılayıcılarının gürültü özellikleri analiz edilmiştir. Jiroskop ölçümleri 2B nokta takibine yardım etmek için kullanılırken, ivmeölçer ölçümleri, sonradan görsel poz kestiriminde kullanılmak üzere seyreltik 3B noktalardan oluşan haritanın yaratılmasına yardım için kullanılmaktadır. Metrik hız kestirimi eylemsizlik ölçümleri ve 2B bir noktanın gözlemleri kullanılarak formüle edilmiştir. Kestirilen değişkenler üzerindeki belirsizliklerin ayrıntılı formülleştirilmesi sunulmuştur. Son olarak, kamera pozunu metrik ölçekle birlikte kestirme yetisine sahip, özgün, hafif bir süzgeç önerilmektedir. Önerilen süzgeç, kovaryans türetmek için hiçbir buluşsal parametre gerektirmemektedir.

Anahtar Kelimeler: Poz Kestirimi, Görsel Poz Takibi, Eylemsizliksel Gezinim, Olasılıksal Robotik, Zenginleştirilmiş Gerçeklik

Hold on, hold on!

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor Prof. Dr. A. Aydin Alatan for his support and guidance throughout the first three years of my research life. His friendly and at the same time professional way of supervision boosted the satisfaction and joy I get from academia. I am thankful to him also for introducing me to the field of computer vision.

Being a part of Multimedia Research Group has been a delight. I want to thank Emin Zerman for his closest friendship and all the help he provided during the preparation of this thesis. The lab would not be as a joyful environment as it is if it weren't for Emrecan Bati and Beril Besbinar and their sincere as well as *divergent* conversations. I am also grateful to *the-nicest-person-on-the-Earth* Akin Caliskan, the juggler of the lab Ozan Sener and Yeti Ziya Gurbuz for their friendship. I am thankful to Salim Sirtkaya for the valuable discussions about inertial navigation. I also acknowledge Mehmet Mutlu, Ahmet Saracoglu and O. Serdar Gedik for their friendship and support.

I would like to express my gratitude for Dr. Engin Tola, Dr. Emrah Tasli, Dr. Cevahir Cigla and Dr. Yoldas Ataseven for their kind friendship as well as their help and guidance about my studies and my future plans.

It has been a pleasure to work with Dr. Zafer Arican during the construction of this thesis. I would also like to thank Argela and Scientific and Technological Research Council of Turkey (TUBITAK) for their financial support during my Master's studies.

The biggest thanks go to my family. I can not thank enough for the opportunities and support my father Nadir Aksoy and my mother Vicdan Aksoy provided me. I also thank my brother Yalin Aksoy for all the moments we have been sharing since the day he was born. This thesis is dedicated to them for their unconditional love.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vi

ACKNOWLEDGEMENTS viii

TABLE OF CONTENTS ix

LIST OF TABLES xiii

LIST OF FIGURES xiv

CHAPTERS

1 INTRODUCTION 1

1.1 Problem Statement 1

1.2 Augmented Reality 1

1.3 Visual Odometry 2

1.4 Combination of Inertial and Visual Information 3

1.4.1 Augmented Reality 3

1.4.2 Robot Navigation 4

1.5 Scope and Organization of the Thesis 5

2 VISUAL POSE ESTIMATION 7

2.1 Feature Tracking 7

2.1.1 Keypoint Detection 7
2.1.2 2D Feature Tracking 8
2.1.3 Gyroscope Aided Feature Tracking 9
2.1.4 Uncertainty on Feature Position 11
2.2 Map Generation and Map Handling 11
2.2.1 Map Generation 11
2.2.1.1 Computation of translation during map initialization 12
2.2.1.2 Computation of scaling between estimated visual pose and metric scale 15
2.2.2 Map Handling 16
2.3 Pose Estimation from 2D-3D Point Correspondences 16
2.3.1 Uncertainty on Estimated Pose 17
2.4 Experimental Results 18
2.4.1 Gyroscope-aided 2D Tracking 18
2.4.2 Visual Pose Estimation 20
2.4.3 Map Handling 21
3 INERTIAL POSE ESTIMATION 25
3.1 Inertial Sensors on Mobile Devices 25
3.1.1 Noise Characteristics 26
3.1.1.1 Bias Analysis 26
3.1.1.2 Distribution of Noise 27
3.1.1.3 Power Spectral Density Estimation 30

3.2 External Calibration of the IMU and the Camera 30
3.3 Metric Translational Velocity Computation 32
3.3.1 Uncertainty on Estimated Velocity 37
3.4 Computation of Change in Attitude 38
3.4.1 Uncertainty on Change in Attitude 38
3.5 Inertial Measurement Vector 39
3.6 Experimental Results 39
4 FUSING INERTIAL AND VISUAL INFORMATION FOR ODOMETRY . . 43
4.1 Proposed Kalman Filter 43
4.1.1 The Prediction Stage 44
4.1.2 The Correction Stage 45
4.1.3 System Initialization 45
4.2 Experimental Results 45
4.2.1 Metric Scale Estimation 45
4.2.2 Pose Estimation 46
5 CONCLUSIONS AND FUTURE WORK 53
REFERENCES 55
APPENDICES
A KALMAN FILTER 63
A.1 Kalman Filter 63
A.2 Extended Kalman Filter 64

B QUATERNIONS FOR REPRESENTING ROTATIONS 67

B.1 Transformation between Rotation Quaternions and Rotation Matrices 68

B.2 Angular Displacement Represented in Quaternions 69

B.3 Related Jacobians 70

B.4 Advantages of Using Quaternions for Representing Rotations 71

C MULTIVIEW GEOMETRY BASICS 73

C.1 Pinhole Camera Model 73

C.2 Triangulation 74

D COMPUTATION OF JACOBIAN OF 3D-TO-2D PROJECTION FUNCTION 77

E COMPUTATION OF JACOBIAN OF MATRICES USED IN METRIC VELOCITY ESTIMATION 79

LIST OF TABLES

TABLES

Table 2.1 Average time required for visual pose and covariance computation for different map sizes 20

Table 3.1 Standard deviance of noise on the accelerometer and the gyroscope 30

Table 4.1 Average time required for system propagation for different map sizes 48

LIST OF FIGURES

FIGURES

Figure 2.1 FAST keypoint detection (image taken from [72]). FAST algorithm compares intensity values of the pixels numbered from 1 to 16 with intensity of the center pixel. If there are long series of points in the circle that have higher/lower intensities than the center pixel, the center pixel is marked as a keypoint. 8

Figure 2.2 Amount of 2D displacements resulting from rotation of the camera (image taken from [25]) 10

Figure 2.3 Uncertainty on 3D point positions after triangulation (image taken from [22]). Note that higher the angle between two rays from the camera center, lower the uncertainty on the 3D feature position. 13

Figure 2.4 A typical example of the prediction of the next positions of the keypoints using gyroscope readings (Red lines are drawn between the initial position of a point and the predicted position while green lines are between the predicted position and the final position) 19

Figure 2.5 An ideal example of the prediction of the next positions of the keypoints using gyroscope readings (Red lines are drawn between the initial position of a point and the predicted position while green lines are between the predicted position and the final position) 19

Figure 2.6 Worst-case scenario for the prediction of the next positions of the keypoints using gyroscope readings (Red lines are drawn between the initial position of a point and the predicted position while green lines are between the predicted position and the final position) 20

Figure 2.7 Histograms of the distances between the previous or the predicted positions and the final positions of tracked keypoints 21

Figure 2.8 Position estimation performances with various map sizes 22

Figure 2.9 Attitude estimation performances with various map sizes 22

Figure 2.10 Position estimation performances with various map sizes 23

Figure 2.11 Attitude estimation performances with various map sizes 23

Figure 2.12 Different steps of map handling 24

Figure 3.1 Bias terms on linear accelerometer readings 26

Figure 3.2 Bias terms on gyroscope readings 27

Figure 3.3 Distribution of linear accelerometer readings 28

Figure 3.4 Distribution of gyroscope readings 29

Figure 3.5 Power spectral density of linear accelerometer readings 31

Figure 3.6 Power spectral density of gyroscope readings 31

Figure 3.7 Visualization of asynchronous operation 32

Figure 3.8 Definition of three poses and relative transformations 33

Figure 3.9 Distribution of the error on estimated translation when attitude change is computed from the visual measurements or from the gyroscope 40

Figure 3.10 Distribution of the error on estimated translation on three axes 41

Figure 3.11 Distribution of the error on estimated rotation 41

Figure 4.1 Estimated scale in a video with complex motion 46

Figure 4.2 Estimated scale in a video with mostly translational motion 47

Figure 4.3 Variance of the estimated scale in a video with mostly translational motion 48

Figure 4.4 Position estimation performance of the Kalman filter with regular velocity estimation and velocity estimation with discarded $z-axis$ 49

Figure 4.5 Position estimation performance of the Kalman filter with a map size of 16, compared to the corresponding visual pose estimation 49

Figure 4.6 Attitude estimation performance of the Kalman filter with a map size of 16, compared to the corresponding visual pose estimation 50

Figure 4.7 Position estimation performance of the Kalman filter with a map size of 8, compared to the corresponding visual pose estimation 50

Figure 4.8 Attitude estimation performance of the Kalman filter with a map size of 8, compared to the corresponding visual pose estimation 51

Figure 4.9 Position estimation performance of the Kalman filter with a map size of 20, compared to the corresponding visual pose estimation 51

Figure 4.10 Attitude estimation performance of the Kalman filter with a map size of 20, compared to the corresponding visual pose estimation 52

Figure C.1 Pinhole camera model. Here, U is the point in 3D, u is the projection of the 3D point on the image plane, f is the focal distance, C is the camera center and c is the principal point. 73

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Enhancing a real environment by rendering information, artificial objects or similar content for entertainment or productivity is called Augmented Reality (AR). AR has an evergrowing volume in today's mobile world. One of the most important requirements of AR is finding the pose of the camera with respect to the world in order to augment the objects in the right place. As a result, creating a robust and lightweight pose estimation system is a big challenge in the academic community.

Pose estimation is generally conducted using the visual cues in AR applications. When the environment is controlled, *i.e.* a known target is placed in the environment and the augmentation is done on that object, pose estimation is no harder than detecting a known target in the image stream. On the other hand, if the challenge of pose estimation in an unknown environment is taken, algorithms that are able to understand the structure of the environment and then compute the camera pose are needed. This requirement results in computationally heavy algorithms when high accuracy is needed. Since today's AR applications are mainly directed to be run on mobile devices, achieving real-time operation is a great challenge.

Most commercial off-the-shelf (COTS) mobile devices are equipped with cameras as well as additional sensors such as gyroscopes, accelerometers or compasses. These sensors carry important information on the pose or pose change of the devices. Since the errors on information coming from these sensors are independent of the visual information, utilization of the two mediums together is expected to give a more robust pose estimation.

In this thesis, efficiency being the main focus, inertially aided pose computation using a monocular image stream is studied. Ways to utilize inertial information to enable a computationally lighter pose estimation are analyzed.

1.2 Augmented Reality

In parallel with the increasing number and computational power of commercial mobile devices, AR applications get widespread everyday. There are many AR applications designed to be used on tablets or mobile phones. For instance, Nokia announced an AR application called *City Lens*, that shows information about the place the camera is pointed at [91]. IKEA uses AR to show the items in the shopping catalogue in 3D and give the opportunity to furnish

a room using virtual furniture [71]. AR can also be used in increasing productivity at work. One of the first real-life applications of AR is enhanced vision for aircraft pilots during flight [16]. In [97], authors propose an AR system to guide teachers during lectures. Poelman *et al.* [69] describes an application to guide crime scene investigations.

Today, in addition to tablets and mobile phones, a more recent technology, smart glasses technology create a new volume for AR applications. For instance, recently-announced *Google Glass* promises many AR applications such as AR assisted navigation [64]. The patent applications, such as [40] by Microsoft, show the commercial interest towards the future of AR.

Increasing applicability results in development of hardware that is specifically designed for AR. Metaio and ST Ericsson recently announced a collaboration to develop *AREngine*, a chipset aimed at providing real-time AR experience on mobile devices [58].

The increase in the commercial volume of AR is supported by active AR research. The active research topics in augmented reality, as listed in [99], are:

- Graphics rendering
- Pose tracking
- Tracker calibration and registration
- Display hardware
- Computer processing hardware
- Human interaction techniques

International Symposium on Mixed and Augmented Reality (ISMAR) can be said to be the most widely known international academic event that focuses on AR research. According to the survey conducted in [99], the most active topic in AR research appears to be pose tracking in 10 years of ISMAR, between 1998 - 2008.

1.3 Visual Odometry

Camera pose extraction is an active topic in several different research areas, mainly in computer vision and mobile robotics. In this section, a brief review of literature is presented. For a more thorough literature survey, readers are referred to two tutorials published by Scaramuzza and Fraundorfer [77, 18]. Vision research that is aimed at augmented reality has been producing methods for pose estimation in two main categories: filter-based and bundle adjustment (*i.e.* optimisation) -based.

For filter-based approaches, Davison's work [9], followed by the MonoSLAM algorithm by Davison *et al.* [10] can be considered as a typical example. In MonoSLAM, authors construct a probabilistic map of *high quality* sparse features [10]. They report achieving real-time performance on a PC while also maintaining a high accuracy in the 3D feature positions and the estimated camera pose.

One of the leading techniques in optimization-based category is Parallel Tracking and Mapping (PTAM) proposed by Klein and Murray [33]. They separate the tracking (odometry) and mapping processes and run them at different speeds at different threads. The mapping process is run at a much smaller speed than the tracking. This allows them to map many sparse features, resulting in an almost dense map. They define *keyframes*, selected such that they are far from each other and has high tracking quality. Keyframes are then used in the mapping process. The authors develop a mobile version of PTAM [34], which is computationally more efficient while less accurate. They report running it real-time on a mobile phone.

Strasdat *et al.* [81] compare the advantages of filter- and optimization-based approaches in terms of accuracy and computational cost. They conclude that although optimization-based methods are shown to be more accurate, for a resource constrained system, filter-based systems should be more advantageous [81].

Fua and Lepetit [19] create a system to track a visual marker in order to augment objects on it. In their paper, they describe planar object tracking as well as deformable object tracking. They are able to estimate how a planar target on a napkin or a t-shirt is deformed and they augment the new object according to the estimated deformation.

In order to simplify the problem and achieve real-time operation on a mobile phone, Pirchheim and Reitmayr [68] assume a planar target and track the estimated homography of the target rather than computing the 6 degree-of-freedom pose.

1.4 Combination of Inertial and Visual Information

Fusing inertial and visual information for tracking or pose estimation purposes gained attention in two main academic communities: computer vision and mobile robotics. We will briefly present the computer vision literature focusing on the AR and mobile robotics literature focusing mainly on robot navigation.

1.4.1 Augmented Reality

There are several ways to incorporate inertial measurements to augmented reality systems proposed in the literature. We can roughly divide them into two categories: tightly coupled and loosely coupled. In tightly coupled systems, inertial measurements are used to aid generation of visual measurements at a low level. Loosely coupled systems, however, treat two mediums independently and then fuse two information sources to generate the final estimation.

Accelerometers can be used to estimate the direction of gravity. This is used to create *gravity-aware* systems. Kurz and Benhimane [37] propose a gravity-aware interest point description scheme. By describing interest points that are on a vertical plane by taking the gravity vector as reference, they improve the descriptiveness of the features and diminish the need for rotation-invariant interest point description. They then use the gravity-aware keypoints in an augmented reality system [38] for vertical planes, such as building façades. Gravity is also used for effective artificial object rendering on horizontal surfaces in their system. Gravity vector is utilized also by Arth *et al.* [2] for a fast and accurate localization from images of a city.

Gyroscopes are shown to be useful to aid tracking applications. Kaheel *et al.* [27] compensate for the rotation of the camera between two frames using gyroscope measurements so that no rotational invariance is required during interest point description and matching. Klein and Drummond [31, 32] use gyroscope measurements to estimate the direction of motion blur. In [96, 95], You *et al.* use a gyroscope and a compass to estimate relative change in orientation between two frames. The estimated change is then used to predict new locations of the tracked 2D points, making the tracking robust against fast motions and motion blur [96, 95]. Inertial measurements are supported by vision to prevent them from diverging. In [26], they combine gyroscope measurements with a line-based tracking system and achieve real time operation. Hwangbo *et al.* [25] also use gyroscope measurements to predict locations of the tracked points and propose an enhanced Kanade-Lukas-Tomasi tracking algorithm.

Inertially aided visual odometry is studied briefly in the augmented reality literature. In 1998, Azuma *et al.* [3] noted that hybrid tracking is indeed necessary especially for outdoors augmented reality. You and Neumann [94] utilizes an extended Kalman filter in which the state is iterated with each gyroscope or vision measurement separately. The authors note an increase in the accuracy of the combined algorithm when compared to two information sources alone. Lang *et al.* [39] iterate inertial measurements at a high frequency and correct the drift in position at a lower frequency using visual measurements. Foxlin *et al.* [16] uses two inertial sensors differentially in order to find relative motion of the pilot's head from the cockpit in an augmented reality application for enhanced vision during flight. In [7], effect of use of inertial measurements while tracking CAD models is analyzed. Recently, Oskiper *et al.* [65, 66] proposed an error state Kalman filter where all tracked points are considered as individual visual measurements. Together with inertial measurements, their system use also GPS measurements for wide area localization and odometry.

1.4.2 Robot Navigation

In the robot navigation literature, generally, a visually-aided inertial navigation is preferred rather than inertially-aided visual odometry. Visual information is generally used to prevent inertial navigation from diverging due to the bias. The most commonly used technique for visually aided inertial odometry is indirect Kalman filter. Indirect Kalman filter is an extended Kalman filter where the state consists of errors on navigation input together with optional inertial sensor biases. For example, Roumeliotis *et al.* [75] and Trawny *et al.* [87] utilize error-state Kalman filters for spacecraft navigation. In [11], authors propose an indirect Kalman filter formulation that is enhanced with epipolar geometry constraints. In [67], the error state is defined with errors on both inertial and visual sensors.

Tardif *et al.* [84] use a stereo camera setup aided by inertial measurements. They use a *delayed-state* Kalman filter, where the prediction step of the Kalman filter is repeated with each inertial measurement and the state is updated once new visual measurement comes. They combine the filtering approach with optimization algorithms that are run in small time windows to increase accuracy.

Strelow and Singh [82] use an *iterated* Kalman filter, where, similar to [94], filter is updated with either of inertial or visual measurements as soon as they arrive. [82] also proposes a batch optimization method and compares the results with the online, filtering based approach.

Lobo and Dias [49] determine the horizon line using the vertical reference from inertial mea-

surements. By using only one vanishing point, together with the inertial measurements, they show that it is possible to estimate the focal distance of the camera. They also utilize inertial measurements with visual information for reconstruction of the scene.

Troiani and Martinelli [88] and Martinelli [55] analyze which modes are observable when visual observations are combined with inertial readings. With the metric scale being observable, Nutzi *et al.* [63], Kneip *et al.* [36] and Lupton and Sukkarieh [54] develop algorithms to estimate the metric scale of the visual pose measurements.

Mourikis and Roumeliotis [61] propose the so-called multi-state constraint Kalman filter (MSCKF). In this paper, authors propose a motion model that is constrained by each tracked point observation separately. They achieve this without augmenting each observed point in the Kalman filter state, making the filter more efficient. Instead, each measured pose is augmented to the state while the state is enlarged up to a maximum number of previous poses. The processing is done off-line using SIFT [51], which is a very robust but computationally heavy keypoint detection and description algorithm. In [44], Li and Mourikis propose modifications on computation of covariance matrices in order to improve the performance of MSCKF. The authors propose a computationally lighter version of MSCKF in [46], allowing a smaller number of poses to be augmented to the state with several other performance improvements. They show that operation at 10 Hz is possible on a mobile phone with the proposed modifications. They report optimizing the estimators in their system in [45]. Li and Mourikis then propose MCKFC 2.0 [47], which is reported to have better performance than conventional EKF-SLAM systems without explicitly storing the mapped points. One should note that the mobile applications presented in [46] and [45] does not implicitly report using the inertial sensors on COTS mobile devices and the reported sampling rates of the inertial sensors are much higher than the one utilized in this thesis. In [43], authors use MSCKF 2.0 for long distance odometry, successfully utilizing inertial sensors on a COTS mobile device with visual information at 5 frames per second processing speed.

Other than the commonly-used extended Kalman filter, there are several other filters proposed to be used in inertial - visual sensor cooperation in robot navigation. A similar but more computationally complex Kalman filter type, called Unscented Kalman filter, is utilized in [6]. Two examples that utilize particle filter are [93] and [14].

1.5 Scope and Organization of the Thesis

In this thesis, the task of utilizing accelerometer and gyroscope measurements to aid the visual pose estimation is undertaken.

In Chapter 2, visual pose estimation is described. This chapter starts with 2D feature tracking. A gyroscope-aided 2D feature tracking algorithm is also described. 2D feature tracking is followed by 3D sparse map generation and handling using the tracked features. The initial 3D positions of the mapped points are determined with help from accelerometer readings. The theoretical formulation in this chapter ends with the pose estimation from the tracked points and generated map.

Chapter 3 is on computation of relative pose using the inertial measurements. The chapter starts with a detailed analysis of the error characteristics of the inertial sensors utilized. Then, formulation of metric velocity estimation using inertial measurements and one tracked feature

is presented. The velocity estimation is followed by relative attitude change computation.

The Extended Kalman Filter that is used to fuse two information mediums for pose estimation is presented in Chapter 4. The state vector of the proposed filter is kept as small as possible for computational efficiency. The state also includes the scaling between visual measurements and metric scale. In the filter, inertial measurements are used for the prediction stage while visual measurements are used in the correction stage. Using measurements in both stages of the filter results in covariance propagation without any assumptions or heuristics needed, separating our method from competing algorithms such as [46]. This makes our system theoretically complete while also easy to utilize in other devices with different inertial sensor characteristics out-of-the-box.

Chapters 2, 3 and 4 feature detailed formulation of uncertainties on the estimated variables and experimental results.

The thesis is concluded in Chapter 5.

CHAPTER 2

VISUAL POSE ESTIMATION

Computing the pose of the camera visually requires some features, such as surfaces, lines or points, be present in the image while the locations and/or orientations of these landmarks are known in 3D world coordinates. In this thesis, point features are utilized. Detection and tracking of 2D features on the image plane are described in Section 2.1. Section 2.1 also includes a gyroscope-aided tracking algorithm. Determining the 3D positions of the tracked keypoints are then described in Section 2.2. Computation of the visual pose measurement using the tracked points and their corresponding 3D positions are described in Section 2.3. Each section features an analysis of uncertainty on the estimated parameters.

2.1 Feature Tracking

2.1.1 Keypoint Detection

A *local feature* is a distinctive point or region in its immediate neighborhood. These features are utilized for three main tasks: semantic interpretation, identifying localized anchor points and image representation [89]. In this thesis, we focus on tracking anchor points - or *keypoints* - in image streams in order to locate them in 3D and eventually compute camera pose from 2D-3D point correspondences.

Literature for keypoint detection is very rich. There are two main categories: blob detection and corner detection. The blob detectors, such as SIFT [51] or SURF [5], are generally more robust compared to corner detectors. Corner detectors like Harris [21], Good Features to Track [78], BRISK [42] or FAST [72], on the other hand, stand out by being more efficient.

In our system, ORB [76] keypoints are utilized. ORB keypoints are consistent FAST keypoints [72, 73, 74] that occurs simultaneously in several scales at the same point in the Gaussian scale space. As seen in Figure 2.1, FAST locates keypoints by comparing intensities of pixels on a circle around the center point (see Figure 2.1). If there is a large serie of low-intensity or high-intensity points, the center is marked as a FAST corner.

We detect the keypoints using FAST and select the ones with highest Harris scores [21]. Harris score is a cornerness measurement representing how sharp the detected corner is. It is based on second order moments in the neighborhood of the detected corner. Points with high Harris scores are also favorable by the utilized Kanade-Lucas-Tomasi tracking algorithm [53, 78] as the Harris cornerness measure is based on second order moments.

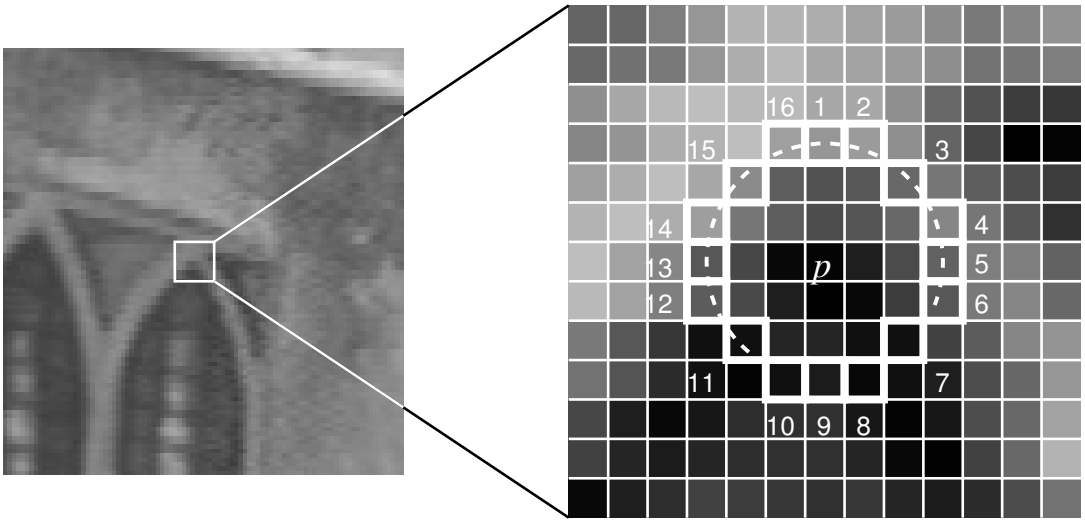


Figure 2.1: FAST keypoint detection (image taken from [72]). FAST algorithm compares intensity values of the pixels numbered from 1 to 16 with intensity of the center pixel. If there are long series of points in the circle that have higher/lower intensities than the center pixel, the center pixel is marked as a keypoint.

2.1.2 2D Feature Tracking

The detected keypoints are tracked in consecutive frames for 3D map generation and pose computation. For the tracking, we use a robust version of the iterative Kanade-Lucas-Tomasi (KLT) feature tracker [4, 53, 78].

