CAMERA TRAJECTORY ESTIMATION FOR INDOOR ROBOT ODOMETRY USING STEREO IMAGES AND INERTIAL MEASUREMENTS

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

BY

ANIL HORASAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

SEPTEMBER 2016

Approval of the thesis:

CAMERA TRAJECTORY ESTIMATION FOR INDOOR ROBOT ODOMETRY USING STEREO IMAGES AND INERTIAL MEASUREMENTS

submitted by **ANIL HORASAN** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Science** _____

Prof. Dr. Tuna Balkan
Head of Department, **Mechanical Engineering** _____

Assoc. Prof. Dr. Melik Dölen
Supervisor, **Mechanical Engineering** _____

Examining Committee Members:

Assoc. Prof. Dr. Erhan İlhan Konukseven
Dept. of Mechanical Engineering, METU _____

Assoc. Prof. Dr. Melik Dölen (Supervisor)
Dept. of Mechanical Engineering, METU _____

Assoc. Prof. Dr. Yiğit Yazıcıoğlu
Dept. of Mechanical Engineering, METU _____

Asst. Prof. Dr. Ali Emre Turgut
Dept. of Mechanical Engineering, METU _____

Asst. Prof. Dr. Kutluk Bilge Arıkan
Dept. of Mechatronics Engineering, Atılım University _____

Date: 09.09.2016

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 : ANIL HORASAN

Signature :

ABSTRACT

CAMERA TRAJECTORY ESTIMATION FOR INDOOR ROBOT ODOMETRY USING STEREO IMAGES AND INERTIAL MEASUREMENTS

Horasan, Anıl
M.S., Department of Mechanical Engineering
Supervisor: Assoc. Prof. Dr. Melik Dölen

September 2016, 135 pages

In this study, the development and implementation of an algorithm for stereo visual-inertial odometry are described. The study spans the complete process from analyzing the sensory data to the development of a robot odometry algorithm. The criteria for indoor visual-inertial odometry include using low-cost sensor systems, having an error less than five percent of the movement regardless of the distance covered, and building a robust algorithm in the presence of geometric and photometric invariances as well as noise. Utilizing the complementary characteristics of two different sensors, these steps are followed: First, orientation, velocity and position are estimated using inertial measurements. Second, the machine vision algorithm is developed consisting of feature detection and extraction, feature tracking in consecutive images, disparity map calculation, outlier rejection, motion estimation and optimization. Finally, inertial estimates are fused to visual pose estimates using EKF and a proposed filter. In this research, all the algorithms are implemented offline and tested using EuRoC MAV datasets. The results show that it is possible to achieve less than five percent positional errors in different indoor environments.

Keywords: Stereo Vision, Visual-Inertial Odometry, Robot Trajectory Estimation, Indoor Localization, Sensor Fusion

ÖZ

KAPALI ALANLARDA ROBOT KONUMLAMASI İÇİN STEREO GÖRÜNTÜLER VE EYLEMSİZLİK DUYUCUSU KULLANILARAK KAMERA YÖRÜNGESİNİN KESTİRİLMESİ

Horasan, Anıl
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Melik Dölen

Eylül 2016, 135 sayfa

Bu çalışmada bir görsel-ataletsel odometre algoritmasının geliştirilmesi ve uygulanması anlatılmıştır. Çalışma, duyu verisinin incelenmesinden robot odometre algoritmasına kadar olan bütün süreci kapsamaktadır. Kapalı alanlarda görsel/ataletsel odometre için belirlenen ölçütler, düşük maliyetli duyucu sistemlerin kullanılması, hareket edilen mesafeden bağımsız olarak yüzde beşten az hata payına sahip olunması ve gürültü, fotometrik ve geometrik değişimler altında gürbüz bir algoritma oluşturulmasıdır. İki farklı duyucu sisteminin tamamlayıcı özellikleri kullanılarak şu aşamalar izlenmiştir: İlk olarak, ataletsel duyuculardan gelen ölçümler kullanılarak doğrultu, hız ve konum tahmini yapılmıştır. İkinci olarak, özellik bulma ve çıkarma, özellik takibi, fark değeri haritası çıkarma, aykırı değerleri temizleme, hareket kestirimi ve en iyileme aşamalarıyla yapay görü algoritması oluşturulmuştur. Son olarak, EKF ve önerilen filtre ile tahminler görüntüden elde edilen tahminlerle birleştirilmiştir. Bu çalışmada tüm algoritmalar çevrimdışı olarak çalıştırılmış ve EuRoC MAV veri kümesi ile test edilmiştir. Sonuçlar, farklı kapalı alanlarda yüzde beşten daha düşük hataları yakalamanın mümkün olduğunu göstermektedir.