Let us denote two consecutive grayscale frames with I_0 and I_1 and a point $\vec{u} = \begin{bmatrix} u_x & u_y \end{bmatrix}^T$ in I_0 with corresponding location $\vec{v} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ in I_1. The KLT algorithm minimizes the following cost function [8]:

$$\xi(\vec{d}) = \xi(\begin{bmatrix} d_x \\ d_y \end{bmatrix}) = \sum_W (I_0(x,y) - I_1(x+d_x, y+d_y))^2 \tag{2.1}$$

where W is the search window and $\vec{d}$. To find $\vec{d}$ that minimizes the cost function above, a method based on image derivatives is proposed [4, 78]. Firstly, let us define the image derivatives:

$$\frac{\partial I_0(x,y)}{\partial x} = I_{0,x} = \frac{I_0(x+1,y) - I_0(x-1,y))}{2} \tag{2.2}$$

$$\frac{\partial I_0(x,y)}{\partial y} = I_{0,y} = \frac{I_0(x,y+1) - I_0(x,y-1))}{2} \tag{2.3}$$

The derivatives are used to construct the *spatial gradient matrix* [8]:

$$G = \sum_{W} \begin{bmatrix} I_{0,x}^2(x,y) & I_{0,x}(x,y)I_{0,y}(x,y) \\ I_{0,x}(x,y)I_{0,y}(x,y) & I_{0,y}^2(x,y) \end{bmatrix} \tag{2.4}$$

After the construction of G, an iterative process starts with an initial guess for $\vec{v}$. The initial guess can be selected as the previous location of the point, $\vec{u}$. The new location can also set to a predicted location using a motion model or additional measurements such as gyroscopes (see Section 2.1.3). Let us denote the translation between the original location of the point and the estimated current location by $\hat{\vec{v}} - \vec{u} = \hat{\vec{d}}$. Difference between two image windows during the k^{th} iteration is:

$$\delta I_k(x,y) = I_0(x,y) - I_1(x + \hat{d}_{x,k}, y + \hat{d}_{y,k}) \tag{2.5}$$

Then, *image mismatch vector* is constructed as:

$$\vec{m}_k = \sum_{W} \begin{bmatrix} \delta I_k(x,y) I_{0,x}(x,y) \\ \delta I_k(x,y) I_{0,y}(x,y) \end{bmatrix} \tag{2.6}$$

As proposed by Lucas and Kanade [53], using G and $\vec{m}_k$, the flow of the point can be computed by:

$$\vec{f}_k = G^{-1}\vec{m}_k \tag{2.7}$$

Then the new estimation for the point location becomes:

$$\hat{\vec{v}}_k = \hat{\vec{v}}_{k-1} + \vec{f}_k \tag{2.8}$$

The equations (2.5) - (2.8) are iterated until the flow $\vec{f}_k$ becomes too small or maximum number of iterations is reached.

The point is declared lost if the cost function is higher than a threshold after an affine transformation is fit between two image patches, and those patches are compared [78].

In KLT, since the tracking is performed between consecutive frames only, the tracked points can drift off their original location. To minimize the effect of this shortcoming, lost points should be detected with high precision. There are additional methods to detect the lost points that are erroneously marked as not lost by KLT tracker, such as *Template Inverse Matching* (TIM) [48]. In TIM, after new locations of the tracked points are found by KLT, they are tracked backwards to the previous frame. If the original and the re-tracked positions of a point in the previous frame are far from each other, the point is marked as lost.

2.1.3 Gyroscope Aided Feature Tracking

Under large changes in the attitude of a camera, displacement of 2D points on the image plane can be very large [25]. As it can be seen in Figure 2.2, if the camera rotates rapidly, a larger

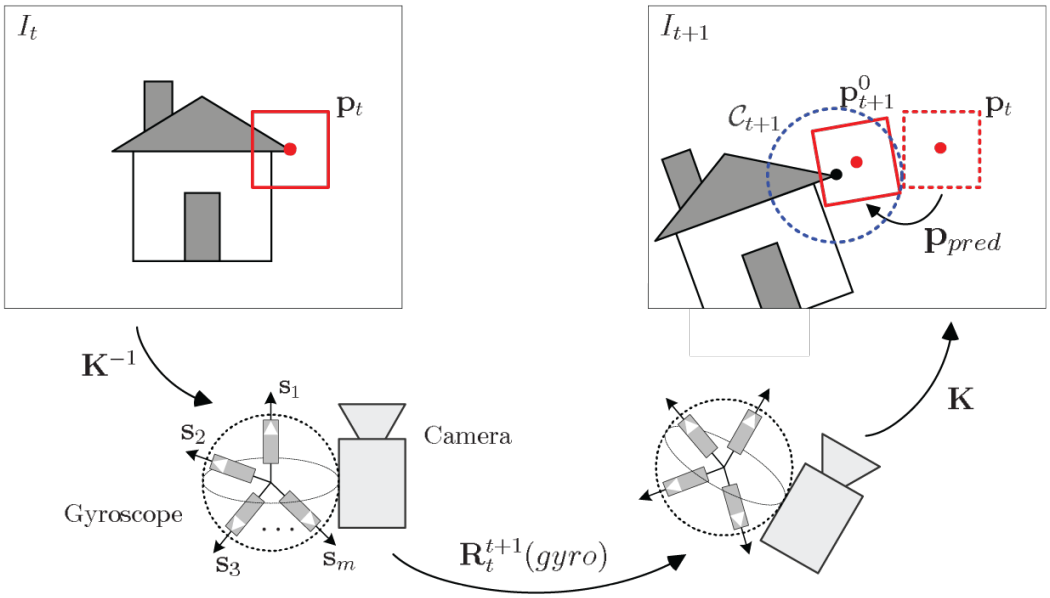


Figure 2.2: Amount of 2D displacements resulting from rotation of the camera (image taken from [25])

search window is needed not to lose the tracked point, which means a longer computation time.

Now, consider two consecutive frames in an image stream. Let the projection matrices corresponding to two poses of the camera be $P_1 = K[I_{3\times3}\|\vec{0}_{3\times1}]$ and $P_2 = K[R\|\vec{t}]$. Also consider a plane in 3-dimensional space represented with a normal vector $\vec{n}$ and a scalar d such that $\vec{n}^T X + d = 0$ for points X on the defined plane. Then, projections of the points on the plane in two images can be related by a homography matrix H [22].

$$H = K(R - \frac{\vec{t}\vec{n}^T}{d})K^{-1} \tag{2.9}$$

This homography matrix relates the points in two images as in (2.10).

$$z_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{2.10}$$

where z_2 is a normalization parameter converting homogeneous coordinates to actual point coordinates. If the translation between two camera poses is zero, (2.9) becomes:

$$H = KRK^{-1} \tag{2.11}$$

Notice that (2.11) is independent of the plane. This means that if the translation between two poses is close to zero, every 2D point in the first frame can be approximately mapped to the second frame by a single homography without a planarity assumption.

When gyroscope readings are available, we readily have a measurement of interframe rotation. Then, using (2.11), one can generate predictions for new point locations in the new frame [25]. By placing the search window around the predicted points, fine locations of the tracked points can be estimated using the utilized tracking algorithm. This way, under large camera rotations, tracking can be achieved with a smaller search window size, as seen in Figure 2.2.

2.1.4 Uncertainty on Feature Position

Computation of uncertainty on locations for the tracked points is based on the work of Nickels and Hutchinson [62]. We compute the sum of squared distances (SSD) (2.12) in a small window around the tracked point (W) (we used a window of 5×5) in the neighborhood of the point (SW). The neighborhood is selected with the same size as the search window used in KLT tracker. The *response distribution* (RD) is obtained by applying an exponential function to SSD_i (2.13). A 2D Gaussian distribution is fit to RD_i in order to estimate the covariance P_i of the i^{th} tracked point (2.14).

$$SSD_i(u,v) = \sum_{m,n \in W} \big(W(m,n) - SW(u+m, v+n)\big)^2 \tag{2.12}$$

$$RD_i = e^{-k_i \times SSD_i};\ such\ that\ k_i = \arg\min_{k_i} |1 - \sum_{m,n \in SSD} e^{-k_i \times SSD}| \tag{2.13}$$

$$\Sigma_i = \begin{bmatrix} \frac{\sum_{u,v \in RD} RD_i(u,v)(u-u_i)^2}{\sum_{u,v \in RD} RD_i(u,v)} & \frac{\sum_{u,v \in RD} RD_i(u,v)(u-u_i)(v-v_i)}{\sum_{u,v \in RD} RD_i(u,v)} \\ \frac{\sum_{u,v \in RD} RD_i(u,v)(u-u_i)(v-v_i)}{\sum_{u,v \in RD} RD_i(u,v)} & \frac{\sum_{u,v \in RD} RD_i(u,v)(v-v_i)^2}{\sum_{u,v \in RD} RD_i(u,v)} \end{bmatrix} \tag{2.14}$$

2.2 Map Generation and Map Handling

Computation of visual pose measurement requires features on the image plane together with their corresponding positions in 3D with respect to a reference coordinate frame. The set of points with known (or as described in this section, estimated) 3D positions is called the *map*. As the tracked 2D points are sparse, our map consists of sparse points in 3D. For a more effective visual pose estimation, 2D points are selected such that they are far from each other on the image plane.

2.2.1 Map Generation

In order to find three-dimensional locations of the 2D points detected in a monocular system, one should observe the points in at least two different poses. It is not straightforward to initialize a previously unknown map in monocular camera setup because of the two-way dependency: for determination of a 3D points depth from the camera center, at least two different poses have to be known while for visual pose computation, the 3D point locations and their corresponding 2D projections are needed.

There are several approaches for map initialization. Davison [9] uses a particle filter-like approach and generate depth hypotheses for the 2D features observed in the first frame. The depth hypotheses are tested with each new frame by projecting the 3D point for each hypothesis and comparing the projection with the tracked 2D point location. After the point depth can be safely approximated as a Gaussian distribution, the point is added to the map.

Klein and Murray [33], on the other hand, explicitly ask the user to move the camera for map initialization. After the user signals that the movement of the camera is finalized, by assuming that the translation of the camera is 10 centimeters in order to create a map with a metric scale, 3D feature points are initialized by triangulation of 2D feature pairs observed in the first and the latest frames. The initial map is refined via bundle adjustment as new frames arrive.

When inertial measurements are available, an estimate for the camera pose can be computed without the map being unavailable in the beginning. By computing the relative pose between the first and n^{th} frames by integration of accelerometer and gyroscope measurements, 3D locations of the tracked 2D points can be found via triangulation, as formulated in Appendix C.

In the proposed system, the user is asked to move the camera horizontally for one seconds for initialization. The reasons for the restricted movement are:

- In order to initialize 3D point locations with high-accuracy, the rays from the two camera centers to the triangulated point should be separated with a sufficiently large angle [22] (see Figure 2.3). This means that the system should allow sufficient time for the user to move the camera.
- As analyzed in Section 3.1.1.1, inertial measurements tend to diverge rapidly due to the bias on them. This means that the pose of the device can not be effectively estimated after even a short time interval by using inertial measurements only. 3D points found by highly noisy initial pose estimates damage the system operation dramatically. By restricting the motion to a single dimension, only readings on one axis of the accelerometer are integrated. This way, the error is trapped to be in only one direction, affecting only the scale of the pose measurement as explained below.

2.2.1.1 Computation of translation during map initialization

We set the world coordinate system as the initial pose of the camera. In the camera coordinate system, the principal axis of the camera is defined as the $z-axis$ while the $x-$ and $y-axes$ define the principal plane, the $x-axis$ being the horizontal component. Hence, the initial pose can be represented as:

$$\vec{\phi}_1 = [\vec{\tau}_1^T \mid \vec{q}_1^T]^T = [0\ 0\ 0 \mid 1\ 0\ 0\ 0]^T \tag{2.15}$$

where $\vec{q}_1$ represents the attitude in quaternion representation such that $\tilde{q}_1 = 1$. By restricting the motion to be only in the $x-axis$, the pose at the n_i^{th} frame, corresponding to time t_i, can be approximated as:

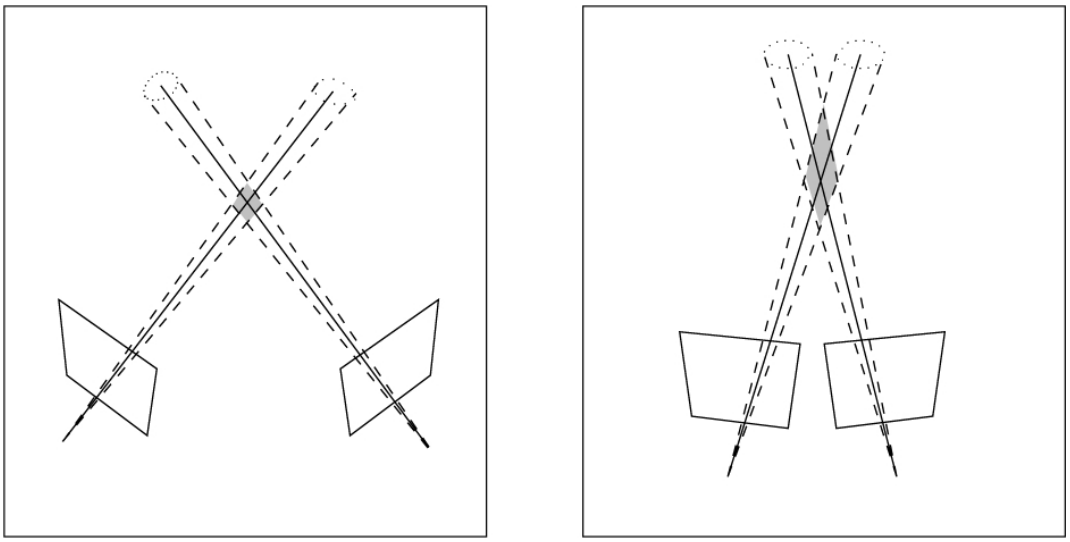


Figure 2.3: Uncertainty on 3D point positions after triangulation (image taken from [22]). Note that higher the angle between two rays from the camera center, lower the uncertainty on the 3D feature position.

$$\vec{\phi}_{n_i} = [\hat{\tau}_{x,n_i}\ 0\ 0 \mid 1\ 0\ 0\ 0]^T \tag{2.16}$$

$\hat{\tau}_{x,n_i}$ is the translation estimated using the accelerometer readings by integration until time t_i. Assuming that the initial velocity is zero, we can write $\hat{\tau}_{x,n_i}$ as sum of displacements during each interval between inertial readings:

$$\hat{\tau}_{x,n_i} = \sum_{k=0}^{k_i} \tau_{x,k} \tag{2.17}$$

Here, k_i represents the index of the last inertial measurement during the initialization time. We will drop the subscript x here for a more clean mathematical derivation. The displacements for each time interval can be written as:

$$\tau_k = v_k t_s + \frac{1}{2}\alpha_k t_s^2 \tag{2.18}$$

t_s represents the time between two inertial readings such that $k_i \times t_s \approx t_i$. By writing the velocity v_k in terms of the preceding accelerometer readings α_j, the previous equation becomes:

$$\tau_k = \sum_{j=0}^{k-1} (\alpha_j t_s) t_s + \frac{1}{2}\alpha_k t_s^2 \tag{2.19a}$$

$$= \left(\sum_{j=0}^{k-1} \alpha_j + \frac{1}{2}\alpha_k \right) t_s^2 \tag{2.19b}$$

Plugging the above equation into (2.17) results in:

$$\hat{\tau}_{x,n_i} = \sum_{k=0}^{k_i} \left(\sum_{j=0}^{k-1} \alpha_j + \frac{1}{2}\alpha_k \right) t_s^2 \tag{2.20a}$$

$$= \left(\sum_{k=0}^{k_i} \sum_{j=0}^{k-1} \alpha_j + \frac{1}{2} \sum_{k=0}^{k_i} \alpha_k \right) t_s^2 \tag{2.20b}$$

$$= \left(\sum_{k=0}^{k_i} (k_i - k)\alpha_k + \frac{1}{2} \sum_{k=0}^{k_i} \alpha_k \right) t_s^2 \tag{2.20c}$$

$$= \sum_{k=0}^{k_i} (k_i + 0.5 - k)\alpha_k t_s^2 \tag{2.20d}$$

In order to compute the variance of the translation, let us represent the accelerometer reading α_k by the sum of the actual acceleration and the zero mean Gaussian additional white noise on sensor (as analyzed in Section 3.1.1.2) such that $\alpha_k = a_k + e_k$. Then:

$$E\{\alpha_k\} = E\{a_k + e_k\} \tag{2.21a}$$

$$= E\{a_k\} + E\{e_k\} \tag{2.21b}$$

$$= a_k \tag{2.21c}$$

$$E\left\{(\alpha_k - E\{\alpha_k\})^2\right\} = E\left\{(a_k + e_k - a_k)^2\right\} \tag{2.22a}$$

$$= E\left\{e_k^2\right\} \tag{2.22b}$$

$$= \sigma_{e_k}^2 \tag{2.22c}$$

The mean value of $\hat{\tau}_{n_i}$ is computed as:

$$E\{\hat{\tau}_{n_i}\} = E\left\{ \sum_{k=0}^{k_i} (k_i + 0.5 - k)\alpha_k t_s^2 \right\} \tag{2.23a}$$

$$= \sum_{k=0}^{k_i} E\{\alpha_k\} (k_i + 0.5 - k) t_s^2 \tag{2.23b}$$

$$= \sum_{k=0}^{k_i} a_k (k_i + 0.5 - k) t_s^2 \tag{2.23c}$$

$$= \tau_{n_i} \tag{2.23d}$$

The variance of $\hat{\tau}_{n_i}$ is then computed as:

$$\sigma^2_{\hat{\tau}_{n_i}} = E\left\{(\hat{\tau}_{n_i} - E\{\hat{\tau}_{n_i}\})^2\right\} \tag{2.24a}$$

$$= E\left\{\left(\sum_{k=0}^{k_i}(k_i+0.5-k)\alpha_k t_s^2 - \sum_{k=0}^{k_i} a_k(k_i+0.5-k)t_s^2\right)^2\right\} \tag{2.24b}$$

$$= E\left\{\left(\sum_{k=0}^{k_i}(k_i+0.5-k)(\alpha_k - a_k)t_s^2\right)^2\right\} \tag{2.24c}$$

$$= \sum_{k=0}^{k_i} E\left\{(\alpha_k - a_k)^2\right\}(k_i+0.5-k)^2 t_s^4 \tag{2.24d}$$

$$= \sum_{k=0}^{k_i} \sigma^2_{e_k}(k_i+0.5-k)^2 t_s^4 \tag{2.24e}$$

$$= \sigma^2_{e_k} t_s^4 \sum_{k=0}^{k_i}(k_i+0.5-k)^2 \tag{2.24f}$$

$$= \sigma^2_{e_k} t_s^4 \frac{(k_i+1)(4k_i^2+8k_i+3)}{12} \tag{2.24g}$$

2.2.1.2 Computation of scaling between estimated visual pose and metric scale

Uncertainty on $\hat{\tau}_{n_i}$ causes a scaling error between the true and estimated 3D coordinates of the mapped points. This results in estimation of visually estimated position of the camera at a slightly different scale during the system operation. This scaling is defined as:

$$\lambda \triangleq \frac{\|\tau\| \text{ measured visually}}{\text{metric } \|\tau\|} \tag{2.25}$$

Following the definition above, λ becomes:

$$\lambda = \frac{\hat{\tau}_{n_i}}{\tau_{n_i}} \tag{2.26}$$

Using (2.23), initial estimate of λ can be computed as:

$$E\{\lambda\} = E\left\{\frac{\hat{\tau}_{n_i}}{\tau_{n_i}}\right\} \tag{2.27a}$$

$$= \frac{E\{\hat{\tau}_{n_i}\}}{\tau_{n_i}} \tag{2.27b}$$

$$= \frac{\tau_{n_i}}{\tau_{n_i}} \tag{2.27c}$$

$$= 1 \tag{2.27d}$$

Uncertainty on the scale estimate is, then:

$$E\left\{(\lambda - E\{\lambda\})^2\right\} = \sigma_\lambda^2 \tag{2.28a}$$

$$= E\left\{(\frac{\hat{\tau}_{n_i}}{\tau_{n_i}} - 1)^2\right\} \tag{2.28b}$$

$$= E\left\{(\frac{\hat{\tau}_{n_i} - \tau_{n_i}}{\tau_{n_i}})^2\right\} \tag{2.28c}$$

$$= \frac{E\left\{(\hat{\tau}_{n_i} - \tau_{n_i})^2\right\}}{{\tau_{n_i}}^2} \tag{2.28d}$$

$$= \sigma_{\hat{\tau}_{n_i}}^2 / \tau_{n_i}^2 \tag{2.28e}$$

2.2.2 Map Handling

The initial map includes points only from the first n_i frames. As time progresses, the field of view of the camera is expected to change. As old points are lost and new points are introduced, the map should be altered in order to include newly explored areas.

Systems like [33] includes currently observed and previously observed points together, expanding the map as new areas are observed. This is a computationally heavy approach and effects the real time operation as time progresses. In the mobile version [34], some of the previous points are deleted from the map, keeping the map smaller than a defined maximum size to guarantee the needed speed.

In the proposed system, we choose to keep the map at a defined maximum size of m points consisting of only currently observed points while $l > m$ points are actively tracked. The map is refined every n_i frames and the points that have gone out of sight or lost in the tracking process are deleted from the map. The map size is preserved by triangulating the tracked (but not mapped) points and adding them to the map.

2.3 Pose Estimation from 2D-3D Point Correspondences

In this thesis, the phrase *visual measurement* corresponds to the position and the attitude - two of them simultaneously called *the pose* - of the camera as measured from 3D points with known locations in the world frame and their 2D correspondences on the image plane. The visual measurement is represented by $\vec{\phi}$.

$$\vec{\phi} = \begin{bmatrix} \vec{\tau} \\ \vec{q} \end{bmatrix} \tag{2.29}$$

Here $\vec{\tau}$ represents the translation from the world center to the camera center in three axes and $\vec{q}$ is composed of the four parameters of the rotation quaternion that represents the attitude of the camera.

$$\vec{\tau} = \begin{bmatrix} \tau_x & \tau_y & \tau_z \end{bmatrix}^T \tag{2.30}$$

$$\vec{q} = \begin{bmatrix} q_s & q_a & q_b & q_c \end{bmatrix}^T \Rightarrow \tilde{q} = q_s + q_a i + q_b j + q_c k \tag{2.31}$$

Assuming that the calibration matrix K of a camera is known, it is possible to determine the pose of the camera with respect to the world coordinates using 3D points in the world frame and their 2D correspondences on the image plane [22]. This problem is called the Perspective-n-Point (PnP) problem [41].

There are several algorithms to solve the PnP problem. the Direct Linear Transformation (DLT) algorithm, as presented in [22], finds the projection matrix P from at least 6 point correspondences. There are iterative methods, such as [52], that provides highly accurate estimates. Non-iterative methods, on the other hand, appear as faster while being less accurate when compared to iterative ones. Due to limited computational resources on mobile devices, we utilize a non-iterative algorithm called EPnP [60, 41]. It is reported in [41] that high speed can be achieved with a small sacrifice from accuracy.

Briefly, EPnP algorithm creates four virtual *control points* from linear combinations of available n 3D points in the map. The algorithm requires n to be at least 4. The projection equations in the camera coordinate frame are written using the control points. The equations are used to define a matrix, the nullspace of which will then lead to the solution [60]. The result can be further refined using Gauss-Newton optimization [41].

2.3.1 Uncertainty on Estimated Pose

The constructed hybrid pose estimation system requires covariance of the visual measurement due to its stochastic nature. In order to compute the covariance Φ of the estimated visual pose ϕ, the method presented in [23] is adopted.

The projection function that maps a 3D point $\vec{X} = [X\ Y\ Z]^T$ onto the image plane with coordinates $\vec{x} = [x\ y]^T$ is defined as:

$$\vec{x} = p(\vec{\phi}, \vec{X}) \tag{2.32}$$

Using Taylor expansion, we can linearize $p(\vec{\phi}, \vec{X})$ around $\vec{\phi}_n$ and $\vec{X}_i$ as shown in (2.33).

$$\vec{x}_i + \Delta\vec{x}_i = p(\vec{\phi}_n + \Delta\vec{\phi}, \vec{X}_i) \approx p(\vec{\phi}_n, \vec{X}_i) + J_{p(\vec{\phi}_n, \vec{X}_i)}(\vec{\phi})\Delta\vec{\phi} \tag{2.33}$$

which yields

$$\Delta\vec{x}_i \approx J_{p(\vec{\phi}, \vec{X}_i)}(\vec{\phi})\Delta\vec{\phi} \tag{2.34}$$

The Jacobian $J_{p(\vec{\phi}, \vec{X}_i)}$ is defined in Appendix D. We can concatenate (2.34) for every point $\vec{x}_i|_{i=1}^m$ and obtain:

$$\begin{bmatrix} \Delta\vec{x}_1 \\ \Delta\vec{x}_2 \\ \vdots \\ \Delta\vec{x}_m \end{bmatrix} = \begin{bmatrix} J_{p(\vec{\phi},\vec{X}_1)}(\vec{\phi}) \\ J_{p(\vec{\phi},\vec{X}_2)}(\vec{\phi}) \\ \vdots \\ J_{p(\vec{\phi},\vec{X}_m)}(\vec{\phi}=\vec{\phi}_n) \end{bmatrix} \Delta\vec{\phi} \Rightarrow \Delta\vec{x}_{cont} = J_{cont}\Delta\vec{\phi} \tag{2.35}$$

The least-squares solution of (2.35) for $\Delta\vec{\phi}$ yields

$$\Delta\vec{\phi} = J_{cont}^{\dagger}\Delta\vec{x}_{cont} \tag{2.36}$$

where $J_{cont}^{\dagger}$ is the *pseudoinverse* (or *generalized inverse*) of J_{cont} defined as

$$J_{cont}^{\dagger} = (J_{cont}^T J_{cont})^{-1} J_{cont}^T \tag{2.37}$$

Then, we can compute the covariance matrix of the visual measurement vector ϕ, using the point uncertainties Σ_i as:

$$\begin{aligned}
\Phi &= E\{\Delta\vec{\phi}\Delta\vec{\phi}^T\} && \text{(2.38a)}\\
&= E\{(J_{cont}^{\dagger}\Delta\vec{x}_{cont})(J_{cont}^{\dagger}\Delta\vec{x}_{cont})^T\} && \text{(2.38b)}\\
&= E\{J_{cont}^{\dagger}\Delta\vec{x}_{cont}\Delta\vec{x}_{cont}^T(J_{cont}^{\dagger})^T\} && \text{(2.38c)}\\
&= J_{cont}^{\dagger}E\{\Delta\vec{x}_{cont}\Delta\vec{x}_{cont}^T\}(J_{cont}^{\dagger})^T && \text{(2.38d)}\\
&= J_{cont}^{\dagger}\begin{bmatrix} \Sigma_1 & 0 & \dots & 0 \\ 0 & \Sigma_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & \Sigma_m \end{bmatrix}(J_{cont}^{\dagger})^T && \text{(2.38e)}
\end{aligned}$$

It is reported in [23] that the effect of the uncertainties on 3D point locations is negligible. Following that observation, in the computation of measurement covariance, uncertainty on the locations of 3D points are not taken into account.

2.4 Experimental Results

During the experiments for visual tracking and pose estimation, Grafitti and Boat images from Afine Covariant Features dataset [59] are used.

2.4.1 Gyroscope-aided 2D Tracking

In Section 2.1.3, we formulated the gyroscope-aided feature tracking, where new positions of the features are predicted using the gyroscope readings before the KLT tracking algorithm is

run. A typical result of this approach can be seen in Figure 2.4. As seen in the figure, the predicted points are closer to the actual new positions of the features. The resultant prediction sometimes result in almost exact positions, as seen in Figure 2.5. This case occurs when translation between two poses of the camera is actually small. When the translation is somewhat high or the gyroscope reading is erroneous, the prediction may result in slightly worse positions as seen in Figure 2.6.

In order to compare the performance of regular and gyroscope-aided tracking, we can look at the distribution of distances between the initial and final positions of the tracked points for the two cases. Figure 2.7 shows that distribution. We can conclude that use of gyroscopes effectively reduces the expected distance to the new feature position, making the tracking more robust against fast camera rotations, which was the expected outcome.