Anahtar Kelimeler: İkili Kamera Sistemi, Görsel-Ataletsel Odometre, Robot Yörünge Kestirimi, Kapalı Alan Konumlama, Duyucu Tümleştirme

To My Beloved Sister

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my supervisor, Assoc. Prof. Dr. Melik Dölen, for his guidance, support, and sincere help throughout my thesis study. His suggestions and encouragement helped me finalize the study.

I would also like to thank to the jury members Assoc. Prof. Dr. Erhan İlhan Konukseven, Assoc. Prof. Dr. Yiğit Yazıcıoğlu, Asst. Prof. Dr. Ali Emre Turgut, and Asst. Prof. Dr. Kutluk Bilge Arıkan for reviewing and evaluating my thesis. Their insightful comments contributed a lot to my study.

I owe gratitude to Asst. Prof. Dr. Buğra Koku who helped me shape the study through fruitful brainstorming discussions. His contributions are invaluable.

I would like to express my appreciation to my employers Murat Aykan, Fatih Altunel and Bülent Başaran for helping and supporting me. I am also grateful to my friend and colleague İsmail Mustafa Engiz for his help to analyze my work. In addition, I would like to thank to all my friends who encouraged me throughout my thesis.

My special thanks, however, are conveyed to my family. I feel the deepest gratitude to my mother, Hüsniye Horasan, and my father, Hikmet Horasan, for their never-ending support and faith in me. Without their encouragement, I could not have finished this thesis. I am particularly indebted to my sister, Seçil Horasan Doğan, and brother-in-law Hüseyin Can Doğan, for their endless support and understanding. If it had not been for my sister, this thesis would not have been possible.

This study was supported by METU BAP grant no: 20150431 and TÜBİTAK 2220-A Graduate Student Grant Program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AR:	Augmented Reality
ARM:	Acorn RISC Machine
ATE:	Absolute Trajectory Error
BA:	Bundle Adjustment
BRIEF:	Binary Robust Independent Elementary Features
BRISK:	Binary Robust Invariant Scalable Keypoints
CCD:	Charge Coupled Device
CenSurE:	Center Surround Extrema
CMOS:	Complementary Metal-Oxide Semiconductor
CPU:	Central Processing Unit
CV:	Computer Vision
CWT:	Continuous Wavelet Transform
DLT:	Direct Linear Transform
DOF:	Degree-of-Freedom
EKF:	Extended Kalman Filter
EuRoC:	European Robotics Challenge
FAST:	Features from Accelerated Segment Test
FFT:	Fast Fourier Transform
FPGA:	Field Programmable Gate Array
FREAK:	Fast Retina Key point
GPS:	Global Positioning System
GT:	Ground Truth
IEKF:	Iterated Extended Kalman Filter
IMU:	Inertial Measurement Unit
INS:	Inertial Navigation System

IRU:	Inertial Reference Units
ISPKF:	Iterative Sigma Point Kalman Filter
KF:	Kalman Filter
KLT:	Kanade-Lucas-Tomasi
LIDAR:	Laser Imaging Detection and Ranging
LOG:	Laplacian of Gaussian
MAV:	Micro Air Vehicle
MEMS:	Micro Electro-Mechanical Systems
MSER:	Maximally Stable Extremal Regions
NASA:	National Aeronautics and Space Administration
OpenCV:	Open Computer Vision
ORB:	Oriented FAST and Rotated BRIEF
PSD:	Power Spectral Density
RANSAC:	Random Sample Consensus
RGB-D:	Red Green Blue – Depth
RMS:	Root Mean Square
SBA:	Sparse Bundle Adjustment
SfM:	Structure from Motion
SGM:	Semi-Global Matching
SIFT:	Scale Invariant Feature Transform
SLAM:	Simultaneous Localization and Mapping
SoC:	System-on-a-chip
SURF:	Speeded up Robust Features
SVD:	Singular Value Decomposition
SVO:	Semi-direct Visual Odometry
ToF:	Time-of-Flight
UKF:	Unscented Kalman Filter
VIO:	Visual-Inertial Odometry
VO:	Visual Odometry
V-SLAM:	Visual Simultaneous Localization and Mapping

CHAPTER 1

INTRODUCTION

The last few decades have witnessed the most significant leaps in the field of robotics. Especially, the development and use of unmanned vehicles have shown a growing trend [1] as the technology is developing rapidly and the world is becoming more autonomous. Accordingly, so far innovative household robotics such as robotic vacuum cleaners, cook and ironing robots, security homebots, and robotic lawnmowers have been developed and released to the market. Moreover, it is estimated that soon human driven vehicles are to going to be replaced by self-driving cars; help and rescue robots are going to be used in disasters in which the rescue teams are unable to reach; and transportation systems in big depots and warehouses are going to be operated by autonomous trucks. Considering the developments in the robotics field, it is highly important for these autonomous systems to know their location and navigate accordingly. This situation is called motion estimation or localization problem in the field of robotics.

Regarding the localization problem there have been a number of developments, the most important of which is the navigation systems in vehicles. For the vehicle navigation systems, an old technique to calculate the current position of a platform was dead reckoning, using previous pose and velocity information over a period of time and course. Adding encoders to dead reckoning systems, wheel odometry

incrementally estimates the distance travelled by a vehicle using the readings of wheel turns. These are still popular methods due to their simplicity and practicality. However, because of the accurate localization needs of different moving platforms, most of the navigation systems nowadays are relying on two primary technologies: *Global Positioning System (GPS)* and inertial navigation. GPS provides a global coordinate unless the navigating platform is outside the Earth or in urban areas, forests or indoors. Besides, GPS is subject to jamming and it is only owned and operated by US Government. Still, it is a widely used navigation system because of its global positioning and can provide less than 0.7 meters horizontal median error in open air [2]. On the other hand, inertial navigation systems can be easily degraded in long runs and must be corrected using GPS or another global localization system to know its location correctly. Without the fixes, inertial navigation systems are rather counterproductive.

Since GPS introduced a great solution to outdoor positioning, interest in indoor positioning problem has attracted attention in robotics studies. One of the most elegant solutions to this problem is to use sequential images from single or multiple cameras to estimate the motion of the platform with respect to a reference starting point. This problem is called *Visual Odometry (VO)* in the literature. Different from the wheel odometry, slippage cannot affect VO in uneven terrains. Furthermore, VO can be used by platforms without wheels, such as humanoid robots and unmanned air vehicles. Considering the pros and cons of wheel odometry, GPS and VO; space explorations and indoor applications are the main areas of usage for VO. In this particular study, a stereo visual-inertial odometry system is discussed, analyzed and developed.

Following the state-of-the-art two-part visual odometry tutorial [3, 4], a path extraction methodology is described and compared with the noted VO algorithms in the literature. Different combinations of feature detectors/descriptors, optimization techniques and fusion algorithms are experimented and compared among each other using available datasets. The optimal technique is verified with ground truth data in different indoor environments.

1.1. Motivation of the Thesis

One need in the robotic industry is that a robotic system will almost always need to localize itself while at the same time being able to see the actual world. In fact, in some autonomous systems, cameras have already been utilized for the purposes of obstacle detection and object recognition. At this very point, these cameras present in robotic vehicles can also be exploited to solve the localization problem. To this end, two functions of a robot – localizing itself and seeing the actual world– are combined. Thus, the solution of the localization problem through the use of visual sensors has led the main purpose of this thesis: Visual Odometry

In robotic navigation, vision has received a great amount of interest. Thanks to the developments in feature detectors and feature tracking innovative high-end cameras, and processing algorithms, the function of vision has gained so much importance that its contributions to robotic navigation have become inevitable and vital [5]. Vision provides not only a less drift-prone navigation than low-cost inertial navigation systems, but also supports indoor navigation which GPS does not.

There has been a growing trend in the use of visual sensors for robot localization. The importance of visual sensors in robot localization has increased along with the increase in the number of robotic systems. According to International Federation of Robotics, the total number of service robots sold in 2014 is increased by 11.5% compared to preceding year, and it is estimated to be further increased by 2018 [6] as shown in Figure 1.1.

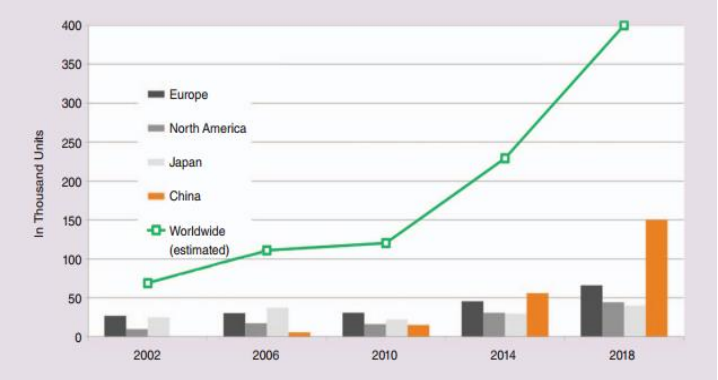

Figure 1.1 Estimated number of industrial robots in 2002-2018 [6]

Applications of service robotics can be as follows [6]:

- Household applications
    - Security robots,
    - Cleaning robots,
    - Handicap assistance robots,
    - Lawn mowing robots,
- Logistics applications
    - Automated guided vehicles,
    - Autonomous trucks and forklifts,
- Defense applications
    - Unmanned air and ground vehicles,
    - Search & rescue robots.
    - Bomb fighting and demining robots,
- Agriculture and livestock applications
    - Harvesting robots,
    - Mobile barn cleaners
    - Robotic fencers,
    - Milking robots,
- Medical applications
    - Surgery robots,
    - Rehabilitation robots,
    - Assistant robots.