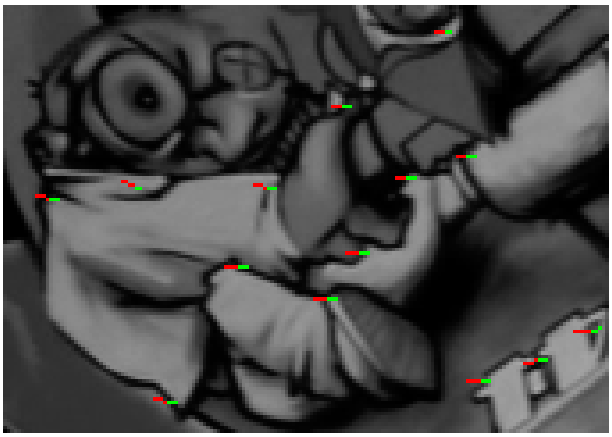


Figure 2.4: A typical example of the prediction of the next positions of the keypoints using gyroscope readings (Red lines are drawn between the initial position of a point and the predicted position while green lines are between the predicted position and the final position)

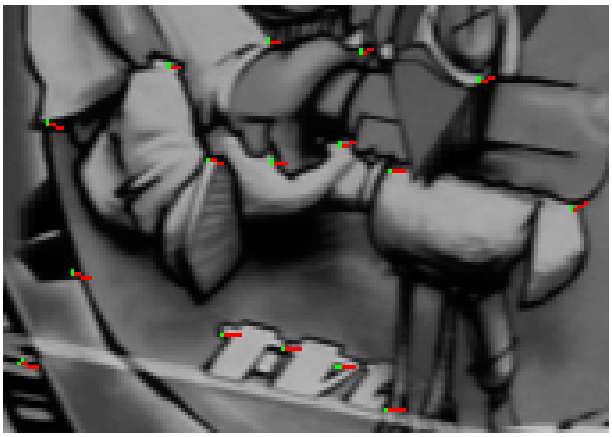


Figure 2.5: An ideal example of the prediction of the next positions of the keypoints using gyroscope readings (Red lines are drawn between the initial position of a point and the predicted position while green lines are between the predicted position and the final position)

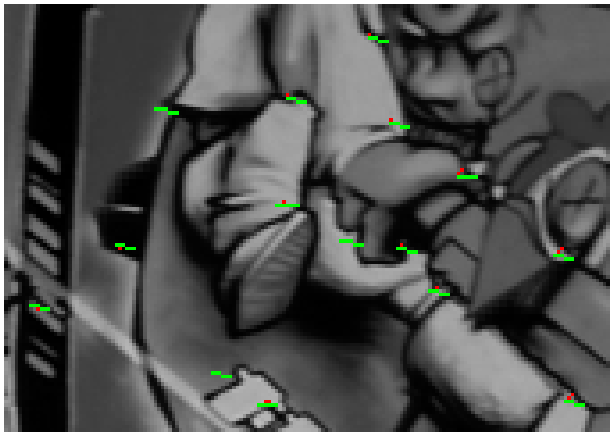


Figure 2.6: Worst-case scenario for the prediction of the next positions of the keypoints using gyroscope readings (Red lines are drawn between the initial position of a point and the predicted position while green lines are between the predicted position and the final position)

2.4.2 Visual Pose Estimation

In order to test the performance of the pose estimation, ground truth of four image sequences is prepared. Two of the sequences consist of complex camera motion while the others has mostly rotational and mostly translational motion. The ground truth is prepared by selecting four points with known 3D locations on each image by hand and computing the pose using the EPnP algorithm.

During the visual tests, if there are m points in the map, we track $m/2$ extra points in order to keep the map size constant when points in the map are lost due to tracking errors or the point going out of sight. So, if the performance of the visual pose estimation with a map size of 20 is reported, there are a total of 30 2D points tracked.

On a laptop computer, the time required for one visual pose estimation iteration for different map sizes are listed in Table 2.1.

Table 2.1: Average time required for visual pose and covariance computation for different map sizes

# of tracked points	Time (ms)
20 in map, 30 total	20.08
16 in map, 24 total	13.38
12 in map, 18 total	10.42
8 in map, 12 total	7.53
6 in map, 9 total	4.47

Figure 2.8 and 2.9 show a typical example of performance difference with different map sizes. One can observe that although the performances are similar in the beginning, as the tracked points and the map starts to get corrupted, maps with smaller sizes start to degenerate the pose estimation. It is observed that the quality of 3D points affects the the error performance of the

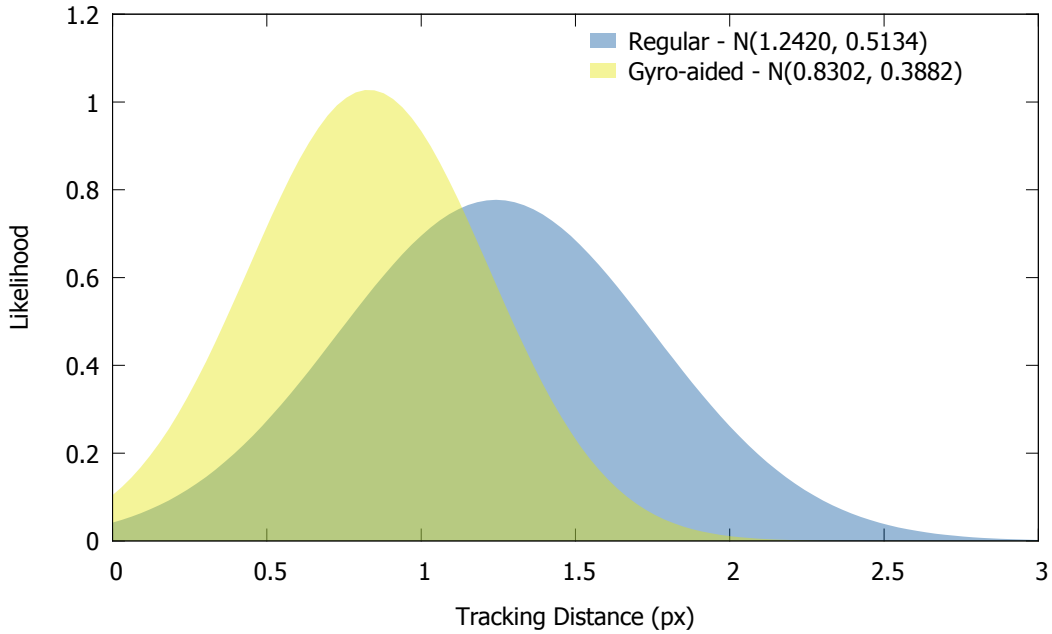


Figure 2.7: Histograms of the distances between the previous or the predicted positions and the final positions of tracked keypoints

translation and rotation equally.

The degeneration can be more dramatic when there is highly erroneously triangulated 3D points in the map. Figure 2.10 and Figure 2.11 shows that case. When there are small number of points in the map, new 3D points that are introduced to the map can degenerate the performance rapidly, while if the map is bigger, pose estimation is more robust, as expected.

2.4.3 Map Handling

Figure 2.12 describes the mapped points in different moments during the system operation. One can notice that the detected points are as distributed as possible, in order to get a better visual pose estimation.

One should note that when new points are introduced, they may be erroneously triangulated. Although it can be detected after some time, this problem may affect the visual pose estimation performance, as illustrated in Section 2.4.2.

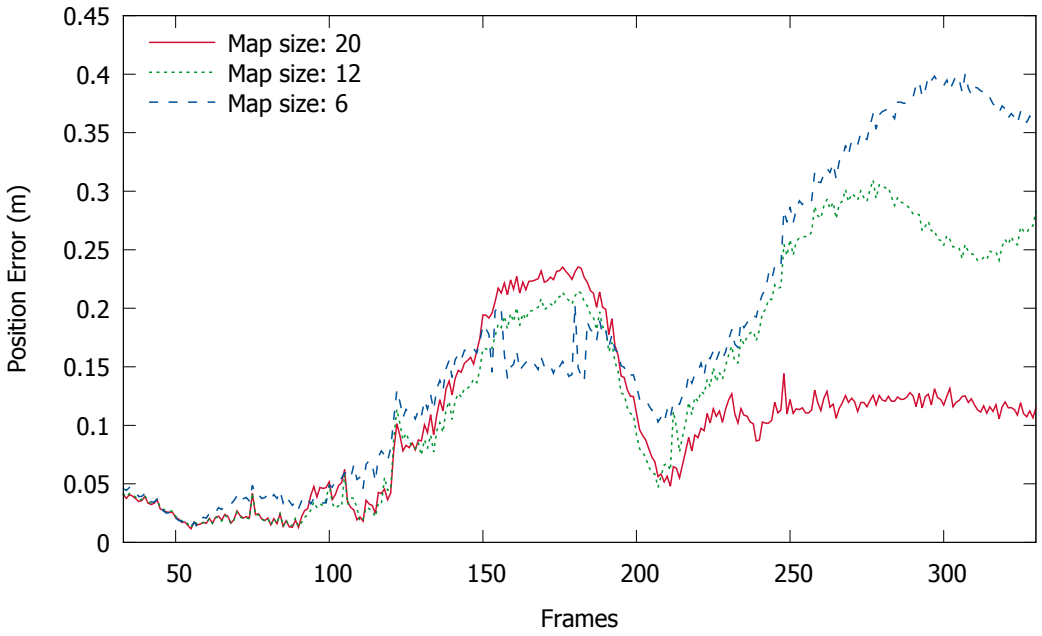


Figure 2.8: Position estimation performances with various map sizes

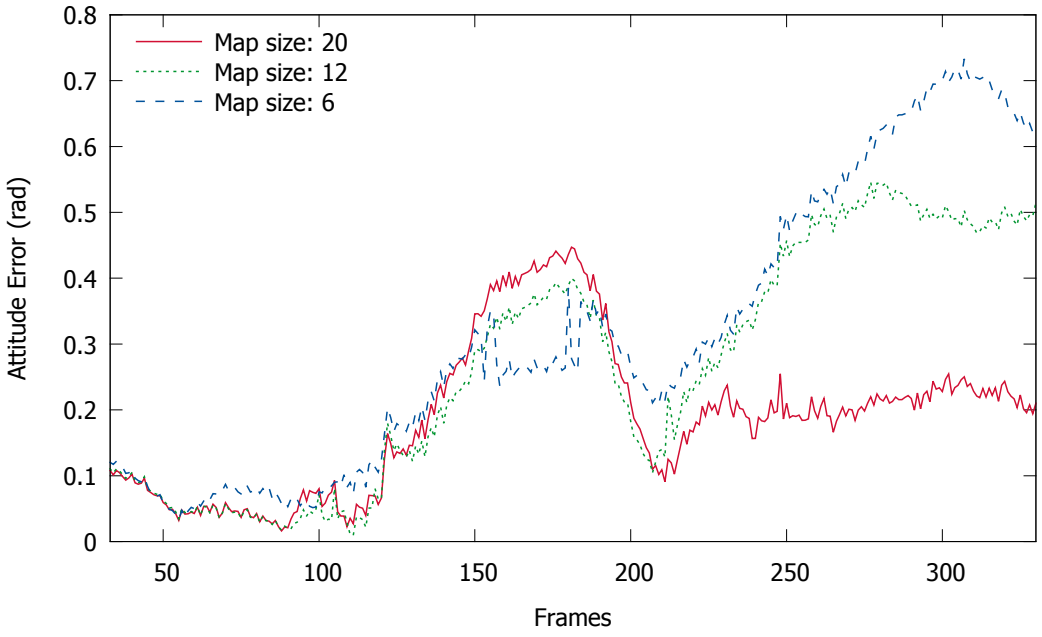


Figure 2.9: Attitude estimation performances with various map sizes

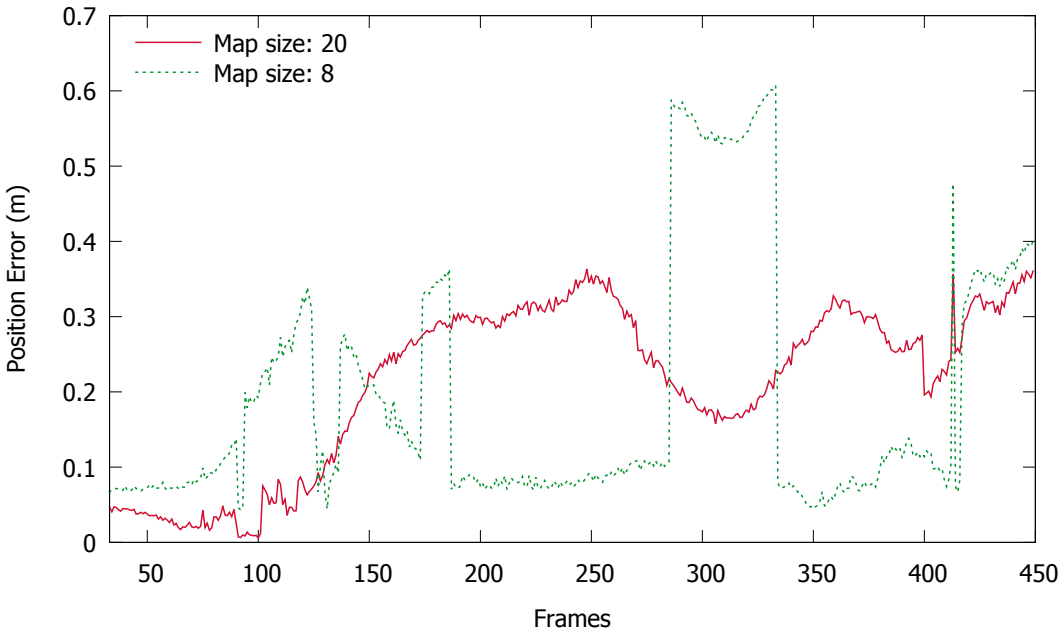


Figure 2.10: Position estimation performances with various map sizes

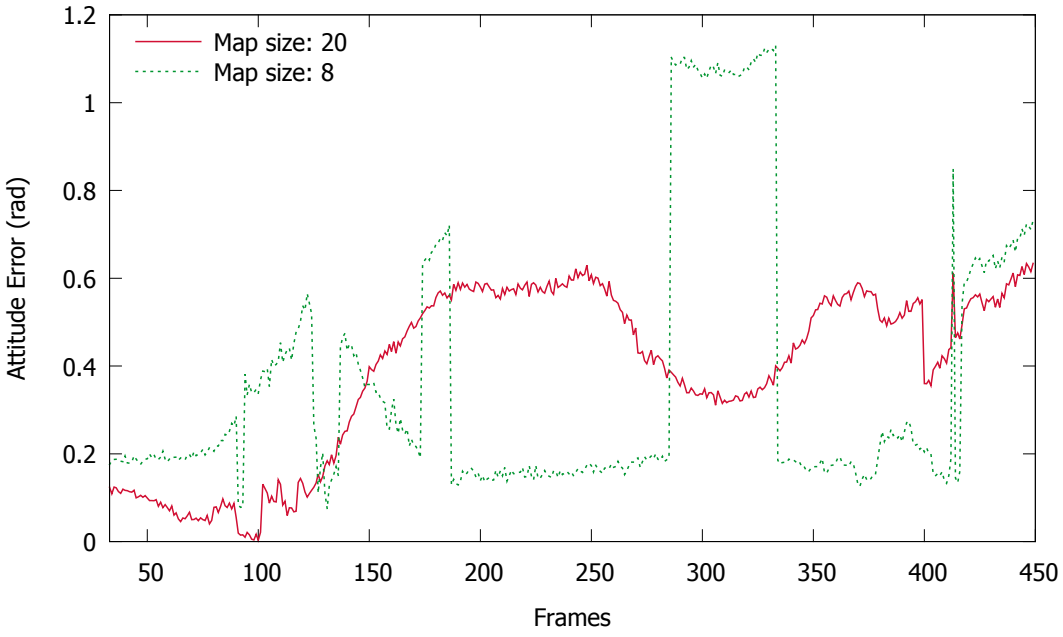


Figure 2.11: Attitude estimation performances with various map sizes

(a) Initial points in the map

(b) Tracked-only points start to appear in the newly discovered regions

(c) The map starts to move to the new regions

(d) The map is completely renewed after the initial points went out of sight

Figure 2.12: Different steps of map handling

CHAPTER 3

INERTIAL POSE ESTIMATION

We use the inertial sensor readings to compute the change in pose between consecutive frames. We utilize two inertial sensors: the accelerometer and the gyroscope. The accelerometer readings reflect the sum of the actual acceleration of the device and the gravity vector. We assume an accelerometer that only measures the acceleration of the device and eliminate the gravity vector from actual accelerometer measurements by using the gravity vector provided by Android system, which utilizes a combination of different sensors to estimate the direction of the gravity. The gyroscope measures the rotational velocity.

3.1 Inertial Sensors on Mobile Devices

Inertial navigation systems generally utilize standalone inertial sensors. Since the main purpose of inertial measurement units on mobile devices is detecting simple gestures, such as orientation change or shake, they are not generally well-suited for the odometry applications. It should be noted that characteristics of inertial measurement units on mobile devices are far worse than the ones used in inertial navigation systems. In this section, we analyze the noise characteristics of the sensors on ASUS TF201 Transformer Prime tablet, which is utilized in our experiments.

One of the shortcomings of inertial sensors on mobile devices is low sample rate. For example, the sensors used in [94] deliver readings at 1 kHz, or the ones used in [17] has the sample rate of 1.92 kHz while the mobile device used in [61] records readings at 100 Hz. The device utilized in our system has the sample rate of only 48 Hz.

The other shortcomings, which can be listed as highly varying bias and low signal-to-noise ratio, are analyzed in this section.

It should be noted that we utilize the inertial sensors in an augmented reality system for small workspaces. This comes with the characteristic that the translational motion is very small when compared to odometry systems for cars or autonomous robots. As analyzed below, since noise on accelerometers are modeled as an additional Gaussian noise that is independent of the actual acceleration, small actual acceleration results in a lower signal-to-noise ratio (SNR) which makes it harder to extract information from the sensors.

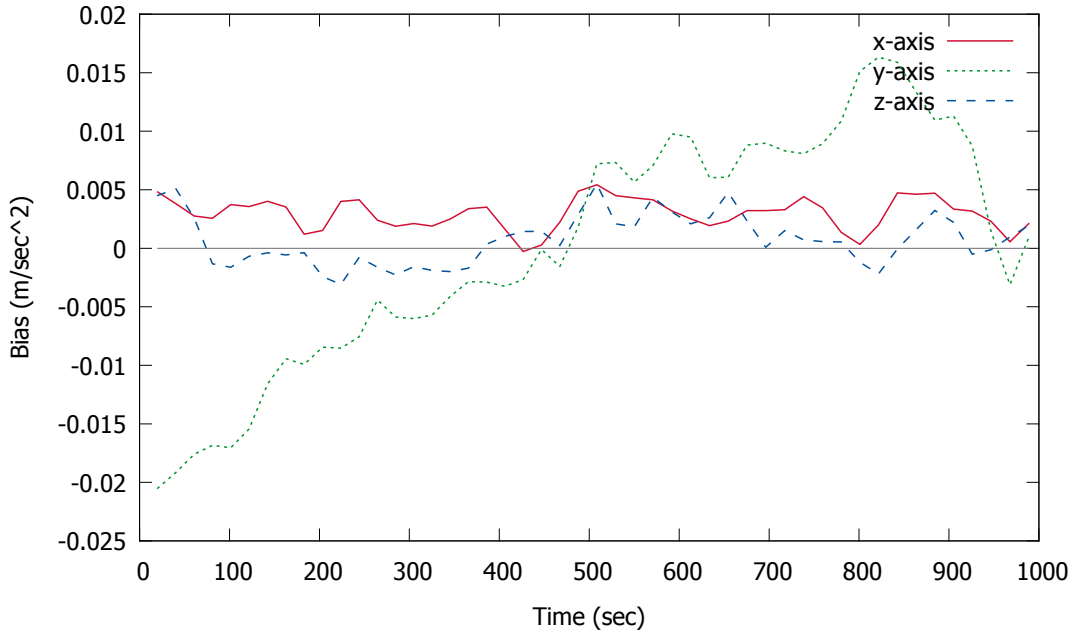


Figure 3.1: Bias terms on linear accelerometer readings

3.1.1 Noise Characteristics

Noise characteristics of the accelerometer and the gyroscope of the utilized mobile device are found experimentally. The device is left resting on a still surface for 20 minutes. Since the actual motion is known to be zero, the readings represent the noise on the sensors.

Firstly, the sampling rate of the sensors are found to be 48.53 samples per second. Note that this rate is only slightly higher than the 30 frames per second operation of the camera.

3.1.1.1 Bias Analysis

Bias is defined as an additional non-zero term on the noise, which is particularly harmful in the long term operation. As integration of the bias diverges with time, it should be carefully handled in inertial navigation systems.

When bias term is constant on a sensor, it can easily be eliminated by listening to the sensor in resting condition and determining the bias as the mean value of samples. However, generally, bias changes with sensor turn-on and then varies slowly over time. This requires tracking of bias on the sensors as done in [11, 67]. Tracking of 6 independent biases on 3-axis linear accelerometer and 3-axis gyroscope measurements comes with the price of increased mathematical and computational complexity.

The bias terms on the linear accelerometer and gyroscope are analyzed by taking the average of 1000 consecutive samples. The plots showing the variation of bias terms with time on two sensors can be seen in Figure 3.1 and 3.2.

Due to the highly varying nature of bias on inertial measurements of the mobile device, it is

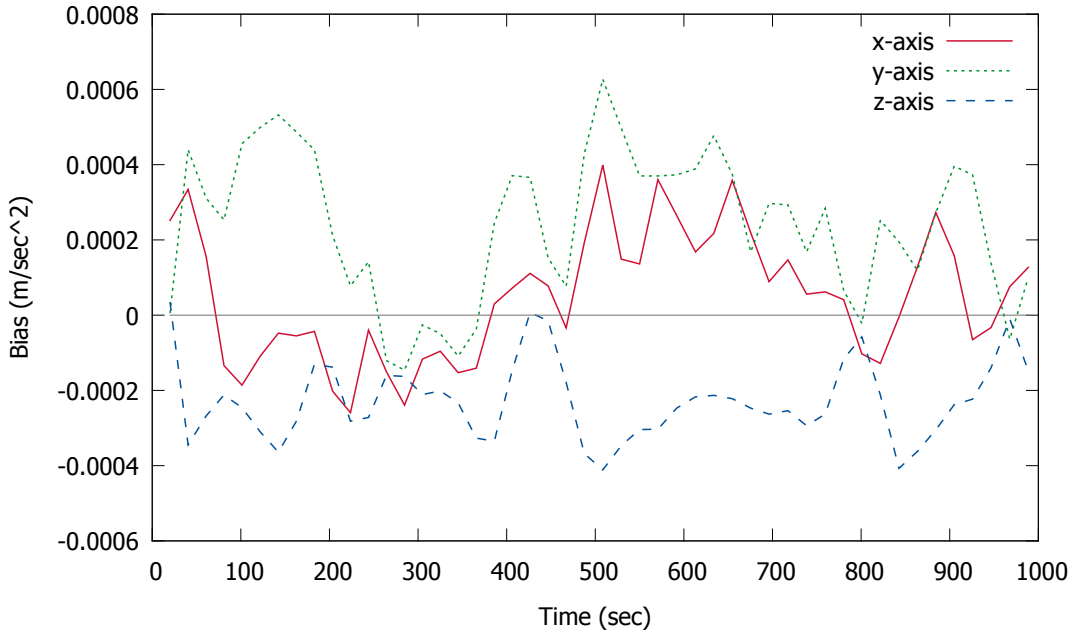


Figure 3.2: Bias terms on gyroscope readings

not an easy task to track them. To prevent the bias terms from causing the inertial navigation to diverge [26], we utilize the inertial measurements only in small time intervals, reducing the effect of the bias terms and hence reducing the need of determination of instantaneous bias terms on sensor readings.

3.1.1.2 Distribution of Noise

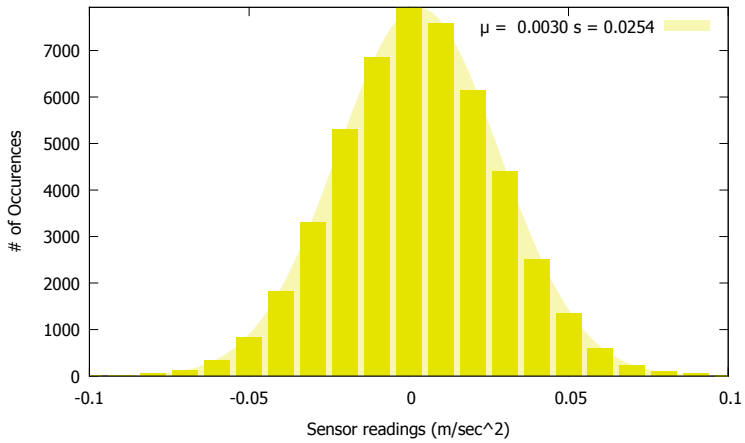
Figure 3.3 and 3.4 show the histogram of sensor readings and their corresponding Gaussian approximations.

Notice that each of the distributions can be effectively approximated by a Gaussian. This is a very important result for the operation of the proposed system because the utilized Kalman filter (see Appendix A) explicitly requires Gaussian distributed measurements for proper operation.

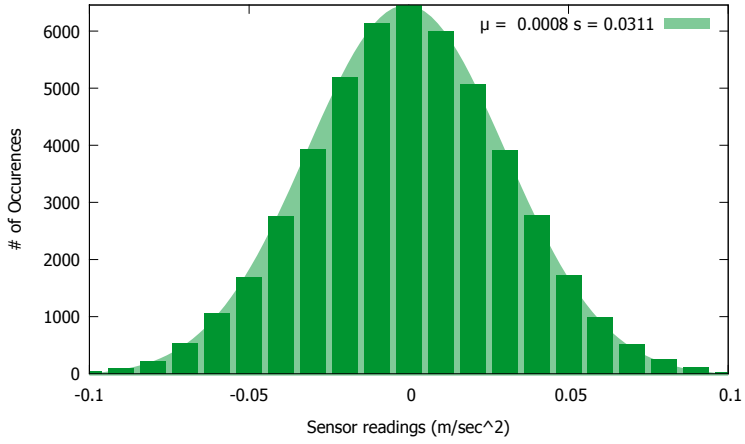
Variance of the noise on the sensors are presented in Table 3.1. From these values, covariance matrix A of accelerometer readings $\vec{\alpha}$ and covariance matrix Ω of gyroscope readings $\vec{\omega}$ can be defined as:

$$A = \begin{bmatrix} 0.0254^2 & 0 & 0 \\ 0 & 0.0311^2 & 0 \\ 0 & 0 & 0.0238^2 \end{bmatrix} \tag{3.1}$$

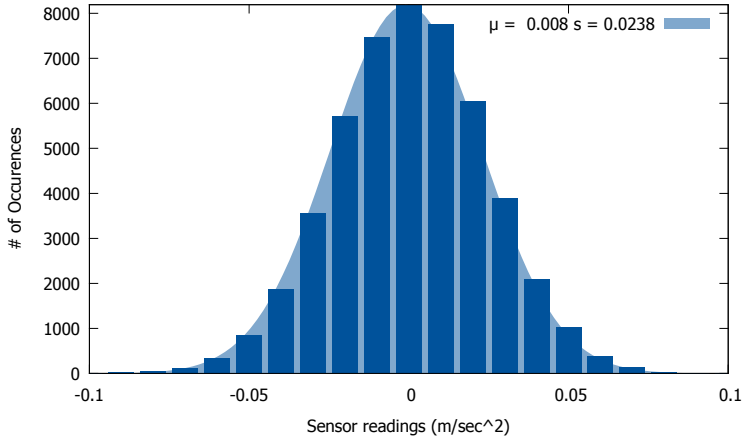
$$\Omega = \begin{bmatrix} 0.0011^2 & 0 & 0 \\ 0 & 0.0008^2 & 0 \\ 0 & 0 & 0.0007^2 \end{bmatrix} \tag{3.2}$$



(a) $x-axis$

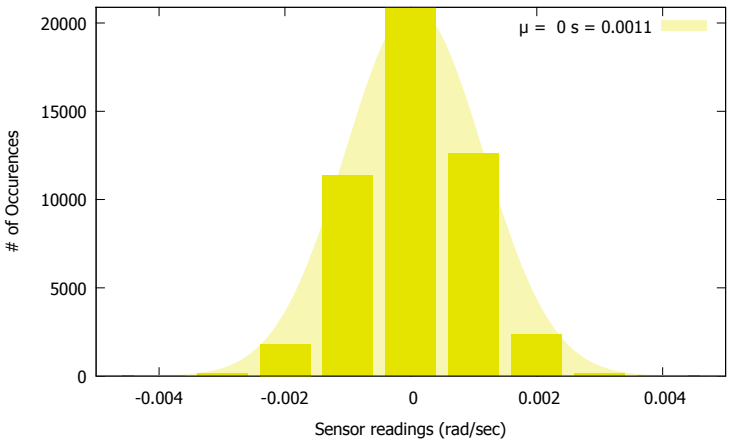


(b) $y-axis$

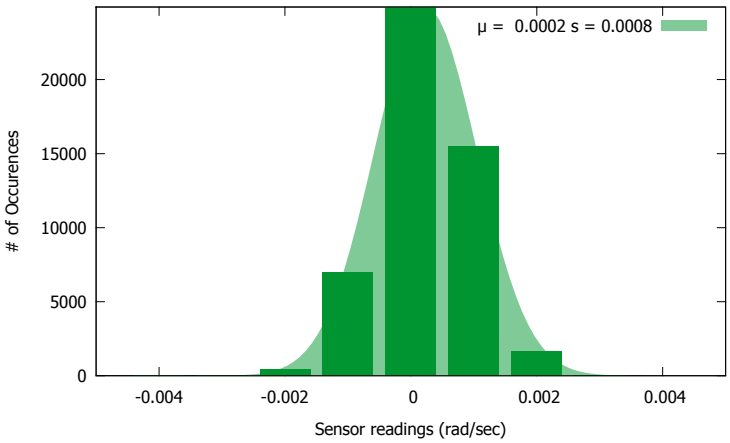


(c) $z-axis$

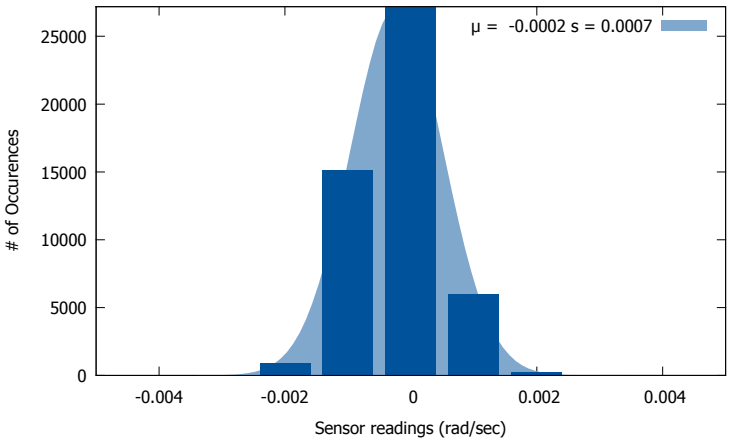
Figure 3.3: Distribution of linear accelerometer readings



(a) $x-axis$



(b) $y-axis$



(c) $z-axis$

Figure 3.4: Distribution of gyroscope readings

Table 3.1: Standard deviance of noise on the accelerometer and the gyroscope

	$x-axis$	$y-axis$	$z-axis$
Accelerometer (m/s^2)	0.0254	0.0311	0.0238
Gyroscope (rad/s)	0.0011	0.0008	0.0007

3.1.1.3 Power Spectral Density Estimation

Power spectral density (PSD) of a signal represents the frequency content of that signal. If y_n denotes a random signal, PSD can be formally defined as:

$$\phi(\omega) = \lim_{N\to\infty} E\left\{\frac{1}{N}\left|\sum_{n=1}^{N} y_n e^{-i\omega n}\right|^2\right\} \tag{3.3}$$

There are several different algorithms for estimating the PSD of a random signal from its samples. The algorithms can be categorized under two main categories: parametric and non-parametric [80]. Parametric approaches need prior knowledge on the shape of the PSD. Non-parametric methods, on the other hand, assumes no underlying structure of the PSD, causing a decrease in the accuracy.

Since we want to estimate the PSD of the noise signals without a prior knowledge, a nonparametric method known as the *Welch method* [92] is used. This method computes the PSD in overlapping windows of samples of the random sequence and then estimates the actual PSD by taking the average of the computed PSD's of overlapping windows.

The resultant PSD's of accelerometer and gyroscope readings can be seen in Figure 3.5 and 3.6.

As it can be seen from the PSD's, the noise signal is white except for a DC term, which represents the bias. This is an important result for the operation of the Kalman filter because the filter requires Markov property, which states that the measurements and the inputs to the system should depend only on the previous sample and not the ones before them.

3.2 External Calibration of the IMU and the Camera

In order to aid visual pose measurements, inertial sensor readings have to be transformed to camera coordinate frame. The relative translation and rotation between two sensors are called *external calibration* parameters. In the literature, there are several online, such as [13], and offline methods, such as [50, 24, 29, 30]. These algorithms are utilized when there is an ambiguity in the external calibration parameters.

In our case, since we utilize a COTS mobile device, the difference between two coordinate frames are fixed and known. Although there may be small shifts in the coordinate frames due to the manufacturing process, we will assume their effect on our system operation is

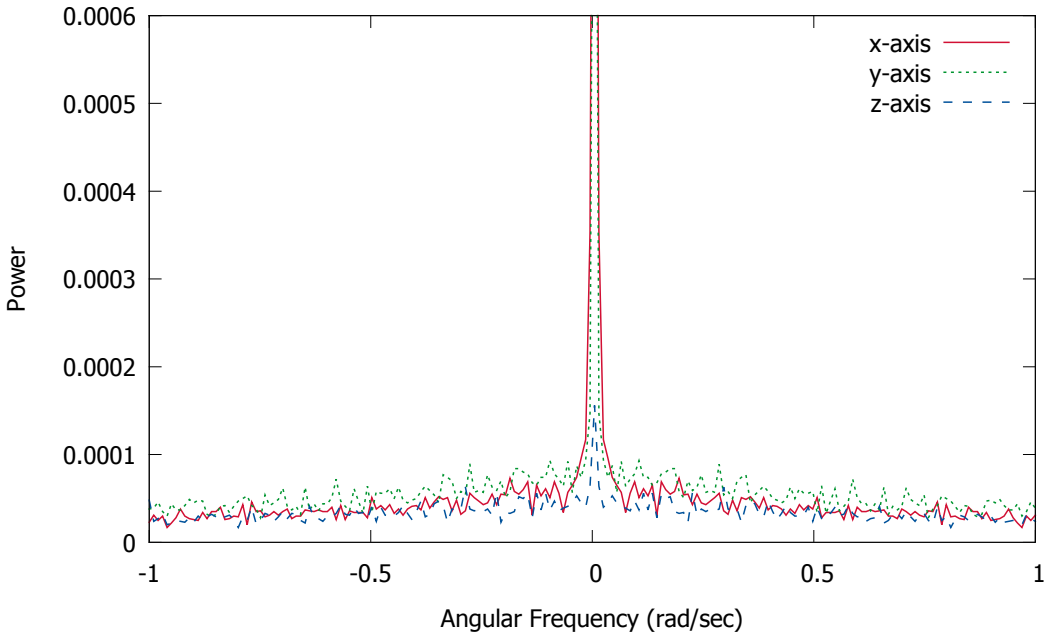


Figure 3.5: Power spectral density of linear accelerometer readings

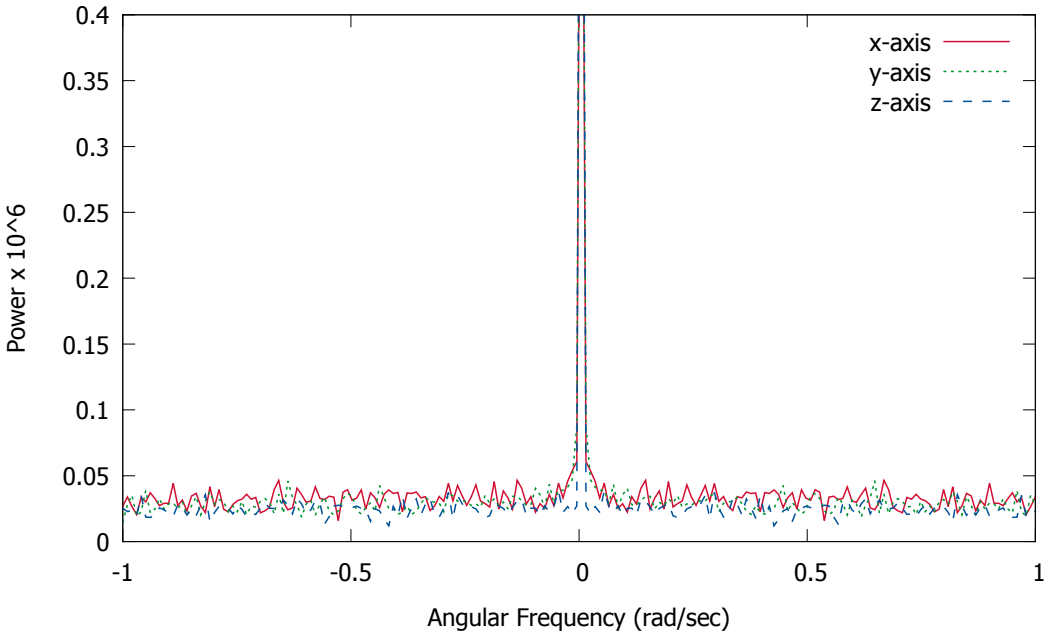


Figure 3.6: Power spectral density of gyroscope readings

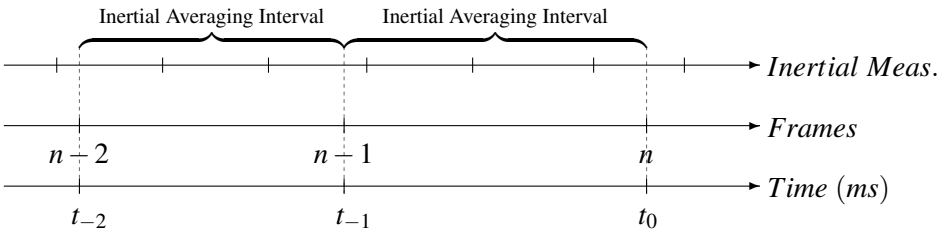


Figure 3.7: Visualization of asynchronous operation

negligible.

3.3 Metric Translational Velocity Computation

In order to compute relative position change between two time instances, in addition to the accelerometer readings, translational velocity of the camera is needed. In this section, we present a method to estimate the instantaneous velocity.

Kneip *et al.* [35] formulated the computation of metric translational velocity from accelerometer measurements and a visually tracked point in three frames. In this section, we adopt their formulation with several small differences in order to derive the equation relating camera velocity with inertial and visual measurements. Following the formulation, computation of the covariance matrix of the estimated velocity is presented.

In [35], the sample rate of the accelerometer is assumed to be much greater than frame rate of the image stream and a recursive formulation is developed for integration of accelerometer readings. Since two rates are similar in our case (48 Hz accelerometer and 30 Hz image stream), for a reasonable formulation, we assume that accelerometer readings are taken simultaneously with each image. We compute the accelerometer reading corresponding to an image by simply taking weighted average of accelerometer readings in the time interval between two consecutive frames as shown in Figure 3.7.

As it can be seen in Figure 3.8, three poses of the camera, $\vec{\phi}_n$, $\vec{\phi}_{n-1}$ and $\vec{\phi}_{n-2}$ at time instants t_n, t_{n-1} and t_{n-2}, are considered. Time intervals are defined as $t_1 = t_n - t_{n-1}$, $t_2 = t_n - t_{n-2}$ and $t_3 = t_{n-1} - t_{n-2}$. Accelerometer readings $\vec{\alpha}_{n-1}$ and $\vec{\alpha}_{n-2}$ are retrieved at t_{n-1} and t_{n-2}. Instantaneous velocities are denoted as $\vec{v}_n$, $\vec{v}_{n-1}$ and $\vec{v}_{n-2}$. Accelerations and velocities are represented in the coordinate frames corresponding to the poses at their time instant. The rotation from $\vec{\phi}_{n-1}$ to $\vec{\phi}_n$ is defined as $\tilde{q}_1$, from ϕ_{n-2} to ϕ_n as q_2 and from ϕ_{n-2} to ϕ_{n-1} as q_3, *i.e.* $q_2 = q_1 q_3$. These rotation quaternions are assumed to be known and can be computed from visual measurements or gyroscope readings. The translation from ϕ_n to ϕ_{n-1} is defined as τ_1, from ϕ_n to ϕ_{n-2} as τ_2 and from ϕ_{n-1} to ϕ_{n-2} as τ_3. Translations are defined in the latest coordinate frame.

We use the vector and quaternion representation of the variables interchangeably such that:

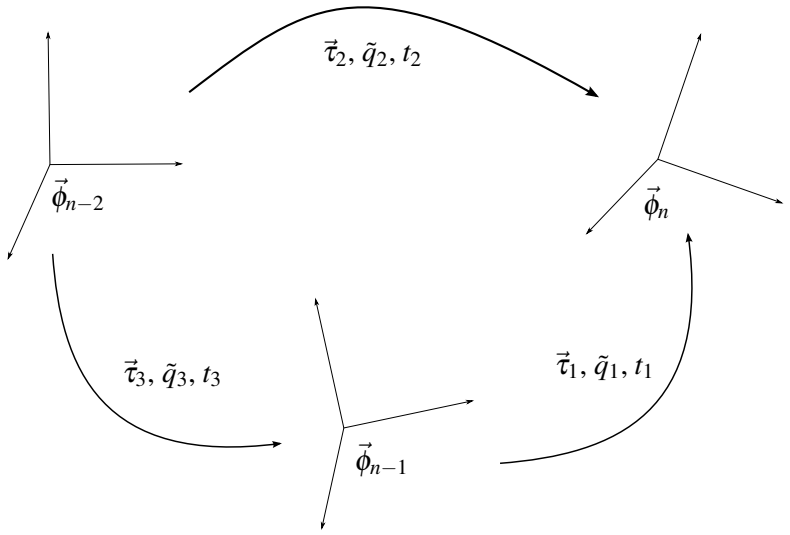


Figure 3.8: Definition of three poses and relative transformations

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \vec{\alpha} \Leftrightarrow \tilde{\alpha} = \alpha_1 i + \alpha_2 j + \alpha_3 k \tag{3.4}$$

$$\begin{bmatrix} q_s \\ q_a \\ q_b \\ q_c \end{bmatrix} = \vec{q} \Leftrightarrow \tilde{q} = q_s + q_a i + q_b j + q_c k \tag{3.5}$$

We want to determine $\vec{v}_n$ using the accelerations $\vec{a}_{n-1}$ and $\vec{a}_{n-2}$, locations $\vec{x}_n$, $\vec{x}_{n-1}$ and $\vec{x}_{n-2}$ of a tracked point in three frames and attitudes $\tilde{q}_n$, $\tilde{q}_{n-1}$ and $\tilde{q}_{n-2}$ of three frames. Let us start by representing $\vec{v}_{n-1}$ using the known and target variables.

$$\vec{v}_{n-1} = \tilde{q}_1^* \tilde{v}_n \tilde{q}_1 - \vec{a}_{(n-1)} t_1 \tag{3.6}$$

We can represent $\vec{v}_{n-1}$ in the latest coordinate frame as $\vec{v}_{(n-1|n)}$ such that $\vec{v}_{(n-1|n)} = \tilde{q}\tilde{v}_{n-1}\tilde{q}^*$.

$$\vec{v}_{(n-1|n)} = \vec{v}_n - \tilde{q}_1 \tilde{a}_{n-1} \tilde{q}_1^* t_1 \tag{3.7a}$$

$$= \vec{v}_n - \vec{a}_{(n-1|n)} t_1 \tag{3.7b}$$

The translation $\vec{\tau}_{(1|n-1)}$ can be computed as [35]:

$$\vec{\tau}_{(1|n-1)} = \vec{v}_{n-1} t_1 + \frac{1}{2} \vec{a}_{n-1} t_1^2 \tag{3.8}$$

Then $\vec{\tau}_1$ becomes:

$$\vec{\tau}_1 = \vec{v}_{(n-1|n)} t_1 + \frac{1}{2}\vec{a}_{(n-1|n)} t_1^2 \tag{3.9}$$

By substituting (3.7b) in (3.9), we get:

$$\vec{\tau}_1 = \vec{v}_n t_1 - \vec{a}_{(n-1|n)} t_1^2 + \frac{1}{2}\vec{a}_{(n-1|n)} t_1^2 \tag{3.10a}$$

$$= \vec{v}_n t_1 - \frac{1}{2}\vec{a}_{(n-1|n)} t_1^2 \tag{3.10b}$$

Similarly, $\vec{\tau}_{(3|n-1)}$ is:

$$\vec{\tau}_{(3|n-1)} = \vec{v}_{n-1} t_3 - \frac{1}{2}\vec{a}_{(n-2|n-1)} t_3^2 \tag{3.11}$$

$$\vec{\tau}_3 = \tilde{q}_1 \tilde{\tau}_{(3|n-1)} \tilde{q}_1^* \tag{3.12a}$$

$$= \vec{v}_{(n-1|n)} t_3 - \frac{1}{2}\vec{a}_{(n-2|n)} t_3^2 \tag{3.12b}$$

$$= \vec{v}_n t_3 - \vec{a}_{(n-1|n)} t_1 t_3 - \frac{1}{2}\vec{a}_{(n-2|n)} t_3^2 \tag{3.12c}$$

We can then compute $\vec{\tau}_2$ as:

$$\vec{\tau}_2 = \vec{\tau}_1 + \vec{\tau}_3 \tag{3.13a}$$

$$= \vec{v}_n (t_1 + t_3) - \vec{a}_{(n-1|n)} (\frac{1}{2} t_1^2 + t_1 t_3) - \vec{a}_{(n-2|n)} (\frac{1}{2} t_3^2) \tag{3.13b}$$

$$= \vec{v}_n t_2 - \vec{a}_{(n-1|n)} (\frac{1}{2} t_1^2 + t_1 t_3) - \vec{a}_{(n-2|n)} (\frac{1}{2} t_3^2) \tag{3.13c}$$

When we define $\vec{\eta}_1$ and $\vec{\eta}_2$ as:

$$\vec{\eta}_1 = -\frac{1}{2}\vec{a}_{(n-1|n)} t_1^2 \tag{3.14a}$$

$$\vec{\eta}_2 = -\vec{a}_{(n-1|n)} (\frac{1}{2} t_1^2 + t_1 t_3) - \vec{a}_{(n-2|n)} (\frac{1}{2} t_3^2) \tag{3.14b}$$

$\vec{\tau}_1$ and $\vec{\tau}_2$ simply becomes:

$$\vec{\tau}_1 = \vec{v}_n t_1 + \vec{\eta}_1 \tag{3.15a}$$

$$\vec{\tau}_2 = \vec{v}_n t_2 + \vec{\eta}_2 \tag{3.15b}$$

We now have the translation between three frames in terms of the known variables and the desired velocity vector. Now, in order to relate the kinematic equations with the tracked interest point, we will relate the point locations in three frames with each other. Firstly, let us define *normalized image coordinates* as:

$$z'\vec{x}' = z' \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.16}$$

where K is the camera calibration matrix and x and y are the regular image coordinates. This representation of point locations is called *normalized* because the calibration matrix corresponding to the new coordinates is an identity matrix.

We can relate the normalized coordinates of the point at n^{th} and $(n-1)^{th}$ frames as [35]:

$$z_{n-1}\vec{x}'_{n-1} = \tilde{q}_1^*(z_n\tilde{x}'_n + \tilde{\tau}_1)\tilde{q}_1 \tag{3.17}$$

When we plug in the value of $\vec{\tau}_1$, (3.17) becomes:

$$z_{n-1}\vec{x}'_{n-1} = \tilde{q}_1^*(z_n\tilde{x}'_n + \tilde{v}_n t_1 + \tilde{\eta}_1)\tilde{q}_1 \tag{3.18}$$

Let us separate three components of the velocity vector using quaternions $\tilde{q}_x = i$, $\tilde{q}_y = j$ and $\tilde{q}_z = k$.

$$\begin{aligned} z_{n-1}\vec{x}'_{n-1} &= \tilde{q}_1^*(z_n\tilde{x}'_n + \tilde{q}_x v_x t_1 + \tilde{q}_y v_y t_1 + \tilde{q}_z v_z t_1 + \tilde{\eta}_1)\tilde{q}_1 && \text{(3.19a)} \\ &= z_n\tilde{q}_1^*\tilde{x}'_n\tilde{q}_1 + v_x t_1\tilde{q}_1^*\tilde{q}_x\tilde{q}_1 + v_y t_1\tilde{q}_1^*\tilde{q}_y\tilde{q}_1 + v_z t_1\tilde{q}_1^*\tilde{q}_z\tilde{q}_1 + \tilde{q}_1^*\tilde{\eta}_1\tilde{q}_1 && \text{(3.19b)} \end{aligned}$$

Let us represent the variables that are rotated by $\tilde{q}_1^*$ with 1^* as an extra subscript, such that $\tilde{q}_1^*\tilde{x}'_n\tilde{q}_1 = \tilde{x}'_{n|1^*}$. Then, the last equation becomes:

$$z_{n-1}\vec{x}'_{n-1} = z_{n-1} \begin{bmatrix} x'_{n-1} \\ y'_{n-1} \\ 1 \end{bmatrix} = z_n\tilde{x}'_{n|1^*} + v_x t_1\tilde{q}_{x|1^*} + v_y t_1\tilde{q}_{y|1^*} + v_z t_1\tilde{q}_{z|1^*} + \tilde{\eta}_{1|1^*} \tag{3.20}$$

Components of the interest point location, x'_{n-1} and y'_{n-1} can be separately computed using the above equation. If we represent the individual components of a quaternion with $[q]_s$, $[q]_a$, $[q]_b$ and $[q]_c$, x'_{n-1} and y'_{n-1} can be computed as below [35].

$$x'_{n-1} = \frac{z_n[\tilde{x}'_{n|1^*}]_a + v_x t_1[\tilde{q}_{x|1^*}]_a + v_y t_1[\tilde{q}_{y|1^*}]_a + v_z t_1[\tilde{q}_{z|1^*}]_a + [\tilde{\eta}_{1|1^*}]_a}{z_n[\tilde{x}'_{n|1^*}]_c + v_x t_1[\tilde{q}_{x|1^*}]_c + v_y t_1[\tilde{q}_{y|1^*}]_c + v_z t_1[\tilde{q}_{z|1^*}]_c + [\tilde{\eta}_{1|1^*}]_c} \tag{3.21}$$

$$y'_{n-1} = \frac{z_n[\tilde{x}'_{n|1^*}]_b + v_x t_1[\tilde{q}_{x|1^*}]_b + v_y t_1[\tilde{q}_{y|1^*}]_b + v_z t_1[\tilde{q}_{z|1^*}]_b + [\tilde{\eta}_{1|1^*}]_b}{z_n[\tilde{x}'_{n|1^*}]_c + v_x t_1[\tilde{q}_{x|1^*}]_c + v_y t_1[\tilde{q}_{y|1^*}]_c + v_z t_1[\tilde{q}_{z|1^*}]_c + [\tilde{\eta}_{1|1^*}]_c} \tag{3.22}$$

Similarly, for x'_{n-2} and y'_{n-2}:

$$x'_{n-2} = \frac{z_n[\tilde{x}'_{n|2^*}]_a + v_x t_2[\tilde{q}_{x|2^*}]_a + v_y t_2[\tilde{q}_{y|2^*}]_a + v_z t_2[\tilde{q}_{z|2^*}]_a + [\tilde{\eta}_{2|2^*}]_a}{z_n[\tilde{x}'_{n|2^*}]_c + v_x t_2[\tilde{q}_{x|2^*}]_c + v_y t_2[\tilde{q}_{y|2^*}]_c + v_z t_2[\tilde{q}_{z|2^*}]_c + [\tilde{\eta}_{2|2^*}]_c} \tag{3.23}$$

$$y'_{n-2} = \frac{z_n[\tilde{x}'_{n|2^*}]_b + v_x t_2[\tilde{q}_{x|2^*}]_b + v_y t_2[\tilde{q}_{y|2^*}]_b + v_z t_2[\tilde{q}_{z|2^*}]_b + [\tilde{\eta}_{2|2^*}]_b}{z_n[\tilde{x}'_{n|2^*}]_c + v_x t_2[\tilde{q}_{x|2^*}]_c + v_y t_2[\tilde{q}_{y|2^*}]_c + v_z t_2[\tilde{q}_{z|2^*}]_c + [\tilde{\eta}_{2|2^*}]_c} \tag{3.24}$$

After some primitive algebraic manipulations, these four equations become:

$$\begin{aligned} v_x\Big(t_1(x'_{n-1}[\tilde{q}_{x|1^*}]_c - [\tilde{q}_{x|1^*}]_a)\Big) + v_y\Big(t_1(x'_{n-1}[\tilde{q}_{y|1^*}]_c - [\tilde{q}_{y|1^*}]_a)\Big) + v_z\Big(t_1(x'_{n-1}[\tilde{q}_{z|1^*}]_c - [\tilde{q}_{z|1^*}]_a)\Big) + \\ z_n\Big(x'_{n-1}[\tilde{x}'_{n|1^*}]_c - [\tilde{x}'_{n|1^*}]_a\Big) = [\tilde{\eta}_{1|1^*}]_a - x'_{n-1}[\tilde{\eta}_{1|1^*}]_c \end{aligned} \tag{3.25}$$

$$\begin{aligned} v_x\Big(t_1(y'_{n-1}[\tilde{q}_{x|1^*}]_c - [\tilde{q}_{x|1^*}]_b)\Big) + v_y\Big(t_1(y'_{n-1}[\tilde{q}_{y|1^*}]_c - [\tilde{q}_{y|1^*}]_b)\Big) + v_z\Big(t_1(y'_{n-1}[\tilde{q}_{z|1^*}]_c - [\tilde{q}_{z|1^*}]_b)\Big) + \\ z_n\Big(y'_{n-1}[\tilde{x}'_{n|1^*}]_c - [\tilde{x}'_{n|1^*}]_b\Big) = [\tilde{\eta}_{1|1^*}]_b - y'_{n-1}[\tilde{\eta}_{1|1^*}]_c \end{aligned} \tag{3.26}$$

$$\begin{aligned} v_x\Big(t_2(x'_{n-2}[\tilde{q}_{x|2^*}]_c - [\tilde{q}_{x|2^*}]_a)\Big) + \Big(v_y t_2(x'_{n-2}[\tilde{q}_{y|2^*}]_c - [\tilde{q}_{y|2^*}]_a)\Big) + v_z\Big(t_2(x'_{n-2}[\tilde{q}_{z|2^*}]_c - [\tilde{q}_{z|2^*}]_a)\Big) + \\ z_n\Big(x'_{n-2}[\tilde{x}'_{n|2^*}]_c - [\tilde{x}'_{n|2^*}]_a\Big) = [\tilde{\eta}_{2|2^*}]_a - x'_{n-2}[\tilde{\eta}_{2|2^*}]_c \end{aligned} \tag{3.27}$$

$$\begin{aligned} v_x\Big(t_2(y'_{n-2}[\tilde{q}_{x|2^*}]_c - [\tilde{q}_{x|2^*}]_b)\Big) + v_y\Big(t_2(y'_{n-2}[\tilde{q}_{y|2^*}]_c - [\tilde{q}_{y|2^*}]_b)\Big) + v_z\Big(t_2(y'_{n-2}[\tilde{q}_{z|2^*}]_c - [\tilde{q}_{z|2^*}]_b)\Big) + \\ z_n\Big(y'_{n-2}[\tilde{x}'_{n|2^*}]_c - [\tilde{x}'_{n|2^*}]_b\Big) = [\tilde{\eta}_{2|2^*}]_b - y'_{n-2}[\tilde{\eta}_{2|2^*}]_c \end{aligned} \tag{3.28}$$

Let us define a matrix D, with elements d_{11} - d_{44} and a column matrix $\vec{f}$ with elements f_1 - f_4. We can rewrite (3.25)-(3.28) using the newly introduced variables as:

$$\text{Eq. (3.25): } v_x d_{11} + v_y d_{12} + v_z d_{13} + z_n d_{14} = f_1 \tag{3.29a}$$
$$\text{Eq. (3.26): } v_x d_{21} + v_y d_{22} + v_z d_{23} + z_n d_{24} = f_2 \tag{3.29b}$$
$$\text{Eq. (3.27): } v_x d_{13} + v_y d_{23} + v_z d_{33} + z_n d_{34} = f_3 \tag{3.29c}$$
$$\text{Eq. (3.28): } v_x d_{41} + v_y d_{42} + v_z d_{43} + z_n d_{44} = f_4 \tag{3.29d}$$

which results in:

$$D \begin{bmatrix} v_x \\ v_y \\ v_z \\ z_n \end{bmatrix} = f \tag{3.30}$$

Finally, we have derived the equation for the unknown velocities:

$$\vec{v}_e = \begin{bmatrix} v_x \\ v_y \\ v_z \\ z_n \end{bmatrix} = D^{-1} f \tag{3.31}$$

3.3.1 Uncertainty on Estimated Velocity

The velocity estimation takes 20 input arguments. The input vector can be written as:

$$\vec{o} = \begin{bmatrix} \vec{x}_n'^T & \vec{x}_{n-1}'^T & \vec{x}_{n-2}'^T & \vec{q}_1^T & \vec{q}_2^T & \vec{\alpha}_{n-1}^T & \vec{\alpha}_{n-1}^T \end{bmatrix}^T \tag{3.32}$$

Following the definition of $\vec{o}$, the covariance matrix O is defined as:

$$O = \begin{bmatrix} \Sigma_n & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Sigma_{n-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Sigma_{n-2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_{n-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A_{n-2} \end{bmatrix} \tag{3.33}$$

Q_1 and Q_2 matrices represent the covariance of the relative attitudes $\tilde{q}_1$ and $\tilde{q}_2$. If we define the velocity estimator function as $\vec{v}_e = l(\vec{o})$, $\vec{v}_e$ being the vector containing the velocities and z_n, the covariance matrix of $\vec{v}_e$ is computed as:

$$V_n = J_{l(\vec{o})}(\vec{o}) O J_{l(\vec{o})}(\vec{o})^T \tag{3.34}$$

Now, we should find derivatives of $l(\vec{o})$ with respect to the elements of $\vec{o}$.

For demonstration, let us find $\frac{\partial l(\vec{o})}{\partial x'_n}$. We use the equivalence[1] $\frac{\partial D^{-1}}{\partial x'_n} = D^{-1}\frac{\partial D}{\partial x'_n}D^{-1}$.

$$\frac{\partial l(\vec{o})}{\partial x'_n} = \frac{\partial D^{-1}\vec{f}}{\partial x'_n} \tag{3.35a}$$

$$= \frac{\partial D^{-1}}{\partial x'_n}\vec{f} + D^{-1}\frac{\partial \vec{f}}{\partial x'_n} \tag{3.35b}$$

$$= D^{-1}\frac{\partial D}{\partial x'_n}D^{-1}\vec{f} + D^{-1}\frac{\partial \vec{f}}{\partial x'_n} \tag{3.35c}$$

$$= D^{-1}\frac{\partial D}{\partial x'_n}\vec{v}_e + D^{-1}\frac{\partial \vec{f}}{\partial x'_n} \tag{3.35d}$$

This requires the computation of derivative of D and $\vec{f}$ for each parameter. After these derivatives are computed as shown in Appendix E, columns of the Jacobian matrix is found by using the formula in (3.35d). The constructed Jacobian matrix is plugged into 3.34 to find V_e. Upper left $3x3$ part of V_e corresponds to the covariance matrix of v_n, V_n.

3.4 Computation of Change in Attitude

The angular motion is sensed as instantaneous angular velocity by the gyroscope in three axes. The change in attitude between two time instants can then be computed simply by integrating the measured velocity.

Let the average gyroscope measurements in a time period of t_m be represented by $\vec{\omega} = [\omega_x\ \omega_y\ \omega_z]^T$ with $\omega = \|\vec{\omega}\|$. Let us represent the function that transforms $\vec{\omega}$ to its quaternion representation $\vec{q}$ by $\vec{q} = r(\vec{\omega}, t_m)$.

$$q = \begin{bmatrix} q_s \\ q_a \\ q_b \\ q_c \end{bmatrix} = r(\vec{\omega}, t_m) = \begin{bmatrix} \cos(\omega t_m/2) \\ \sin(\omega t_m/2)\omega_x t_m/\omega t_m \\ \sin(\omega t_m/2)\omega_y t_m/\omega t_m \\ \sin(\omega t_m/2)\omega_z t_m/\omega t_m \end{bmatrix} \tag{3.36}$$

3.4.1 Uncertainty on Change in Attitude

Since the measurements are represented in quaternions, the uncertainties on gyroscope readings should be represented in resultant quaternions.

In order to transform covariance matrix Ω of $\vec{\omega}$ to the covariance matrix Q of $\vec{q}$, Jacobian of the function $r(\vec{\omega}, t_m)$ is used. The computation of this Jacobian function is presented in Appendix B.

$$Q = J_{r(\vec{\omega},t_m)}(\vec{\omega})(\Omega\, t_m^2)J^T_{r(\vec{\omega},t_m)}(\vec{\omega}) \tag{3.37}$$

[1] One can easily derive this equivalence by evaluating the derivative of $D^{-1}D = I$.

3.5 Inertial Measurement Vector

We define the inertial measurement vector as:

$$\vec{\upsilon} = \begin{bmatrix} \vec{\tau}_\upsilon & \vec{q}_\upsilon \end{bmatrix}^T \tag{3.38}$$

$\vec{\tau}_\upsilon$ represents the translation. It is computed as:

$$\vec{\tau}_\upsilon = \vec{v}t - \frac{1}{2}\alpha t^2 \tag{3.39}$$

v is the estimated velocity and α is the accelerometer reading. Assuming their covariance is negligible, the covariance matrix of $\vec{\tau}_\upsilon$ is computed as:

$$T_\upsilon = Vt^2 + \frac{1}{4}At^4 \tag{3.40}$$

q is the rotation quaternion with covariance Q, as computed in Section 3.4. Assuming that the covariance between the rotation and the velocity is negligible, the covariance matrix of $\vec{\upsilon}$ becomes:

$$\Upsilon = \begin{bmatrix} T_\upsilon & 0 \\ 0 & Q \end{bmatrix} \tag{3.41}$$

3.6 Experimental Results

In this section, we analyze the performance of the inertial measurements between two visual frames. We compute the relative pose between two frames from the ground truth and compare the result with the relative pose estimated using the inertial readings.

Firstly we start with the translation component. As the reader can recall from Section 3.3, relative attitude between frames is needed for translational velocity estimation. The relative attitude can be computed via visual measurements or via gyroscope readings. Figure 3.9 shows the error distribution for two cases. Using the gyroscope readings results in a better performance. If the bias on gyroscope readings is discarded, the gyroscope has very low additional noise. This results in a smoother estimation of relative attitude in small time intervals. On the other hand, as analyzed in Section 2.4, there may be jumps in error on visual attitude measurements between consecutive frames. This results in poorly estimated velocity since the attitude change is very crucial in the formulation of the velocity estimation. This performance difference between inertial and visual relative attitude estimation results in the performance difference seen in Figure 3.9. As a result, gyroscopes should be preferred in the velocity estimation.

Figure (3.10) shows the error distributions for the three axes. The error on $z-axis$ is much higher than the other two axes. This is an expected result since we define $z-axis$ as the principal axis of the camera, and translation in the principal axis results in very small changes

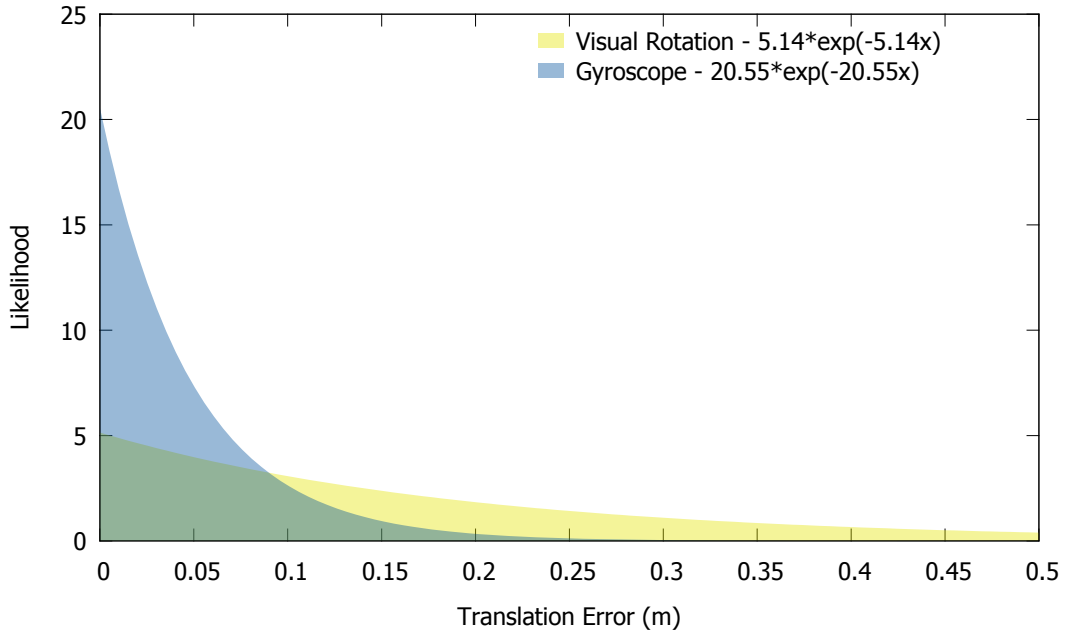


Figure 3.9: Distribution of the error on estimated translation when attitude change is computed from the visual measurements or from the gyroscope

in visual feature positions compared to other axes. Since a tracked feature is used in the velocity estimation, velocity estimation in the $z-axis$ can not receive aid from the visual tracking as much as other two axis does. Low aid from the visual part causes direct reflection of the accelerometer error to the estimated velocity in the $z-axis$.

Distribution of error on relative attitude estimation is shown in Figure 3.11. Since there is a linear relationship between gyroscope readings and estimated relative attitude, the error distribution can be represented by a Gaussian curve, similar to the noise on the gyroscope readings.

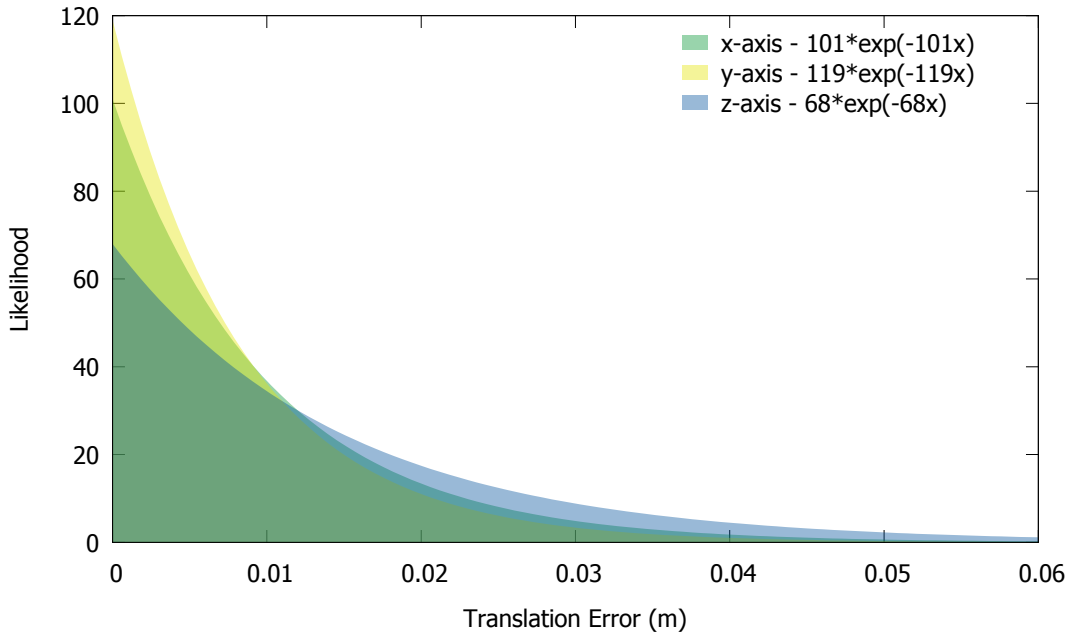


Figure 3.10: Distribution of the error on estimated translation on three axes

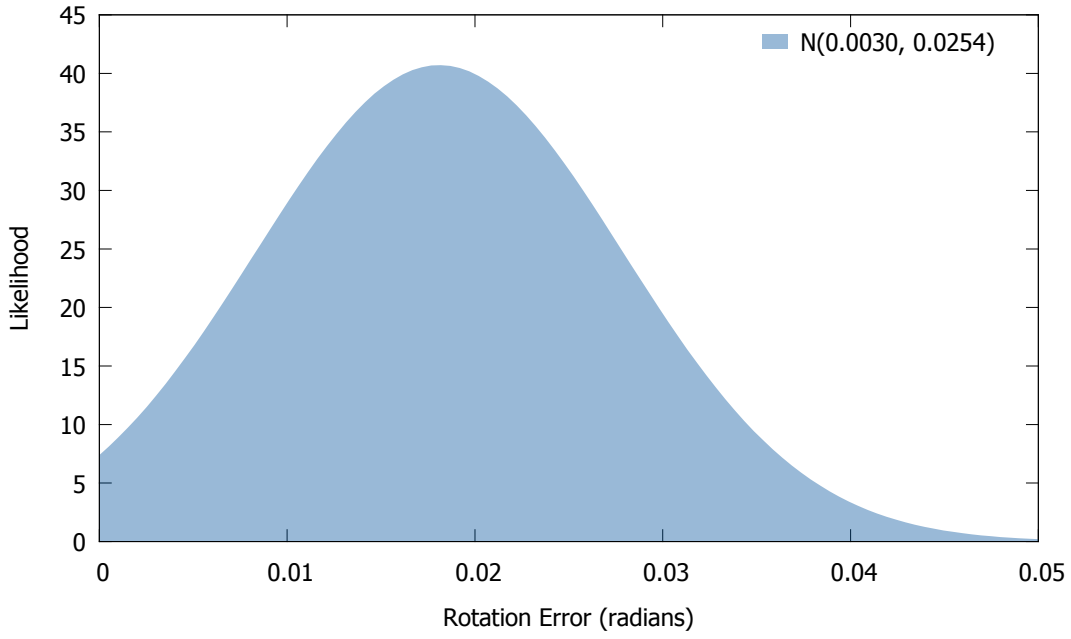


Figure 3.11: Distribution of the error on estimated rotation

CHAPTER 4

FUSING INERTIAL AND VISUAL INFORMATION FOR ODOMETRY

In order to fuse inertial and visual pose measurements, we propose an Extended Kalman Filter based system.

4.1 Proposed Kalman Filter

The state vector of the proposed Kalman filter, $\vec{\mu}$, shown in (4.1), consists of eight parameters: three for the metric position of the camera with respect to the world frame, $\vec{\tau}$, one for the scaling between inertial and visual measurements, λ, and four for orientation of the camera frame with respect to the world frame, $\vec{q}$.

$$\vec{\mu}_n = \begin{bmatrix} \vec{\tau}_n \\ \lambda \\ \vec{q}_n \end{bmatrix} ; \vec{\tau}_n = \begin{bmatrix} \tau_{x,n} \\ \tau_{y,n} \\ \tau_{z,n} \end{bmatrix} ; \vec{q}_n = \begin{bmatrix} s_n \\ a_n \\ b_n \\ c_n \end{bmatrix} ; \tag{4.1}$$

The constructed Kalman filter is presented in (4.2).

$$\hat{\vec{\mu}}_n = g(\vec{\mu}_{n-1}, \vec{\upsilon}_n) \tag{4.2a}$$
$$\hat{M}_n = J_g(\vec{\mu}_{n-1}) M_{n-1} J_g^T(\vec{\mu}_{n-1}) + J_g(\vec{\upsilon}_n) \Upsilon_n J_g^T(\vec{\upsilon}_n) \tag{4.2b}$$
$$\kappa_n = \hat{M}_n C^T (C \hat{M}_n C^T + \Phi_n)^{-1} \tag{4.2c}$$
$$\vec{\mu}_n = \hat{\vec{\mu}}_n + \kappa_n (\vec{\phi}_n - C \hat{\vec{\mu}}_n) \tag{4.2d}$$
$$M_n = (I_{8\times 8} - \kappa_n C) \hat{M}_n \tag{4.2e}$$

Here, $\vec{\mu}_n$ and M_n represent the state vector and its covariance, $\hat{\vec{\mu}}_n$ and $\hat{M}_n$ represent the predicted next state and its covariance, $\vec{\upsilon}_n$ and Υ_n represent the inertial measurement vector and its covariance, $g(\vec{\mu}_{n-1}, \vec{\upsilon}_n)$ represent prediction function, $\vec{\psi}_n$ and Ψ_n represent the visual measurement vector and its covariance, C represents the measurement model vector and κ_n represents the Kalman gain.

Computation of inertial measurements for the prediction stage ((4.2a) & (4.2b)) is detailed in Chapter 3. Computation of visual measurements for the correction stage ((4.2c), (4.2d) & (4.2e)) is detailed in Chapter 2.

4.1.1 The Prediction Stage

When EKF is utilized in odometry, a motion model is generally used for the prediction stage. The motion model represents the expected motion of the device given the previous state. Probably the most commonly used motion model is constant velocity motion model, in which the velocity is assumed to be same as the previous state update and the state transition is carried away based on that assumption. As a result, in the case of sudden changes in the motion behaviour, the predicted state vector $\hat{\vec{\mu}}$ may not appear as a good approximation to the state vector that is being estimated.

An alternative to constant velocity motion model is odometry motion model [85] in case there are additional odometry measurements such as a wheel encoder. If available, measurements from the odometry sensor can be fed to the system as if they are control inputs. This way, since the prediction is also conducted using different measurements, based on the characteristics of the additional odometry sensor, more reliable state predictions can be computed. As a part of this thesis, an odometry motion model-like motion model is utilized by feeding inertial measurements as control inputs to the constructed EKF.

The control input $\vec{\upsilon}$ is defined in Section 3.5. Prediction of the state vector of the next stage is shown in (4.2a). The nonlinear function $g(\vec{\mu}_{n-1}, \vec{\upsilon})$ is defined as:

$$\hat{\vec{\mu}}_n = g(\vec{\mu}_{n-1}, \vec{\upsilon}_n) \tag{4.3}$$

$$\hat{\tilde{\tau}}_n = \tilde{q}_{\upsilon,n}\tilde{\tau}_{n-1}\tilde{q}^*_{\upsilon,n} + \lambda_{n-1}\vec{\tau}_{\upsilon,n} \tag{4.4}$$

$$\hat{\lambda}_n = \lambda_{n-1} \tag{4.5}$$

$$\hat{\tilde{q}}_n = \tilde{q}_{\upsilon,n}\tilde{q}_{n-1} \tag{4.6}$$

Since $g(\vec{\mu}_{n-1}, \vec{\upsilon})$ is a nonlinear function, it should be linearized around the current state and the input. The Jacobians of the prediction function with respect to the previous state and the input vector is presented below.

$$J_g(\vec{m}u_{n-1}) = \begin{bmatrix} R_{\tilde{q}_\upsilon} & \vec{\tau}_\upsilon & 0_{3\times4} \\ 0_{1\times3} & 1 & 0_{1\times4} \\ 0_{4\times3} & 0_{4\times1} & J_{q_\upsilon q_{n-1}}(q_{n-1}) \end{bmatrix} \tag{4.7}$$

$$J_g(\vec{\upsilon}_n) = \begin{bmatrix} \lambda I_{3\times3} & J_{\tilde{q}_\upsilon \tilde{\tau}_{n-1}\tilde{q}^*_\upsilon}(\tilde{q}_\upsilon) \\ 0_{1\times3} & 0_{1\times4} \\ 0_{4\times3} & J_{q_\upsilon q_{n-1}}(q_\upsilon) \end{bmatrix} \tag{4.8}$$

The Jacobian matrices $J_{\tilde{q}_\upsilon \tilde{\tau}_{n-1}\tilde{q}^*_\upsilon}(\tilde{q}_\upsilon)$, $J_{q_\upsilon q_{n-1}}(q_{n-1})$ and $J_{q_\upsilon q_{n-1}}(q_\upsilon)$ and the rotation matrix $R_{\tilde{q}_\upsilon}$ are defined in Appendix B.

4.1.2 The Correction Stage

While the prediction stage is nonlinear, the correction stage is linear. This results in a partially extended Kalman filter.

The measurement vector and its covariance are computed as described in Chapter 2. The measurement vector consists of the position and the attitude represented with a quaternion. The C matrix that relates the state vector with the measurement vector is defined as:

$$\vec{\phi} = C\vec{\mu} = \begin{bmatrix} I_{3\times3} & 0_{3\times1} & 0_{3\times4} \\ 0_{4\times3} & 0_{4\times1} & I_{4\times4} \end{bmatrix} \vec{\mu} \tag{4.9}$$

The visual correction equation does not necessarily keep the magnitude of the rotation quaternion part of the state unity, which results in inconsistent rotation formulas. This problem is fixed by scaling the quaternion part of the state vector to magnitude one. Covariance matrix is also scaled accordingly.

4.1.3 System Initialization

The initial 3D map is generated as described in Section 2.2. The scale λ is initialized as described in Section 2.2.1.2. First pose and its covariance is set as the visual measurement taken just after the map creation.

4.2 Experimental Results

4.2.1 Metric Scale Estimation

As described in Section 2.2.1.2, the initial 3D point locations are set with an erroneous scale. In order to fuse inertial and visual measurements effectively, this scaling should be estimated. The scale λ is hence included in our state. In this section, our scale estimation performance is analyzed and compared to the scale estimation method proposed in [63].

In [63], Nutzi *et al.* proposes a Kalman filter with state vector consisting of the camera position, camera velocity, camera acceleration and scale. They update the state with each inertial and visual measurements separately. In the visual state update, camera position is observed. In the inertial state update, acceleration is observed. Unlike the proposed method, in the prediction state of their Kalman filter, covariance propagation is done heuristically since there is no actual measurement in their prediction stage.

Figure 4.1 shows the estimated scale throughout time at each visual frame arrival. This figure shows a typical result for scenarios with complex motion, consisting of rotations and small translations, typical during an augmented reality application in small workspaces. The algorithm presented in [63] fails rapidly. This is due to the accelerometer being highly erroneous in our device, unlike the accelerometer utilized originally in the mentioned paper. As there is a high bias on the utilized accelerometer, the velocity in the state vector of [63] diverges rapidly, resulting in a scale estimation of almost zero. The proposed algorithm, on the other

hand, converges to a higher scale than the true scale. This is due to the fact that the scale is observable only through the translation component of the estimated pose. When the translation is small, it becomes harder to observe the scale and our scale estimation converges to an erroneous value.

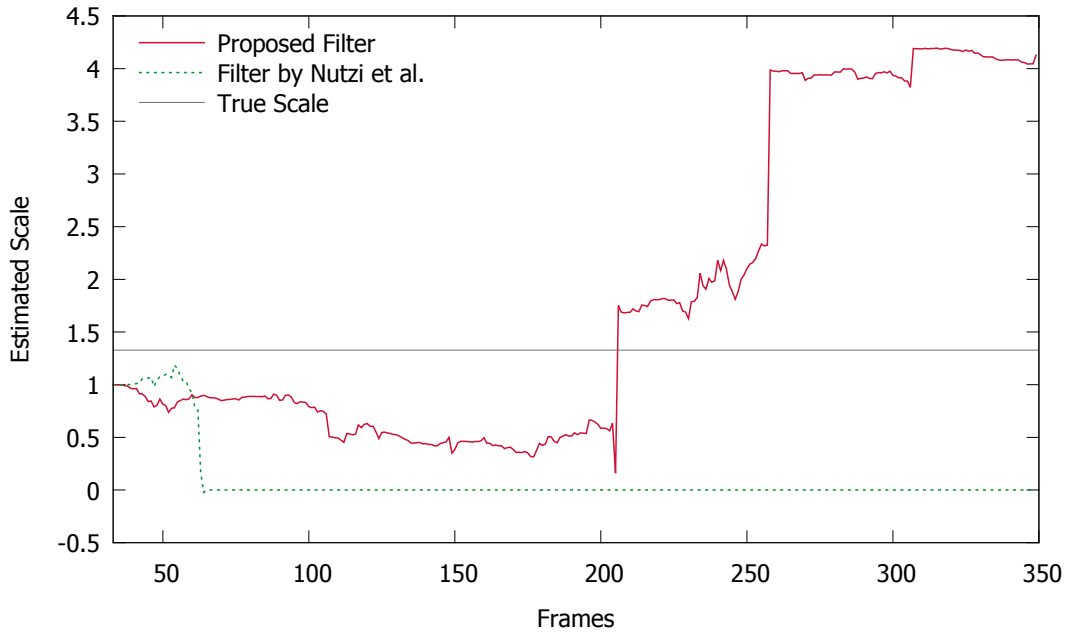


Figure 4.1: Estimated scale in a video with complex motion

We also ran our scale estimation algorithm in scenarios where the camera motion includes a continuous translation. In this case, the scale becomes much more observable and as seen in Figure 4.2 and our scale estimation converges to a value close to the true scale in under only 100 iterations.

As illustrated in Figure 4.3, the variance on the estimated scale decreases with time. The reason is that there is no factor increasing the uncertainty on the estimated scale since in the prediction stage, there is no uncertainty leaking into variance of the scale. The decreasing variance results in the scale estimation converging and eventually getting fixed. Since the scale is set once in the beginning of the system operation as a result of the erroneously measured translation between two initial poses used for map initialization, rather than tracking the scale throughout the system operation, we aim to determine its fixed value. This renders the decreasing variance of the estimated scale harmless if the scale estimation does not get fixed at an erroneous value. Hence, if there is dominantly translational motion during the first several seconds after the system initialization, the scale converges to an acceptable value.

4.2.2 Pose Estimation

We firstly begin by comparing the effect of high noise in $z-axis$ of the estimated velocity, as described in Section 3.6. Figure 4.4 compares the error on position component of the state vector when the velocity in $z-axis$ is taken into account and discarded. As the velocity is very noisy, when full velocity vector is used, the result has a highly varying performance. On the other hand, discarding the $z-axis$ results in a much smoother translation. As a result, for

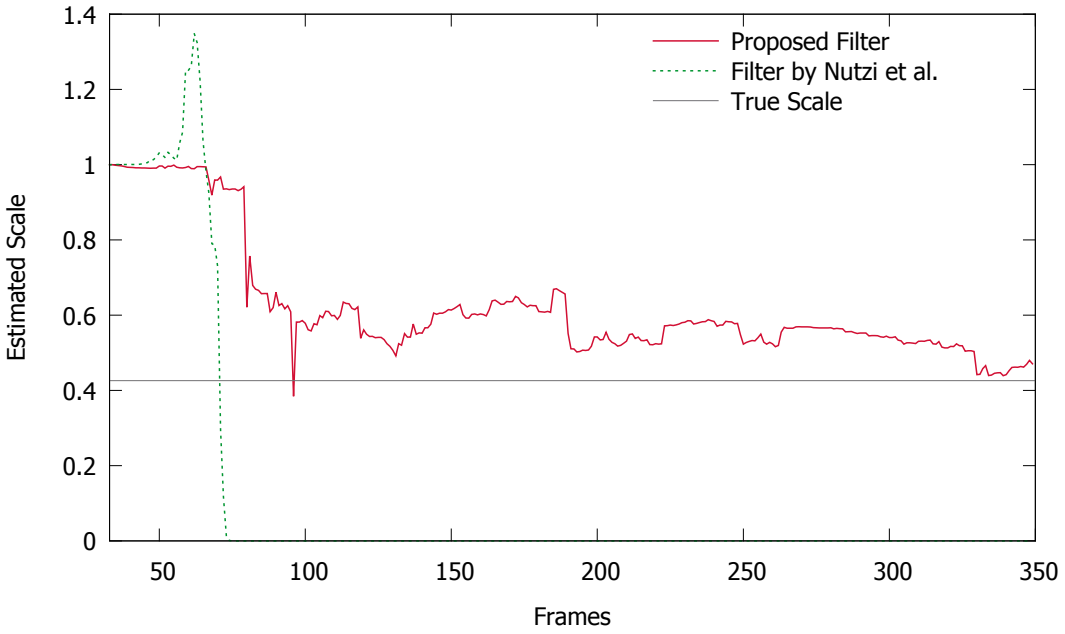


Figure 4.2: Estimated scale in a video with mostly translational motion

the rest of the tests, we will discard the velocity in $z-axis$.

One of the main advantages of filtering is that the result is smoother and robust to instantaneous erroneous measurements as it incorporates the current measurement with the previous pose. Figure 4.5 and Figure 4.6 show a typical example of this behaviour. One can see that the visual measurement gets corrupted for a short while towards the end of the video, but since it is not permanent, our state vector is not affected. This is a very important strength of the Kalman filter for its use in augmented reality applications because one of the biggest problems in current systems is the shaky appearance of the augmented objects.

The smoothing effect of the filter, on the other hand, can not hold the system for long time instances. Figure 4.7 and Figure 4.8 show the filter under highly erroneous visual input. The filter maintains its previous performance when the visual performance suddenly steps up but since the erroneous measurements keep coming persistently, the state transforms its state.

The filter and visual pose performance sometimes fluctuates, as shown in Figure 4.9 and Figure 4.10.

On a laptop computer, the times required for one visual pose estimation iteration for different map sizes are listed in Table 4.1. We can achieve up to 185 frames per second operation on a laptop computer when the image stream is at 30 frames per second. This is a very promising result for enabling real-time operation at 30 Hz on resource constrained COTS mobile devices.

When the required time for each iteration is compared to Table 2.1, one can notice that most of the time is consumed by the visual measurement computation.

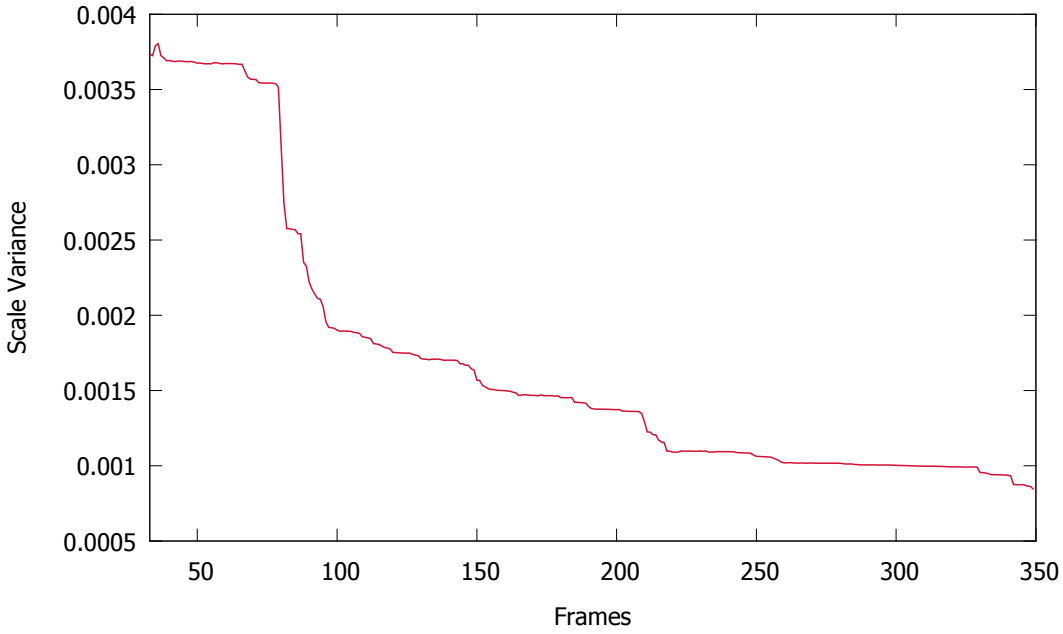


Figure 4.3: Variance of the estimated scale in a video with mostly translational motion

Table 4.1: Average time required for system propagation for different map sizes

# of tracked points	Time (ms)
20 in map, 30 total	20.97
16 in map, 24 total	14.37
12 in map, 18 total	11.40
8 in map, 12 total	8.42
6 in map, 9 total	5.38

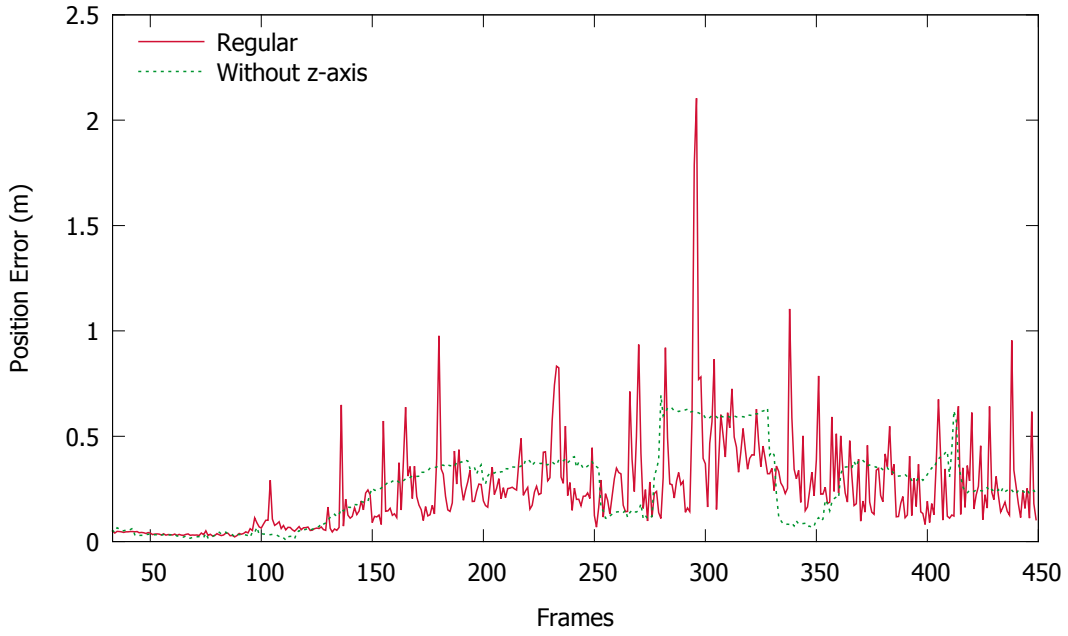


Figure 4.4: Position estimation performance of the Kalman filter with regular velocity estimation and velocity estimation with discarded $z-axis$

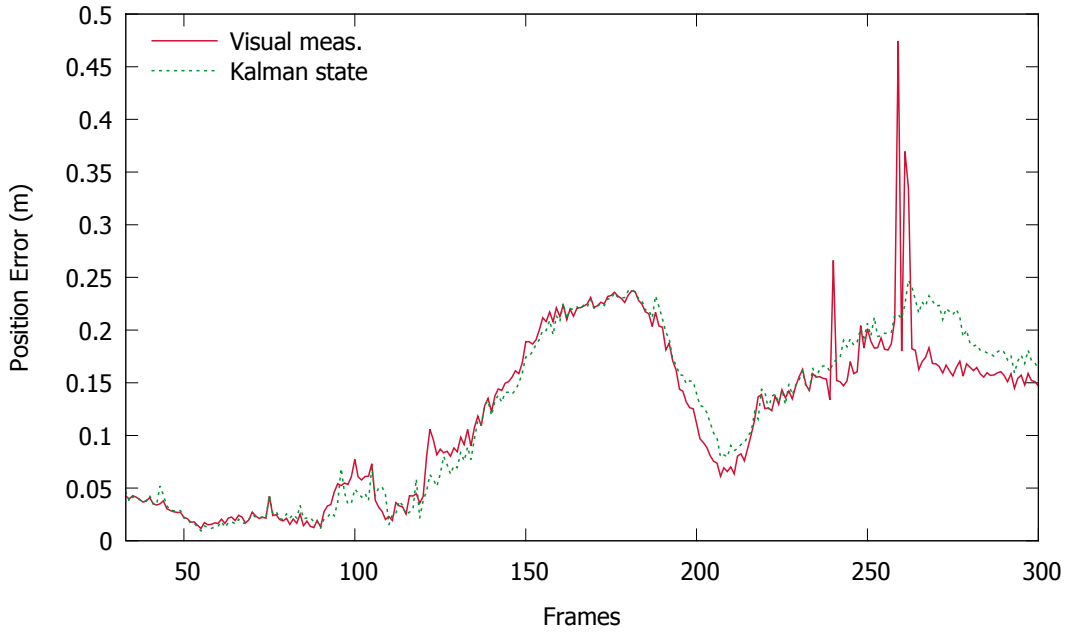


Figure 4.5: Position estimation performance of the Kalman filter with a map size of 16, compared to the corresponding visual pose estimation

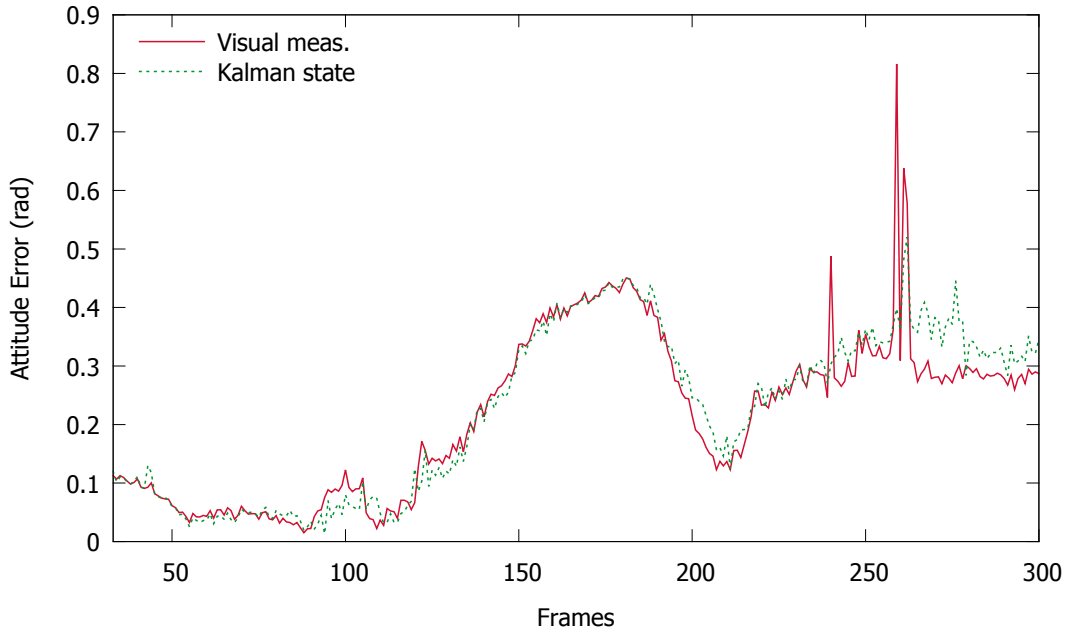


Figure 4.6: Attitude estimation performance of the Kalman filter with a map size of 16, compared to the corresponding visual pose estimation

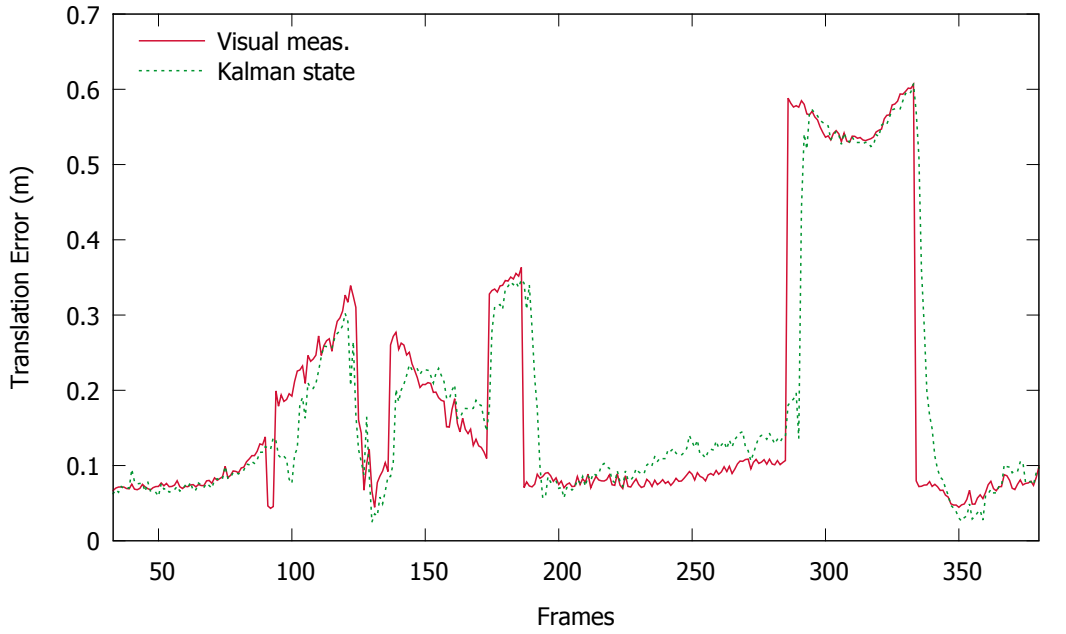


Figure 4.7: Position estimation performance of the Kalman filter with a map size of 8, compared to the corresponding visual pose estimation

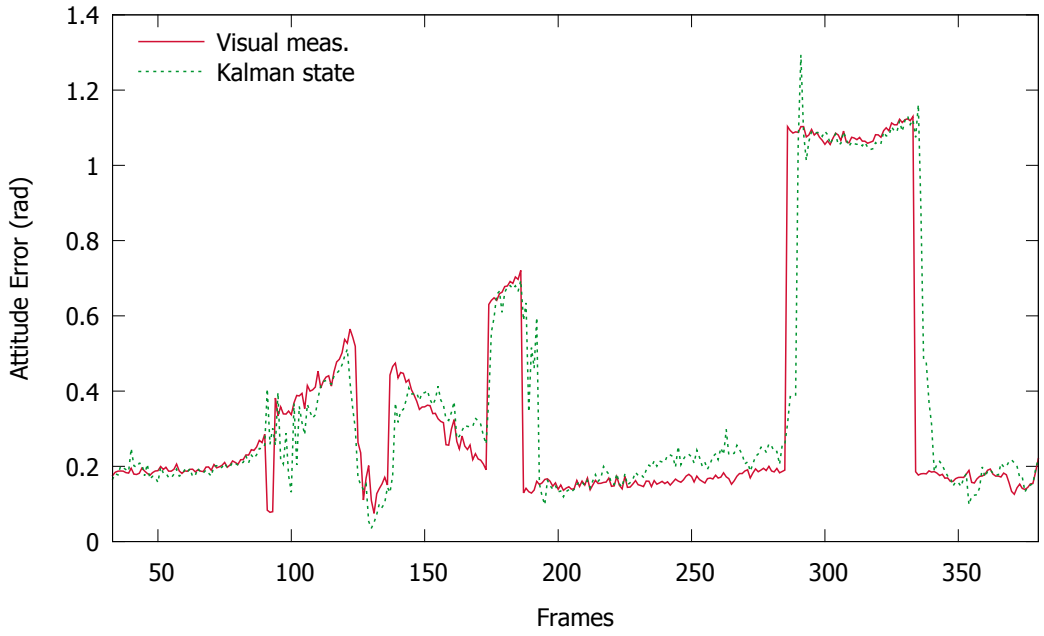


Figure 4.8: Attitude estimation performance of the Kalman filter with a map size of 8, compared to the corresponding visual pose estimation

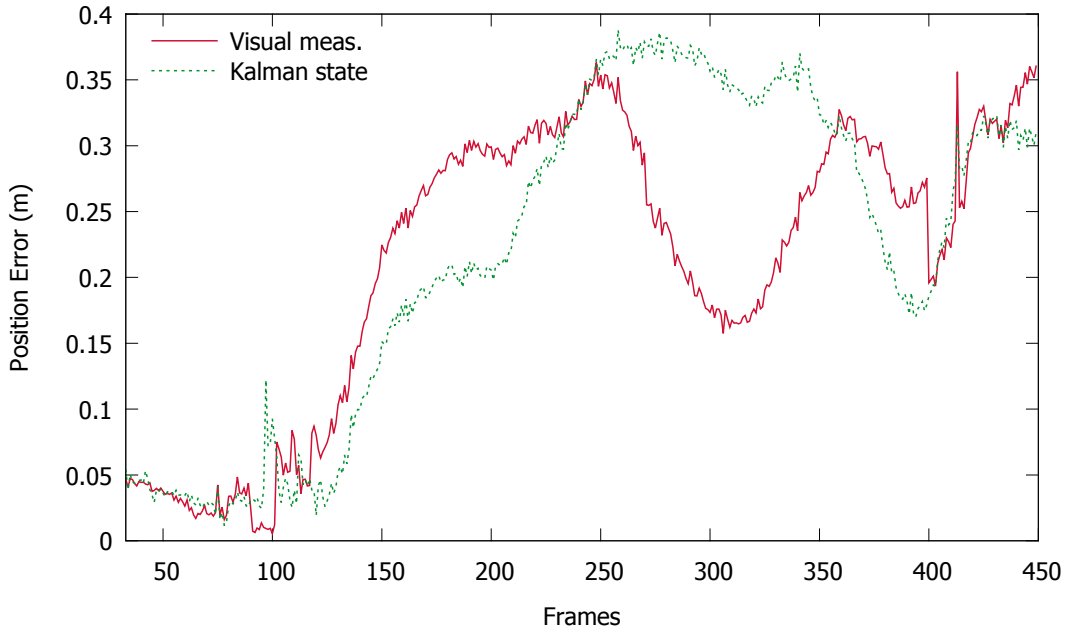


Figure 4.9: Position estimation performance of the Kalman filter with a map size of 20, compared to the corresponding visual pose estimation

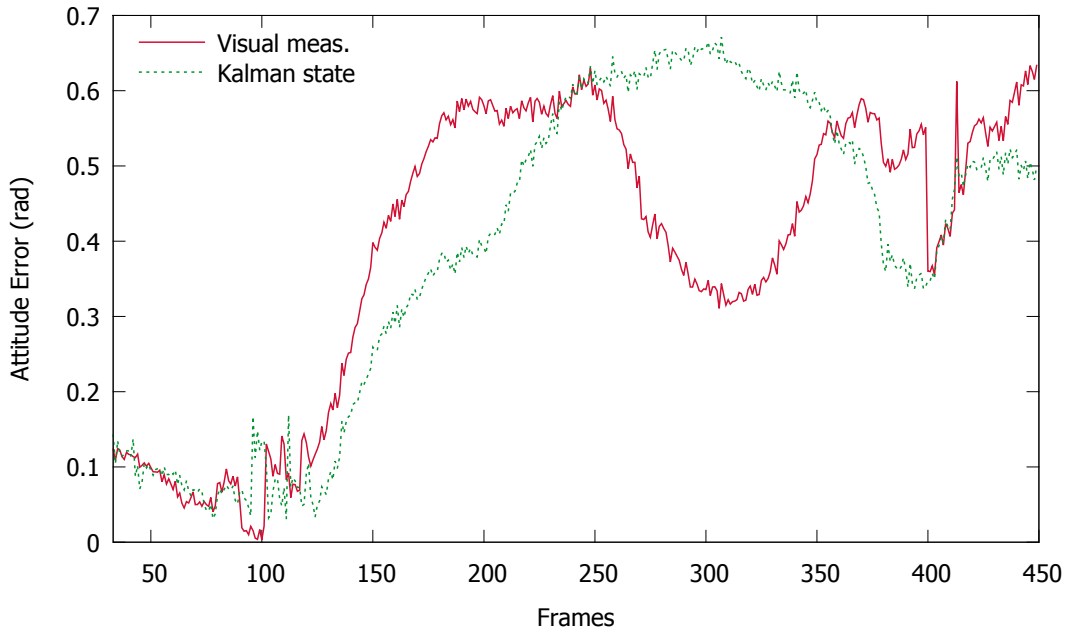


Figure 4.10: Attitude estimation performance of the Kalman filter with a map size of 20, compared to the corresponding visual pose estimation

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this thesis, several ways of cooperation between inertial and visual measurements are studied. Detailed computations of uncertainties on all the estimated variables are also provided.

The camera and inertial sensors of a COTS mobile device are used in the experiments. The utilized inertial sensors on the mobile device have quite highly-varying bias on both accelerometer and gyroscope; hence, it is difficult to include and successfully track them in the state of the Kalman filter. Therefore, the inertial readings are only utilized in short time intervals to predict relative pose between consecutive frames, making the effect of the bias negligible during system operation. In spite of their low SNR and low sample rate, the inertial sensors on the utilized mobile device are successfully used to aid the visual measurements.

The first way of cooperation included in this thesis is gyroscope-aided 2D feature tracking. Gyroscopes are used to generate predictions for the new positions of the tracked features. This technique is shown to reduce the distance between the initial and final positions of a feature in the KLT algorithm and hence increase the robustness of the tracking. Based on experiments, as shown in Figure 2.7, gyroscope aided 2D tracking initiates the iterative KLT tracking algorithm successfully, which yields less number of iterations. Other than the inertial aid, the uncertainty modelling in 2D tracking is analyzed. The employed uncertainty modeling of 2D feature positions makes the system independent of parameter tuning in terms of covariance computation. Although this is not confirmed by simulations, this is a trivial fact.

In any kind of 3D reconstruction or pose estimation scenario, generation of the initial 3D map or initial pose is crucial. In the proposed simple technique described in Section 2.2, it is shown that an initial estimation for a translation-only pose and its corresponding 3D structure could be obtained with an acceptable performance. The constraint on rotation-free camera motion that it is trapped to a single dimension during the map initialization is observed not to be crucial based on experimental results. The initial map generation inevitably creates a scaling error between the metric position and the visually estimated position of the camera due to the noisy accelerometer readings. In order to eliminate the effects of the initial scaling error, the scale is inserted into the state vector of the constructed Kalman filter. Based on the experimental results, the unknown scale converges to an acceptable value in case of a dominant translation in the camera motion. However, for a complex motion, although the proposed scale estimation performs better than the state of the art [63], the scale may converge to an incorrect value.

The experiments on the effect of map size on translation and rotation error indicate that decreasing the map size increases the variation of the error.

The proposed Kalman filter is fed with inertially measured translation and rotation in the prediction stage. In order to compute the relative translation using inertial measurements, the velocity of the camera should somehow be estimated. Based on [35], it can be shown that the translational velocity can be estimated without a double integration of the accelerometer readings of without tracking the velocity with the Kalman filter. Our experiments showed that the required rotation estimation for the velocity estimation formulation should be utilized from gyroscopes rather than the visual pose estimation. It should be noted that the $z-axis$ velocity estimation error is highly varying. The camera motion towards the principal axis of the camera does not yield a high motion of the tracked points on the image plane. This results in insufficient visual aid in the $z-axis$ velocity estimation, resulting in a highly-varying estimation error.

The computational load of the proposed system is shown to be very low. By sacrificing from robustness against possibly erroneous 3D locations of the points in the map and decreasing the number of features tracked, we can achieve up to 185 Hz operation on a laptop computer, which is quite promising for real-time operation at 30 Hz on a mobile device and is very impressive when compared to state-of-the-art.

Kalman filter is a powerful yet delicate tool. Covariance matrices of the initial state and measurement vectors affect the performance of the filter dramatically. One of the biggest advantages of the proposed system is that, unlike state of the art methods such as [46], covariance propagation is conducted without the use of any heuristics. Obtaining the estimate of the predicted covariance in a theoretically complete way is a big advantage of using inertial measurements in the prediction stage of the Kalman filter.

Performance of the visual pose estimation is damaged severely by erroneously initiated 3D locations of the tracked points. In this thesis, we focused on ways to improve visual pose estimation using the inertial readings. In order to increase the performance of visual pose estimation and hence the overall performance of the proposed Kalman filter, again a filter based approach to handle the 3D point locations and covariances can be developed. As real-time operation on resource constrained systems will not allow a filter that contains 3D point locations to run at the video frame rate of 30 Hz, construction of a PTAM-like system that runs two filters in parallel may be beneficial. One of the two filters shall be the proposed filter for pose estimation operating at the video rate and the other one a filter that keeps track of the sparse map at a lower frame rate.

REFERENCES

[1] S. L. Altmann. *Rotations, Quaternions and Double Groups*. Oxford University Press, 1986.

[2] C. Arth, A. Mulloni, and D. Schmalstieg. Exploiting sensors on mobile phones to improve wide-area localization. In *International Conference on Pattern Recognition (ICPR)*, pages 2152–2156, 2012.

[3] R. T. Azuma, B. R. Hoff, H. E. Neely, III, R. Sarfaty, M. J. Daily, G. Bishop, L. Vicci, G. Welch, U. Neumann, S. You, R. Nichols, and J. Cannon. Making augmented reality work outdoors requires hybrid tracking. In *International Workshop on Augmented Reality (IWAR)*, pages 219–224, 1998.

[4] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[5] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.

[6] J. Beich and M. Veth. Tightly-coupled image-aided inertial relative navigation using statistical predictive rendering (spr) techniques and a priori world models. In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 552–560, 2010.

[7] G. Bleser and D. Stricker. Advanced tracking through efficient image processing and visual-inertial sensor fusion. In *IEEE Virtual Reality Conference (VR)*, pages 137–144, 2008.

[8] J. Y. Bouguet. Pyramidal implementation of the Lucas-Kanade feature tracker. Technical report, Intel Corporation, Microprocessor Research Labs, 1999.

[9] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *IEEE International Conference on Computer Vision (ICCV)*, 2003.

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

[11] D. Diel, P. DeBitetto, and S. Teller. Epipolar constraints for vision-aided inertial navigation. In *IEEE Workshop on Motion and Video Computing (WACV/MOTION)*, volume 2, pages 221–228, 2005.

[12] P. S. R. Diniz. *Adaptive Filtering: Algorithms and Practical Implementation*. Springer, 3rd edition, 2008.

[13] T.-C. Dong-Si and A. Mourikis. Estimator initialization in vision-aided inertial navigation with unknown camera-IMU calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1064–1071, 2012.

[14] J. Durrie, T. Gerritsen, E. Frew, and S. Pledgie. Vision-aided inertial navigation on an uncertain map using a particle filter. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4189–4194, 2009.

[15] R. Faragher. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. *Signal Processing Magazine, IEEE*, 29(5):128–132, 2012.

[16] E. Foxlin, Y. Altshuler, L. Naimark, and M. Harrington. FlightTracker: a novel optical/inertial tracker for cockpit enhanced vision. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 212–221, 2004.

[17] E. Foxlin and L. Naimark. VIS-Tracker: a wearable vision-inertial self-tracker. In *IEEE Virtual Reality Proceedings (VR)*, pages 199–206, 2003.

[18] F. Fraundorfer and D. Scaramuzza. Visual odometry part ii: Matching, robustness, optimization, and applications. *IEEE Robotics Automation Magazine*, 19(2):78–90, 2012.

[19] P. Fua and V. Lepetit. Vision based 3D tracking and pose estimation for mixed reality. In M. Haller, M. Billinghurst, and B. H. Thomas, editors, *Emerging Technologies of Augmented Reality Interfaces and Design*, pages 43–63. Idea Group, Hershey, 2007.

[20] W. R. Hamilton. On quaternions, or on a new system of imaginaries in algebra. *Philosophical Magazine*, 25(3):489–495, 1844.

[21] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, pages 147–151, 1988.

[22] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003.

[23] W. Hoff and T. Vincent. Analysis of head pose accuracy in augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):319–334, 2000.

[24] J. D. Hol, T. B. Schön, and F. Gustafsson. Modeling and calibration of inertial and vision sensors. *The International Journal of Robotics Research*, 29(2-3):231–244, 2010.

[25] M. Hwangbo, J.-S. Kim, and T. Kanade. Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and GPU implementation. *The International Journal of Robotics Research*, 30(14):1755–1774, 2011.

[26] B. Jiang, U. Neumann, and S. You. A robust hybrid tracking system for outdoor augmented reality. In *IEEE Virtual Reality Conference (VR)*, pages 3–10, 2004.

[27] A. Kaheel, M. El-Saban, M. Izz, and M. Refaat. Employing 3D accelerometer information for fast and reliable image features matching on mobile devices. In *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 641–646, 2012.

[28] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[29] J. Kelly and G. Sukhatme. Fast relative pose calibration for visual and inertial sensors. In O. Khatib, V. Kumar, and G. Pappas, editors, *Experimental Robotics*, volume 54 of *Springer Tracts in Advanced Robotics*, pages 515–524. Springer Berlin Heidelberg, 2009.

[30] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56–79, 2011.

[31] G. Klein and T. Drummond. Robust visual tracking for non-instrumental augmented reality. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 113–122, 2003.

[32] G. Klein and T. Drummond. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10):769–776, 2004.

[33] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 225–234, 2007.

[34] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *IEEE International Symposium on Mixed and Augmented Reality*, 2009.

[35] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart. Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence. In *IEEE International Conference onRobotics and Automation (ICRA)*, pages 4546–4553, 2011.

[36] L. Kneip, S. Weiss, and R. Siegwart. Deterministic initialization of metric state estimation filters for loosely-coupled monocular vision-inertial systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[37] D. Kurz and S. Ben Himane. Inertial sensor-aligned visual feature descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 161–166, 2011.

[38] D. Kurz and S. Benhimane. Gravity-aware handheld augmented reality. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 111–120, 2011.

[39] P. Lang, A. Kusej, A. Pinz, and G. Brasseur. Inertial tracking for mobile augmented reality. In *IEEE Instrumentation and Measurement Technology Conference (IMTC)*, pages 1583–1587, 2002.

[40] S. Latta, K. Geisner, B. Mount, J. Steed, T. Ambrus, A. Zepeda, and A. Krauss. Multi-player gaming with head-mounted display. Patent Application. [No: A1 20130196757, Assignee: Microsoft Corporation].

[41] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[42] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *IEEE International Conference onComputer Vision (ICCV)*, pages 2548–2555, 2011.

[43] M. Li, B. Kim, and A. I. Mourikis. Real-time motion estimation on a cellphone using inertial sensing and a rolling-shutter camera. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4697–4704, 2013.

[44] M. Li and A. I. Mourikis. Improving the accuracy of ekf-based visual-inertial odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 828–835, 2012.

[45] M. Li and A. I. Mourikis. Optimization-based estimator design for vision-aided inertial navigation. In *Robotics: Science and Systems Conference*, 2012.

[46] M. Li and A. I. Mourikis. Vision-aided inertial navigation for resource-constrained systems. In *IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS)*, pages 1056–1063, 2012.

[47] M. Li and A. I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *International Journal of Robotics Research*, 32(6):690–711, May 2013.

[48] R. Liu, S. Z. Li, X. Yuan, and R. He. Online determination of track loss using template inverse matching. In *International Workshop on Visual Surveillance (VS)*, 2008.

[49] J. Lobo and J. Dias. Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1597–1608, 2003.

[50] J. Lobo and J. Dias. Relative pose calibration between visual and inertial sensors. *The International Journal of Robotics Research*, 26(6):561–575, 2007.

[51] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[52] C.-P. Lu, G. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2000.

[53] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, page 674–679, 1981.

[54] T. Lupton and S. Sukkarieh. Removing scale biases and ambiguity from 6DoF monocular SLAM using inertial. In *IEEE International Conference on Robotics and Automation(ICRA)*, pages 3698–3703, 2008.

[55] A. Martinelli. Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28(1):44–60, 2012.

[56] J. C. Maxwell. Remarks on the mathematical classification of physical quantities. *Proceedings of the London Mathematical Society*, pages 224–233, March 1871.

[57] J. E. Mebius. Derivation of the Euler-Rodrigues formula for three-dimensional rotations from the general formula for four-dimensional rotations. *arXiv:math/0701759*, January 2007.

[58] Metaio and S. Ericsson. Metaio brings world's first augmented reality accelerated chipset to market, signs agreement with ST-Ericsson to integrate future mobile platforms. [press release], Feb 21, 2013.

[59] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[60] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative O(n) solution to the PnP problem. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[61] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572, 2007.

[62] K. Nickels and S. Hutchinson. Estimating uncertainty in SSD-based feature tracking. *Image and Vision Computing*, 20(1):47–58, 2002.

[63] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart. Fusion of IMU and vision for absolute scale estimation in monocular SLAM. *Journal of Intelligent & Robotic Systems*, 61(1-4):287–299, 2011.

[64] J. O'Brian. How AR will change the way we search. in MASHABLE, `http://mashable.com/2013/03/12/visual-search/`, March 2013. [Online; accessed 13-August-2013].

[65] T. Oskiper, S. Samarasekera, and R. Kumar. Tightly-coupled robust vision aided inertial navigation algorithm for augmented reality using monocular camera and IMU. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 255–256, 2011.

[66] T. Oskiper, S. Samarasekera, and R. Kumar. Multi-sensor navigation algorithm using monocular camera, IMU and GPS for large scale augmented reality. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 71–80, 2012.

[67] G. Panahandeh, D. Zachariah, and M. Jansson. Exploiting ground plane constraints for visual-inertial navigation. In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 527–534, 2012.

[68] C. Pirchheim and G. Reitmayr. Homography-based planar mapping and tracking for mobile phones. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 27–36, 2011.

[69] R. Poelman, O. Akman, S. Lukosch, and P. Jonker. As if being there: mediated reality for crime scene investigation. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2012.

[70] A. Robinson. On the use of quaternions in simulation of rigid-body motion. Technical report, Wright Air Development Center, December 1958.

[71] L. Rollenhagen. Virtually furnish a room with IKEA's augmented reality catalog. in MASHABLE, `http://mashable.com/2012/07/19/ikea-augmented-reality-catalog/`, August 2013. [Online; accessed 13-August-2013].

[72] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1508–1511, October 2005.

[73] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443, May 2006.

[74] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010.

[75] S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[76] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[77] D. Scaramuzza and F. Fraundorfer. Visual odometry part i: The first 30 years and fundamentals. *IEEE Robotics Automation Magazine*, 18(4):80–92, 2011.

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

[79] H. Stark and J. W. Woods. *Probability and Random Processes with Applications to Signal Processing*. Prentice Hall, 3rd edition, 2002.

[80] P. Stoica and R. Moses. *Introduction to Spectral Analysis*. Prentice Hall, 1997.

[81] H. Strasdat, J. M. M. Montiel, and A. Davison. Real-time monocular slam: Why filter? In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[82] D. Strelow and S. Singh. Online motion estimation from image and inertial measurements. In *Workshop on Integration of Vision and Inertial Sensors (INERVIS)*, 2003.

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

[84] J.-P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz. A new approach to vision-aided inertial navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4161–4168, 2010.

[85] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press Cambridge, MA, 2005.

[86] D. Titterton and J. Weston. *Strapdown Inertial Navigation Technology*. American Institute of Aeronautics & Astronautics, 2004.

[87] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, and J. Montgomery. Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks. *Journal of Field Robotics*, 24(5):357–378, April 2007.

[88] C. Troiani and A. Martinelli. Vision-aided inertial navigation using virtual features. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4828–4834, 2012.

[89] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, July 2008.

[90] J. Vince. *Mathematics for Computer Graphics*. Springer, 2nd edition, 2006.

[91] C. Warren. Nokia's city lens hopes to make augmented reality mainstream. in MASHABLE, `http://mashable.com/2012/09/05/nokia-city-lens/`, September 2012. [Online; accessed 13-August-2013].

[92] P. D. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *Audio and Electroacoustics, IEEE Transactions on*, 15(2):70–73, 1967.

[93] T. Yap, M. Li, A. I. Mourikis, and C. Shelton. A particle filter for monocular vision-aided odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5663–5669, 2011.

[94] S. You and U. Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *IEEE Virtual Reality Conference (VR)*, pages 71–78, 2001.

[95] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *IEEE Virtual Reality Conference (VR)*, pages 260–267, 1999.

[96] S. You, U. Neumann, and R. Azuma. Orientation tracking for outdoor augmented reality registration. *IEEE Computer Graphics and Applications*, 19(6):36–42, 1999.

[97] T. Zarraonandia, I. Aedo, P. Díaz, and A. Montero. An augmented lecture feedback system to support learner and teacher communication. *British Journal of Educational Technology*, 44(4):616–628, 2013.

[98] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 666–673, 1999.

[99] F. Zhou, H.-L. Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 193–202, 2008.

APPENDIX A

KALMAN FILTER

Kalman filter, proposed by R.E. Kalman in 1960, is an estimation method that makes use of state representation and state transition to propagate a linear system obtaining optimal estimation error [28], and can be seen as an extension of the Wiener filtering theory [12].

It is used to combine two estimates of a quantity optimally in order to find an estimate with minimal uncertainty [86]. In this chapter, a brief introduction to and formulation of Kalman filter and extended Kalman filter is presented. For a more detailed analysis and mathematical derivation, readers are referred to [85]. For a simpler, intuitive explanation of Kalman filters, readers are referred to [15].

Kalman filter makes use of the state-space representation. This representation takes the output of the system as system states [12]. Because of their similarities, it can be considered as a state-space version of Recursive Least Squares (RLS) filtering [12].

Kalman filter propagates a stochastic state - the state is represented by a probability density function. This probability density function is parametrized as a Gaussian distribution and described by a mean vector and a covariance matrix.

Definitions and formulations in the rest of this chapter are mainly based on [85].

A.1 Kalman Filter

As discussed above, Kalman filter is used to combine two full or partial estimates of the state vector. Each iteration of a Kalman filter consists of two stages: state transition or prediction and measurement update or correction.

In the state transition stage, a hypothesis for the next state is generated. Previous state vector and its covariance are the input of this stage with optional user control input vector. Kalman filter requires a linear prediction function, such as (A.1).

$$\vec{\mu}_n = A_n\vec{\mu}_{n-1} + B_n\vec{v}_n + \vec{\varepsilon}_n \tag{A.1}$$

Here, $\vec{\mu}_{n-1}$ and $\vec{\mu}_n$ represents the previous and current state vector, $\vec{v}_n$ represents the user control input and $\vec{\varepsilon}_n$ is a zero-mean Gaussian random vector modeling the uncertainty introduced by state transition with covariance Υ. A_n and B_n are matrices that maps the previous state and the control input to the next state, respectively.

The measurement update stage combines the hypothesis generated in the state transition stage and combines it with an observation of the state, called measurement. It is also characterized by a linear function, shown in (A.2).

$$\vec{\phi}_n = C_n \vec{\mu}_n + \vec{\eta}_n \tag{A.2}$$

$\vec{\phi}_n$ represents the measurement and $\vec{\eta}_n$ is a zero-mean Gaussian random vector modeling the uncertainty in the measurement with covariance Φ. C_n is a matrix that maps the state to the measurement vector.

Since the defined functions are linear in their arguments and the arguments are Gaussian random vectors, estimated new state is also a Gaussian [79].

The prediction for the next state and its covariance is computed as:

$$\hat{\vec{\mu}}_n = A_n \vec{\mu}_{n-1} + B_n \vec{\upsilon}_n \tag{A.3a}$$
$$\hat{M}_n = A_n M_{n-1} A_n^T + \Upsilon_n \tag{A.3b}$$

M represents the covariance of the state vector and the variables with represents that they are prediction of the regular ones.

The power of Kalman filter lies in the *Kalman gain*. Kalman gain (κ_n) specifies how the prediction $\hat{\vec{\mu}}_n$ and the measurement $\vec{\phi}_n$ is combined to get the next state vector $\vec{\mu}$. It is computed as in (A.4a).

$$\kappa_n = \hat{M}_n C_n^T (C_n \hat{M}_n C_n^T + \Phi_n)^{-1} \tag{A.4a}$$
$$\vec{\mu}_n = \hat{\vec{\mu}} + \kappa_n (\vec{\phi} - C_t \hat{\vec{\mu}}_n) \tag{A.4b}$$
$$M_n = (I - \kappa_n C_n) \hat{M}_n \tag{A.4c}$$

(A.4b) and (A.4c) shows the propagation of predicted and measured mean vector and covariance to the new state estimate.

A.2 Extended Kalman Filter

The requirement of linear state propagation and measurement update functions in Kalman filter limit the applicability of the filter in many vision and robotics applications. To be able to utilize Kalman filter in nonlinear estimation problems, extended Kalman filter (EKF) is formulated.

In the case of nonlinear state transition and measurement update functions, estimated new state is no longer a Gaussian. EKF computes a Gaussian approximation to the resultant probability density function. Consider two nonlinear functions, g and h, representing state transition and measurement update functions respectively, as shown in (A.5). Computation of

the new mean of the state is straightforward and done by evaluating the functions at the input variables. In order to propagate the covariance matrices, functions are linearized using Taylor expansion as shown in (A.6).

$$\vec{\mu}_n = g(\vec{\mu}_{n-1}, \vec{v}_n) + \varepsilon_n \tag{A.5a}$$
$$\vec{\phi}_n = h(\vec{\mu}_n) + \eta_n \tag{A.5b}$$

$$g(\vec{\mu}_n, \vec{v}_n) \approx g(\vec{\mu}_{n-1}, \vec{v}_n) + J_{g(\vec{\mu}_{n-1}, \vec{v}_n)}(\vec{\mu}_{n-1})(\vec{\mu} - \vec{\mu}_{n-1}) \tag{A.6}$$

$J_{g(\vec{\mu}, \vec{v}_n)}(\vec{\mu})$ represents the Jacobian of the function $g(\vec{\mu}, \vec{v}_n)$ with respect to $\vec{\mu}$. With the same linearization applied to the function $h(\vec{\mu}_n)$, extended Kalman filter can be formulated as:

$$\hat{\vec{\mu}}_n = g(\vec{\mu}_{n-1}, \vec{v}_n) \tag{A.7a}$$
$$\hat{M}_n = J_{g(\vec{\mu}_{n-1}, \vec{v}_n)}(\vec{\mu}) M_{n-1} J_{g(\vec{\mu}, \vec{v}_n)}(\vec{\mu})^T + \Upsilon_n \tag{A.7b}$$
$$\kappa_n = \hat{M}_n J_{h(\vec{\mu})}(\vec{\mu})^T (J_{h(\vec{\mu})}(\vec{\mu}) \hat{M}_n J_{h(\vec{\mu})}(\vec{\mu}) + \Phi_n)^{-1} \tag{A.7c}$$
$$\vec{\mu}_n = \hat{\vec{\mu}} + \kappa_n (\vec{\phi} - h(\vec{\mu}_n) \tag{A.7d}$$
$$M_n = (I - \kappa_n J_{h(\vec{\mu})}(\vec{\mu})) \hat{M}_n \tag{A.7e}$$

Due to the matrix inversion in the computation of the Kalman gain, the complexity of EKF is $O(n^2)$, n being the state dimension [85]. Hence, especially for resource limited environments, the state vector should be kept as small as possible for real-time operation.

APPENDIX B

QUATERNIONS FOR REPRESENTING ROTATIONS

Quaternions, considered as "the most brilliant formulation of the four-parameter method" by Robinson [70], is an extension to the complex number system where a quaternion is characterized by one real and three imaginary parameters firstly developed by W.R. Hamilton [20]. They constitute an important role in mathematics, computer vision and computer graphics and their invention was once described by James Clerk Maxwell as "a step towards the knowledge of quantities related to space which can only be compared, for its importance, with the invention of triple coordinates by Descartes" [56].

A quaternion is defined as

$$\tilde{q} = q_s + q_a i + q_b j + q_c k$$

where q_s, q_a, q_b and q_c are real numbers and i, j and k are characterized as

$$\begin{aligned} i^2 &= -1, \quad ij = -ji = k \\ j^2 &= -1, \quad jk = -jk = i \\ k^2 &= -1, \quad ki = -ik = j \end{aligned}$$

For ease of notation, we use the following representations interchangeably:

$$q_s + q_a i + q_b j + q_c k = \tilde{q} \Leftrightarrow \vec{q} = \begin{bmatrix} q_s \\ q_a \\ q_b \\ q_c \end{bmatrix} \tag{B.2}$$

Given two quaternions $\tilde{q}_1 = q_{1,s} + q_{1,a} i + q_{1,b} j + q_{1,c} k$ and $\tilde{q}_2 = q_{2,s} + q_{2,a} i + q_{2,b} j + q_{2,c} k$, addition and subtraction of two quaternions are straightforward.

$$\begin{aligned} \tilde{q}_1 + \tilde{q}_2 &= (q_{1,s} + q_{2,s}) + (q_{1,a} + q_{2,a}) i + (q_{1,b} + q_{2,b}) j + (q_{1,c} + q_{2,c}) k \\ \tilde{q}_1 - \tilde{q}_2 &= (q_{1,s} - q_{2,s}) + (q_{1,a} - q_{2,a}) i + (q_{1,b} - q_{2,b}) j + (q_{1,c} - q_{2,c}) k \end{aligned} \tag{B.3a}$$

Multiplication of two quaternions is defined as their *Hamiltonian* product. Making use of the multiplication rules for three imaginary components, the multiplication is defined as:

$$\tilde{q}_3 = \tilde{q}_1\tilde{q}_2 \quad \Rightarrow \quad \begin{array}{l} s_3 = q_{1,s}q_{2,s} - q_{1,a}q_{2,a} - q_{1,b}q_{2,b} - q_{1,c}q_{2,c} \\ a_3 = q_{1,s}q_{2,a} + q_{1,a}q_{2,s} + q_{1,b}q_{2,c} - q_{1,c}q_{2,b} \\ b_3 = q_{1,s}q_{2,b} - q_{1,a}q_{2,c} + q_{1,b}q_{2,s} + q_{1,c}q_{2,a} \\ c_3 = q_{1,s}q_{2,c} + q_{1,a}q_{2,b} - q_{1,b}q_{2,a} + q_{1,c}q_{2,s} \end{array} \tag{B.4}$$

Norm of a quaternion is defined as:

$$\|\tilde{q}\| = \sqrt{q_s^2 + q_a^2 + q_b^2 + q_c^2} \tag{B.5}$$

Inverse of a quaternion is then defined as:

$$\tilde{q} * \tilde{q}^{-1} = 1 \quad \Rightarrow \quad \tilde{q}^{-1} = \frac{q_s - q_a i - q_b j - q_c k}{\|q\|} \tag{B.6}$$

Two special cases of quaternions that are of interest in scope of this thesis are *unit quaternion*, defined as a quaternion with unit norm, and *pure quaternion*, defined as a quaternion with zero real part [1].

Notice that the inverse of a unit quaternion is equal to the complex conjugate of that quaternion, $\tilde{q}^{-1} = \tilde{q}^*$.

Unit quaternions can be used to represent rotations around an arbitrary axis. Let us have a point 3D point p with coordinates p_x, p_y and p_z. When we define an axis of rotation as a unit pure quaternion $\tilde{u} = x_u i + y_u j + z_u k$ and magnitude of rotation in radians as θ, the corresponding quaternion representing that rotation is defined as $\tilde{q} = \cos(\theta/2) + \sin(\theta/2)u$ [90, 1]. Then, the rotation can be formulated according to Hamilton-Cayley formula [57] as:

$$\tilde{P}' = \tilde{q}\tilde{P}\tilde{q}^{-1} \tag{B.7}$$

where $\tilde{P}$ is defined as the pure quaternion representing the point p, $\tilde{P} = p_x i + p_y j + p_z k$. $\tilde{P}'$ is the pure quaternion representing the rotated point, p'.

B.1 Transformation between Rotation Quaternions and Rotation Matrices

A rotation in three dimensions can be represented by an orthogonal 3×3 matrix R with unit determinant. A rotation quaternion can be transformed to a rotation matrix by representing the Hamilton-Cayley formula with a matrix. The resultant formula is called the Euler-Rodrigues formula [57].

$$\begin{array}{c} \tilde{P}' = \tilde{q}\tilde{P}\tilde{q}^{-1} \\ p' = Rp \end{array} \quad \Rightarrow \quad R = \begin{bmatrix} q_s^2 + q_a^2 - q_b^2 - q_c^2 & -2q_sq_c + 2q_aq_b & 2q_sq_b + 2q_aq_c \\ 2q_sq_c + 2q_aq_b & q_s^2 - q_a^2 + q_b^2 - q_c^2 & -2q_sq_a + 2q_bq_c \\ -2q_sq_b + 2q_aq_c & 2q_sq_a + 2q_bq_c & q_s^2 - q_a^2 - q_b^2 + q_c^2 \end{bmatrix} \tag{B.8}$$

For the inverse transformation, one can obtain (B.9) and (B.10) from (B.8) [57].

$$q_s^2 = (1 + r_{11} + r_{22} + r_{33})/4 \tag{B.9a}$$
$$q_a^2 = (1 + r_{11} - r_{22} - r_{33})/4 \tag{B.9b}$$
$$q_b^2 = (1 - r_{11} + r_{22} - r_{33})/4 \tag{B.9c}$$
$$q_c^2 = (1 - r_{11} - r_{22} + r_{33})/4 \tag{B.9d}$$

$$q_s q_a = (r_{32} - r_{23})/4 \tag{B.10a}$$
$$q_s q_b = (r_{13} - r_{31})/4 \tag{B.10b}$$
$$q_s q_c = (r_{21} - r_{12})/4 \tag{B.10c}$$
$$q_a q_c = (r_{13} + r_{31})/4 \tag{B.10d}$$
$$q_a q_b = (r_{21} + r_{12})/4 \tag{B.10e}$$
$$q_b q_c = (r_{32} + r_{23})/4 \tag{B.10f}$$

Notice that signs of the quaternion parameters is ambiguous. One can set a sign for one of the four parameters using one of the equations in (B.9) and set the others using (B.10). In this thesis, for numerical stability, firstly, all parameters are computed using (B.9). Then, the largest one selected and the others are recomputed using (B.10).

B.2 Angular Displacement Represented in Quaternions

Relative attitude between two time instances can be represented or measured - by a gyroscope, for instance - as angular displacements in three axes separately.

Let the change in attitude be represented by:

$$\vec{\theta} = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \tag{B.11}$$

where θ_x, θ_y and θ_z represent change in attitude for three axes in radians. The small angle between two attitudes is found as:

$$\theta = \|\vec{\theta}\| \tag{B.12}$$

The corresponding quaternion is computed as [86]:

$$\vec{q} = \begin{bmatrix} q_s \\ q_a \\ q_b \\ q_c \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\theta_x/\theta \\ \sin(\theta/2)\theta_y/\theta \\ \sin(\theta/2)\theta_z/\theta \end{bmatrix} \tag{B.13}$$

B.3 Related Jacobians

Jacobian of the $\tilde{q}_1\tilde{q}_2$ with respect to $\tilde{q}_1$ is computed as:

$$J_{\tilde{q}_1\tilde{q}_2}(\tilde{q}_1) = \begin{bmatrix} q_{2,s} & -q_{2,a} & -q_{2,b} & -q_{2,c} \\ q_{2,a} & q_{2,s} & q_{2,c} & -q_{2,b} \\ q_{2,b} & -q_{2,c} & q_{2,s} & q_{2,a} \\ q_{2,c} & q_{2,b} & -q_{2,a} & q_{2,s} \end{bmatrix} \tag{B.14}$$

Similarly, Jacobian of the $\tilde{q}_1\tilde{q}_2$ with respect to $\tilde{q}_2$ is computed as:

$$J_{\tilde{q}_1\tilde{q}_2}(\tilde{q}_1) = \begin{bmatrix} q_{1,s} & -q_{1,a} & -q_{1,b} & -q_{1,c} \\ q_{1,a} & q_{1,s} & -q_{1,c} & q_{1,b} \\ q_{1,b} & q_{1,c} & q_{1,s} & -q_{1,a} \\ q_{1,c} & -q_{1,b} & q_{1,a} & q_{1,s} \end{bmatrix} \tag{B.15}$$

If we want to take the Jacobian of the rotation $\tilde{P}' = \tilde{q}\tilde{P}\tilde{q}^{-1}$ with respect to $\tilde{q}$, we have to define $\dfrac{\partial P'}{\partial q_\xi}$ for $\xi = s,a,b,c$ individually:

$$\frac{\partial P'}{\partial q_s} = \begin{bmatrix} 2q_s & -2q_c & 2q_b \\ 2q_c & 2q_s & -2q_a \\ -2q_b & 2q_a & 2q_s \end{bmatrix} P = R_{s,\tilde{q}}P \tag{B.16}$$

$$\frac{\partial P'}{\partial q_a} = \begin{bmatrix} 2q_a & 2q_b & 2q_c \\ 2q_b & -2q_a & -2q_s \\ 2q_c & 2q_s & -2q_a \end{bmatrix} P = R_{a,\tilde{q}}P \tag{B.17}$$

$$\frac{\partial P'}{\partial q_b} = \begin{bmatrix} -2q_b & 2q_a & 2q_s \\ 2q_a & 2q_b & 2q_c \\ -2q_s & 2q_c & -2q_b \end{bmatrix} P = R_{b,\tilde{q}}P \tag{B.18}$$

$$\frac{\partial P'}{\partial q_c} = \begin{bmatrix} -2q_c & -2q_s & 2q_a \\ 2q_s & -2q_c & 2q_b \\ 2q_a & 2q_b & 2q_c \end{bmatrix} P = R_{c,\tilde{q}}P \tag{B.19}$$

Then the Jacobian of the rotation becomes:

$$J_{\tilde{q}P\tilde{q}}(\tilde{q}) = \begin{bmatrix} R_{s,\tilde{q}}P & R_{a,\tilde{q}}P & R_{b,\tilde{q}}P & R_{c,\tilde{q}}P \end{bmatrix} \tag{B.20}$$

The Jacobian of the change-in-attitude to quaternion transformation with respect to change-in-attitude is:

$$J_{\vec{\theta}\to\vec{q}}(\vec{\theta}) = \begin{bmatrix} -\frac{\theta_x \sin(\theta/2)}{2\theta} & -\frac{\theta_y \sin(\theta/2)}{2\theta} & -\frac{\theta_z \sin(\theta/2)}{2\theta} \\ \frac{2(\theta_y^2+\theta_z^2)\sin(\theta/2)+\theta_x^2\theta\cos(\theta/2)}{2\theta^3} & \frac{1}{2}\theta_x\theta_y\frac{\theta\cos(\theta/2)-2\sin(\theta/2)}{\theta^3} & \frac{1}{2}\theta_x\theta_z\frac{\theta\cos(\theta/2)-2\sin(\theta/2)}{\theta^3} \\ \frac{1}{2}\theta_x\theta_y\frac{\theta\cos(\theta/2)-2\sin(\theta/2)}{\theta^3} & \frac{2(\theta_x^2+\theta_z^2)\sin(\theta/2)+\theta_y^2\theta\cos(\theta/2)}{2\theta^3} & \frac{1}{2}\theta_y\theta_z\frac{\theta\cos(\theta/2)-2\sin(\theta/2)}{\theta^3} \\ \frac{1}{2}\theta_x\theta_z\frac{\theta\cos(\theta/2)-2\sin(\theta/2)}{\theta^3} & \frac{1}{2}\theta_y\theta_z\frac{\theta\cos(\theta/2)-2\sin(\theta/2)}{\theta^3} & \frac{2(\theta_x^2+\theta_y^2)\sin(\theta/2)+\theta_z^2\theta\cos(\theta/2)}{2\theta^3} \end{bmatrix} \quad \text{(B.21)}$$

B.4 Advantages of Using Quaternions for Representing Rotations

Other than quaternions, one of the most common ways to represent rotations is Euler angles [90]. There are three Euler angles:

- Pitch: Rotation angle about the x-axis.
- Yaw: Rotation angle about the y-axis.
- Roll: Rotation angle about the z-axis.

Because visualization of rotation in Euler angles is very convenient, this representation is widely used. There are two main advantages of quaternions over Euler angles, though, and these advantages lured us into utilizing quaternions in our system rather than Euler angles.

Firstly, it is shown in [83], a 3-dimensional representation of rotation can not be global and non-singular simultaneously. Being non-singular, Euler angle representation suffers from being unable to represent certain rotations in special conditions. This situation is called *gimble lock* and is one of the biggest shortcomings of this representation [90]. Quaternions, on the other hand, with one additional parameter, proves to be both global and non-singular [83].

The other main advantage of quaternions for representing rotations is their numerical accuracy [70]. The detailed analysis conducted in [70] concludes that, in case of unrestricted rotations about an arbitrary axis, numerical accuracy of quaternions is considerably higher.

APPENDIX C

MULTIVIEW GEOMETRY BASICS

C.1 Pinhole Camera Model

To transform the points in 3D space to the image coordinates, we employ the pinhole camera model. In this appendix, basic definitions of the entities used throughout the thesis is provided.

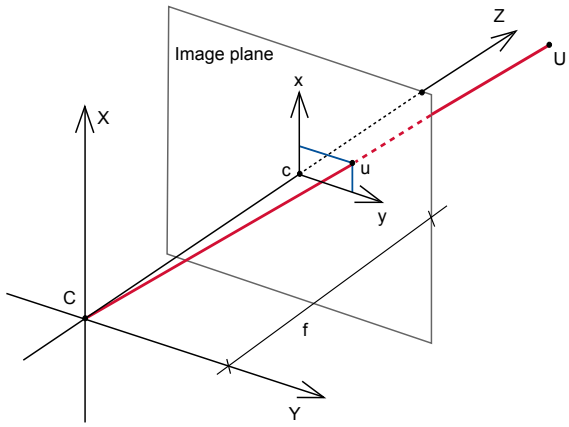


Figure C.1: Pinhole camera model. Here, U is the point in 3D, u is the projection of the 3D point on the image plane, f is the focal distance, C is the camera center and c is the principal point.

Pinhole camera model models the camera as a point in space, and the location of the projection of a 3D point on the image plane is determined by a ray passing through the 3D point and the camera center as seen in Figure C.1 [22].

The distance from the camera center to the image plane is called the *focal distance* and represented by f. Although it is not shown in Figure C.1, the focal distance for the image coordinate axes x and y can be different. If this is the case, corresponding focal distances are denoted by f_x and f_y.

The point where the line, which is perpendicular to the image plane and also passes through the camera center, intersects with the image plane is called the *principal point* and represented by c. The location of c on the image plane is $\vec{c} = \begin{bmatrix} c_x & c_y \end{bmatrix}$.

When the 3D point U is represented on camera coordinates, u is computed as:

$$z\begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_X \\ U_Y \\ U_Z \end{bmatrix} = K \begin{bmatrix} U_X \\ U_Y \\ U_Z \end{bmatrix} \tag{C.1}$$

z represents a scaling constant that transforms u from homogeneous coordinates to the actual image coordinates. Here, K is called the *camera calibration matrix*. There are techniques and tools in the literature to determine the calibration matrix of a camera, such as [98].

Points in 3D space are usually represented in the world coordinate frame, which is usually different from the camera coordinate frame. In this case, in order to project the 3D point onto the image plane, its coordinates should be transformed to the camera coordinates. If we denote the location of the camera center in the world coordinates as $\vec{C}$ and the rotation from the world frame to the camera frame with 3×3 rotation matrix R, the point U^W in the world coordinates can be represented in camera coordinates as:

$$\begin{bmatrix} U_X \\ U_Y \\ U_Z \end{bmatrix} = R(\vec{U}^W - \vec{C}) = \begin{bmatrix} R & -R\vec{C} \end{bmatrix} \begin{bmatrix} U_X^W \\ U_Y^W \\ U_Z^W \\ 1 \end{bmatrix} \tag{C.2}$$

We can denote $-RC$ with $\vec{t}$ representing the translation from the camera center to the world center in the camera coordinate frame. If we plug in the previous equation to the projection equation, we get:

$$z\begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = K \begin{bmatrix} R & \vec{t} \end{bmatrix} \begin{bmatrix} U_X^W \\ U_Y^W \\ U_Z^W \\ 1 \end{bmatrix} = P \begin{bmatrix} U_X^W \\ U_Y^W \\ U_Z^W \\ 1 \end{bmatrix} \tag{C.3}$$

P, which contains all the information to project a point in the world coordinates to the image plane, is called the *projection matrix*.

C.2 Triangulation

Triangulation is a process to determine the 3D location of a point from its observations in at least two images taken with different camera poses.

Let us denote the 3D point with unknown location by $\vec{U}$ and its two observations by $\vec{u}_1$ and $\vec{u}_2$ such that $\vec{u}_1 = P_1\vec{U}$ and $\vec{u}_2 = P_2\vec{U}$. Using these equalities, we can get:

$$\vec{u}_1 \times P_1\vec{U} = 0 \tag{C.4a}$$

$$\vec{u}_2 \times P_2\vec{U} = 0 \tag{C.4b}$$

These two equations give us four constraints for four homogeneous coordinates of the 3D point [22]. These equations can be solved using the homogeneous Direct Linear Transformation (DLT) method or by inhomogeneous method as detailed in [22].

APPENDIX D

COMPUTATION OF JACOBIAN OF 3D-TO-2D PROJECTION FUNCTION

The function that maps 3D point coordinates X_i, Y_i and Z_i to the image coordinates x_i and y_i is defined as:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = p(\vec{\phi}, \vec{X}_i) \rightarrow z_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = K(\tilde{q}\tilde{X}\tilde{q}^{-1} + \vec{\tau}) \tag{D.1}$$

In order to compute Jacobian of the projection function $\begin{bmatrix} x_i \\ y_i \end{bmatrix} = p(\vec{\phi}, \vec{X}_i)$ with respect to the visual measurement vector $\vec{\phi} = [\tau_x, \tau_y, \tau_z, q_s, q_a, q_b, q_c]^T$, let us define $\vec{e}$ as:

$$\vec{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \tilde{q}\tilde{X}\tilde{q}^{-1} + \vec{\tau} \tag{D.2}$$

Then, the definitions of x_i and y_i is simplified to:

$$x_i = f_x \frac{e_1}{e_3} + c_x \tag{D.3a}$$

$$y_i = f_y \frac{e_2}{e_3} + c_y \tag{D.3b}$$

where f_x, f_y, c_x and c_y are elements of camera calibration matrix K defined in Appendix C.

Computation of Jacobian of $\vec{e}$ with respect to $\vec{\phi}$ is rather straightforward.

$$J_{\vec{e}}(\vec{\phi}) = \begin{bmatrix} I_{3\times 3} & J_{\tilde{q}X\tilde{q}^*}(\tilde{q}) \end{bmatrix} \tag{D.4}$$

$J_{\tilde{q}X\tilde{q}^*}(\tilde{q})$ is defined in Appendix B.20. If we shortly represent $J_{\vec{e}}(\tilde{q})$ with J and its elements with $J_{(r,c)}$, the Jacobian $J_{p(\vec{\phi},\vec{X}_i)}(\vec{\phi})$ becomes:

$$J_{p(\vec{\phi},\vec{X}_i)}(\vec{\phi}) = \begin{bmatrix} \frac{f_x}{e_3} & 0 \\ 0 & \frac{f_y}{e_3} \\ -\frac{f_x e_1}{e_3^2} & -\frac{f_y e_2}{e_3^2} \\ f_x \frac{J_{(1,4)} e_3 - e_1 J_{(3,4)}}{e_3^2} & f_y \frac{J_{(2,4)} e_3 - e_2 J_{(3,4)}}{e_3^2} \\ f_x \frac{J_{(1,5)} e_3 - e_1 J_{(3,5)}}{e_3^2} & f_y \frac{J_{(2,5)} e_3 - e_2 J_{(3,5)}}{e_3^2} \\ f_x \frac{J_{(1,6)} e_3 - e_1 J_{(3,6)}}{e_3^2} & f_y \frac{J_{(2,6)} e_3 - e_2 J_{(3,6)}}{e_3^2} \\ f_x \frac{J_{(1,7)} e_3 - e_1 J_{(3,7)}}{e_3^2} & f_y \frac{J_{(2,7)} e_3 - e_2 J_{(3,7)}}{e_3^2} \end{bmatrix}^T \tag{D.5}$$

APPENDIX E

COMPUTATION OF JACOBIAN OF MATRICES USED IN METRIC VELOCITY ESTIMATION

The equation relating the velocity vector with matrix D and $\vec{f}$, defined in Section 3 is:

$$D \begin{bmatrix} v_x \\ v_y \\ v_z \\ z_n \end{bmatrix} = \vec{f} \tag{E.1}$$

This equation is derived from the four equations presented below.

$$\begin{aligned} v_x t1(x'_{n-1}[\tilde{q}_{x|1^*}]_c - [\tilde{q}_{x|1^*}]_a) + v_y t1(x'_{n-1}[\tilde{q}_{y|1^*}]_c - [\tilde{q}_{y|1^*}]_a) + v_z t1(x'_{n-1}[\tilde{q}_{z|1^*}]_c - [\tilde{q}_{z|1^*}]_a) + \\ z_n(x'_{n-1}[\vec{x}'_{n|1^*}]_c - [\vec{x}'_{n|1^*}]_a) = [\vec{\eta}_{1|1^*}]_a - x'_{n-1}[\vec{\eta}_{1|1^*}]_c \end{aligned} \tag{E.2}$$

$$\begin{aligned} v_x t1(y'_{n-1}[\tilde{q}_{x|1^*}]_c - [\tilde{q}_{x|1^*}]_b) + v_y t1(y'_{n-1}[\tilde{q}_{y|1^*}]_c - [\tilde{q}_{y|1^*}]_b) + v_z t1(y'_{n-1}[\tilde{q}_{z|1^*}]_c - [\tilde{q}_{z|1^*}]_b) + \\ z_n(y'_{n-1}[\vec{x}'_{n|1^*}]_c - [\vec{x}'_{n|1^*}]_b) = [\vec{\eta}_{1|1^*}]_b - y'_{n-1}[\vec{\eta}_{1|1^*}]_c \end{aligned} \tag{E.3}$$

$$\begin{aligned} v_x t2(x'_{n-2}[\tilde{q}_{x|2^*}]_c - [\tilde{q}_{x|2^*}]_a) + v_y t2(x'_{n-2}[\tilde{q}_{y|2^*}]_c - [\tilde{q}_{y|2^*}]_a) + v_z t2(x'_{n-2}[\tilde{q}_{z|2^*}]_c - [\tilde{q}_{z|2^*}]_a) + \\ z_n(x'_{n-2}[\vec{x}'_{n|2^*}]_c - [\vec{x}'_{n|2^*}]_a) = [\vec{\eta}_{2|2^*}]_a - x'_{n-2}[\vec{\eta}_{2|2^*}]_c \end{aligned} \tag{E.4}$$

$$\begin{aligned} v_x t2(y'_{n-2}[\tilde{q}_{x|2^*}]_c - [\tilde{q}_{x|2^*}]_b) + v_y t2(y'_{n-2}[\tilde{q}_{y|2^*}]_c - [\tilde{q}_{y|2^*}]_b) + v_z t2(y'_{n-2}[\tilde{q}_{z|2^*}]_c - [\tilde{q}_{z|2^*}]_b) + \\ z_n(y'_{n-2}[\vec{x}'_{n|2^*}]_c - [\vec{x}'_{n|2^*}]_b) = [\vec{\eta}_{2|2^*}]_b - y'_{n-2}[\vec{\eta}_{2|2^*}]_c \end{aligned} \tag{E.5}$$

Following the equations (E.2) - (E.5), the matrix D, with elements d_{11} - d_{44} and the column

matrix $\vec{f}$ with elements f_1 - f_4 is then defined as:

$$\text{Eq. (3.25): } v_x d_{11} + v_y d_{12} + v_z d_{13} + z_n d_{14} = f_1 \tag{E.6a}$$
$$\text{Eq. (3.26): } v_x d_{21} + v_y d_{22} + v_z d_{23} + z_n d_{24} = f_2 \tag{E.6b}$$
$$\text{Eq. (3.27): } v_x d_{13} + v_y d_{23} + v_z d_{33} + z_n d_{34} = f_3 \tag{E.6c}$$
$$\text{Eq. (3.28): } v_x d_{41} + v_y d_{42} + v_z d_{43} + z_n d_{44} = f_4 \tag{E.6d}$$

The function to compute the velocity vector is defined as:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ z_n \end{bmatrix} = D^{-1} f = l(\vec{o}) \tag{E.7}$$

with the input vector $\vec{o}$ being:

$$\vec{o} = \begin{bmatrix} \vec{x}_n'^T & \vec{x}_{n-1}'^T & \vec{x}_{n-2}'^T & \vec{q}_1^T & \vec{q}_2^T & \vec{\alpha}_{n-1}^T & \vec{\alpha}_{n-1}^T \end{bmatrix}^T \tag{E.8}$$

As described in Section 3, we need to find derivatives of D and $\vec{f}$ with respect to the elements of the input vector. In this chapter, we present these derivatives.

Let us firstly define the derivative of a rotation operation with respect to quaternion parameters. Rotating a point with a quaternion is conducted as:

$$P' = qPq^{-1} = RP \tag{E.9}$$

(B.8) shows the equivalent rotation matrix. The derivatives of P' with respect to the quaternion parameters q_s, q_a, q_b and q_c are presented in (B.16) - (B.19). We will denote the quaternion representing $\dfrac{\partial P'}{\partial q_s}$ as $\tilde{J}_{P'}(q_s)$.

Now, the partial derivatives are presented.

$$\frac{\partial D}{\partial x_n'} = \begin{bmatrix} 0 & 0 & 0 & d_{11}/t_1 \\ 0 & 0 & 0 & d_{21}/t_1 \\ 0 & 0 & 0 & d_{31}/t_1 \\ 0 & 0 & 0 & d_{41}/t_1 \end{bmatrix} \tag{E.10}$$

$$\frac{\partial D}{\partial y_n'} = \begin{bmatrix} 0 & 0 & 0 & d_{12}/t_1 \\ 0 & 0 & 0 & d_{22}/t_1 \\ 0 & 0 & 0 & d_{32}/t_1 \\ 0 & 0 & 0 & d_{42}/t_1 \end{bmatrix} \tag{E.11}$$

$$\frac{\partial D}{\partial x'_{n-1}} = \begin{bmatrix} t_1[\tilde{q}_{x|1}]_c & t_1[\tilde{q}_{y|1}]_c & t_1[\tilde{q}_{z|1}]_c & [\vec{x}_{n|1}]_c \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{E.12}$$

$$\frac{\partial D}{\partial y'_{n-1}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ t_1[\tilde{q}_{x|1}]_c & t_1[\tilde{q}_{y|1}]_c & t_1[\tilde{q}_{z|1}]_c & [\vec{x}_{n|1}]_c \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{E.13}$$

$$\frac{\partial D}{\partial x'_{n-2}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ t_2[\tilde{q}_{x|2}]_c & t_2[\tilde{q}_{y|2}]_c & t_2[\tilde{q}_{z|2}]_c & [\vec{x}_{n|2}]_c \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{E.14}$$

$$\frac{\partial D}{\partial y'_{n-2}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ t_2[\tilde{q}_{x|2}]_c & t_2[\tilde{q}_{y|2}]_c & t_2[\tilde{q}_{z|2}]_c & [\vec{x}_{n|2}]_c \end{bmatrix} \tag{E.15}$$

For $\xi = \{s, a, b, c\}$,

$$\frac{\partial D}{\partial q'_{1,\xi}} = \begin{bmatrix} t_1(x'_{n-1}[\tilde{J}_{\tilde{q}_{x|1^*}}(q_{1,\xi})]_c - [\tilde{J}_{\tilde{q}_{x|1^*}}(q_{1,\xi})]_a) & t_1(x'_{n-1}[\tilde{J}_{\tilde{q}_{y|1^*}}(q_{1,\xi})]_c - [\tilde{J}_{\tilde{q}_{y|1^*}}(q_{1,\xi})]_a) \\ t_1(y'_{n-1}[\tilde{J}_{\tilde{q}_{x|1^*}}(q_{1,\xi})]_c - [\tilde{J}_{\tilde{q}_{x|1^*}}(q_{1,\xi})]_b) & t_1(y'_{n-1}[\tilde{J}_{\tilde{q}_{y|1^*}}(q_{1,\xi})]_c - [\tilde{J}_{\tilde{q}_{y|1^*}}(q_{1,\xi})]_b) \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} t_1(x'_{n-1}[\tilde{J}_{\tilde{q}_{z|1^*}}(q_{1,\xi})]_c - [\tilde{J}_{\tilde{q}_{z|1^*}}(q_{1,\xi})]_a) & x'_{n-1}[\tilde{J}_{\vec{x}_{n|1^*}}(q_{1,\xi})]_c - [\tilde{J}_{\vec{x}_{n|1^*}}(q_{1,\xi})]_a \\ t_1(y'_{n-1}[\tilde{J}_{\tilde{q}_{z|1^*}}(q_{1,\xi})]_c - [\tilde{J}_{\tilde{q}_{z|1^*}}(q_{1,\xi})]_b) & y'_{n-1}[\tilde{J}_{\vec{x}_{n|1^*}}(q_{1,\xi})]_c - [\tilde{J}_{\vec{x}_{n|1^*}}(q_{1,\xi})]_b \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{E.16}$$

$$\frac{\partial D}{\partial q'_{2,\xi}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ t_2(x'_{n-2}[\tilde{J}_{\tilde{q}_{x|2^*}}(q_{2,\xi})]_c - [\tilde{J}_{\tilde{q}_{x|2^*}}(q_{2,\xi})]_a) & t_2(x'_{n-2}[\tilde{J}_{\tilde{q}_{y|2^*}}(q_{2,\xi})]_c - [\tilde{J}_{\tilde{q}_{y|2^*}}(q_{2,\xi})]_a) \\ t_2(y'_{n-2}[\tilde{J}_{\tilde{q}_{x|2^*}}(q_{2,\xi})]_c - [\tilde{J}_{\tilde{q}_{x|2^*}}(q_{2,\xi})]_b) & t_2(y'_{n-2}[\tilde{J}_{\tilde{q}_{y|2^*}}(q_{2,\xi})]_c - [\tilde{J}_{\tilde{q}_{y|2^*}}(q_{2,\xi})]_b) \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} t_2(x'_{n-2}[\tilde{J}_{\tilde{q}_{z|2^*}}(q_{2,\xi})]_c - [\tilde{J}_{\tilde{q}_{z|2^*}}(q_{2,\xi})]_a) & x'_{n-2}[\tilde{J}_{\vec{x}_{n|2^*}}(q_{2,\xi})]_c - [\tilde{J}_{\vec{x}_{n|2^*}}(q_{2,\xi})]_a \\ t_2(y'_{n-2}[\tilde{J}_{\tilde{q}_{z|2^*}}(q_{2,\xi})]_c - [\tilde{J}_{\tilde{q}_{z|2^*}}(q_{2,\xi})]_b) & y'_{n-2}[\tilde{J}_{\vec{x}_{n|2^*}}(q_{2,\xi})]_c - [\tilde{J}_{\vec{x}_{n|2^*}}(q_{2,\xi})]_b \end{bmatrix} \tag{E.17}$$

$$\frac{\partial D}{\partial \alpha'_{(n-1,x)}} = \frac{\partial D}{\partial \alpha'_{(n-1,y)}} = \frac{\partial D}{\partial \alpha'_{(n-1,z)}} = \frac{\partial D}{\partial \alpha'_{(n-2,x)}} = \frac{\partial D}{\partial \alpha'_{(n-2,y)}} = \frac{\partial D}{\partial \alpha'_{(n-2,z)}} = 0 \tag{E.18}$$

$$\frac{\partial \vec{f}}{\partial x'_n} = \frac{\partial \vec{f}}{\partial y'_n} = 0 \tag{E.19}$$

$$\frac{\partial \vec{f}}{\partial x'_{n-1}} = \begin{bmatrix} -[\vec{\eta}_{1|1^*}]_c \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{E.20}$$

$$\frac{\partial \vec{f}}{\partial y'_{n-1}} = \begin{bmatrix} 0 \\ -[\vec{\eta}_{1|1^*}]_c \\ 0 \\ 0 \end{bmatrix} \tag{E.21}$$

$$\frac{\partial \vec{f}}{\partial x'_{n-2}} = \begin{bmatrix} 0 \\ 0 \\ -[\vec{\eta}_{2|2^*}]_c \\ 0 \end{bmatrix} \tag{E.22}$$

$$\frac{\partial \vec{f}}{\partial y'_{n-2}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -[\vec{\eta}_{2|2^*}]_c \end{bmatrix} \tag{E.23}$$

For $\xi = \{s, a, b, c\}$,

$$\frac{\partial \vec{f}}{\partial q'_{1,\xi}} = \begin{bmatrix} 0 \\ 0 \\ -(\frac{1}{2}t_1^2 + t_1 t_3)([\tilde{q}_2^* \tilde{J}_{\alpha_{n-1}}(q_{1,\xi})\tilde{q}_2]_a - x_{n-2}[\tilde{q}_2^* \tilde{J}_{\alpha_{n-1}}(q_{1,\xi})\tilde{q}_2]_c) \\ -(\frac{1}{2}t_1^2 + t_1 t_3)([\tilde{q}_2^* \tilde{J}_{\alpha_{n-1}}(q_{1,\xi})\tilde{q}_2]_b - y_{n-2}[\tilde{q}_2^* \tilde{J}_{\alpha_{n-1}}(q_{1,\xi})\tilde{q}_2]_c) \end{bmatrix} \tag{E.24}$$

$$\frac{\partial \vec{f}}{\partial q'_{2,\xi}} = \begin{bmatrix} 0 \\ 0 \\ -(\frac{1}{2}t_1^2 + t_1 t_3)([\tilde{J}_{\alpha_{n-1|1}}(q_{2,s})]_a - x_{n-2}[\tilde{J}_{\alpha_{n-1|1}}(q_{2,s})]_c) \\ -(\frac{1}{2}t_1^2 + t_1 t_3)([\tilde{J}_{\alpha_{n-1|1}}(q_{2,s})]_b - y_{n-2}[\tilde{J}_{\alpha_{n-1|1}}(q_{2,s})]_c) \end{bmatrix} \tag{E.25}$$

$$\frac{\partial \vec{f}}{\partial \alpha'_{(n-1,x)}} \begin{bmatrix} -\frac{1}{2}t_1^2 \\ 0 \\ ([q_{x|3^*}]_a - x'_{n-2}[q_{x|3^*}]_c)(\frac{1}{2}t_1^2 + t_1 t_3) \\ ([q_{x|3^*}]_b - x'_{n-2}[q_{x|3^*}]_c)(\frac{1}{2}t_1^2 + t_1 t_3) \end{bmatrix} \tag{E.26}$$

$$\frac{\partial \vec{f}}{\partial \alpha'_{(n-1,y)}} \begin{bmatrix} 0 \\ -\frac{1}{2}t_1^2 \\ ([q_{y|3^*}]_a - x'_{n-2}[q_{y|3^*}]_c)(\frac{1}{2}t_1^2 + t_1 t_3) \\ ([q_{y|3^*}]_b - x'_{n-2}[q_{y|3^*}]_c)(\frac{1}{2}t_1^2 + t_1 t_3) \end{bmatrix} \tag{E.27}$$

$$\frac{\partial \vec{f}}{\partial \alpha'_{(n-1,z)}} \begin{bmatrix} \frac{1}{2}t_1^2 x_{n-1} \\ \frac{1}{2}t_1^2 y_{n-1} \\ ([q_{z|3^*}]_a - x'_{n-2}[q_{z|3^*}]_c)(\frac{1}{2}t_1^2 + t_1 t_3) \\ ([q_{z|3^*}]_b - x'_{n-2}[q_{z|3^*}]_c)(\frac{1}{2}t_1^2 + t_1 t_3) \end{bmatrix} \tag{E.28}$$

$$\frac{\partial \vec{f}}{\partial \alpha'_{(n-2,x)}} \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{2}t_3^2 \\ 0 \end{bmatrix} \tag{E.29}$$

$$\frac{\partial \vec{f}}{\partial \alpha'_{(n-2,y)}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\frac{1}{2}t_3^2 \end{bmatrix} \tag{E.30}$$

$$\frac{\partial \vec{f}}{\partial \alpha'_{(n-2,z)}} \begin{bmatrix} 0 \\ 0 \\ \frac{1}{2}t_3^2 x_{n-2} \\ \frac{1}{2}t_3^2 y_{n-2} \end{bmatrix} \tag{E.31}$$