These applications of robotic systems function not only outdoors but also indoors as illustrated in the list above. Although there are well-known navigational services such as GPS, employed in outdoor environments, they are not very common indoors. The problem is GPS cannot be used for the localization of indoor autonomous vehicles because satellite signals are thwarted by thick walls of buildings resulting with deflection in accurate localization [7]. What is more important is that indoor robotic applications require sub-meter accuracy in localization which GPS cannot provide.

Accordingly, the idea of developing an independent localization system that has stereo camera and inertial sensors which can be used on robotic platforms has evolved. While developing the algorithms, there were certain criteria that contributed to the motivation of the research. These include developing an open-source and low-cost VO system as well as using portable tools in its development stage such as an average laptop computer and available datasets. Such advantages as being open-source and low-cost will allow future developers to afford and use the system, which constitutes another motivation for this study.

For short, this study is inspired by emerging technologies in robotics and future breakthroughs. Brainstorming about technological developments and future breakthroughs, it is inevitable to question what these emerging technologies are, where the field of robotics is heading to, at what point the national industry is in this field, how researchers feel towards these developing technologies, what paths in the field are promising, and most importantly, what innovations are more needed by people. It is motivational to work on a project that is more needed and promising for the future of the field. It is also important to have an insight and an extensive know-how at the end of the work which will be beneficial for future studies.

In this scope, this thesis is set out to investigate the navigation and localization algorithms for an indoor robot due to the great potential in the robotics field. In addition, the algorithms have been implemented meticulously since the background to construct a completely autonomous robot can be made possible as a result of such research.

1.2. Objective and Evaluation Criteria

Building upon the motivation for this thesis, research question is asked as follows:

> *Using only cost-efficient stereo cameras and inertial sensors, can we robustly estimate the path of an indoors robotic platform?*

Before going into any work to answer this question, a list of criteria should be established to be able to evaluate the results at the end of the study. To this end, there

are 6 main criteria determined to evaluate the work. These are listed and explained below:

- **Accuracy and repeatability:** The system should have a repeatable error less than 5% of the movement for translation components and less than 10 degrees for rotational components. Since there will be a drift, translational error will be relative to the distance covered. Nevertheless, drift can be minimized and corrected with the help absolute indoor localization systems.
- **Robustness:** Since the system mostly relies on camera images, changes in illumination of the scene and noise in the environment will affect the quality of the localization. Therefore, VO algorithm should be robust in the presence of noise such as photometric and geometric invariances.
- **Cost efficiency:** Low-cost but effective sensors should be used to have an efficient overall system if a custom dataset is used. There are high resolution cameras in the market for less than 50 USD and MEMS IMUs for about 10 USD. However, high-end sensors can also be used for comparison with the low-cost sensors. If a dataset would be developed, total cost of the system should not exceed 350 USD for the whole system with the data acquisition cards and the other electronics. If an available dataset is used, effects of sensor quality should be discussed at the end of the study.
- **Offline calculation:** The system does not have to be working real-time. Since, according to the Moore's Law, it is known that the power of processors will double itself every two years, so that the system can be designed to work in real-time several years later. Thus, the computations will be done in an average laptop after data is acquired offline.
- **Open-source:** For the programming language and computer vision libraries, open-source software will be used as much as possible. Source codes should be available so that everyone can be able to contribute and develop the work and the localization system can be adapted to various robotic platforms.
- **Portable test setup:** This criterion is optional as the algorithm can be tested using available datasets. However, yet to explain briefly, if a test setup is built for evaluation of the algorithms, it should be portable so that measurements

could be taken from different indoor environments. This will also be important to see if the algorithms are robust or not, to the changes in different places.

Furthermore, there are also a couple of important considerations that have to keep in mind. These are not must-have criteria but nice-to-have. One of them is to create an easy-to-use, user-friendly and practical system for other researchers and developers so they do not experience any problems while using the algorithm. Furthermore, the system should be able to be used in or integrated to different operating systems and development environments.

1.3. The Outline of the Thesis

In this thesis, a stereo visual-inertial odometry methodology is analyzed and discussed with comparison with mainstream algorithms.

After this brief introduction in Chapter 1, Chapter 2 presents a review of literature on visual-only and visual-inertial odometry. Chapter 2 illustrates samples from previous related works with detailed description of used algorithms and experimental results.

In Chapter 3, theory and methodology for both visual and inertial odometry are described. Fusion of these two sensor inputs are also presented in this section. The chapter begins with a detailed analysis of methodology of VO and inertial navigation techniques. The chapter is concluded with the proposed solution and implementation details.

Chapter 4 consists of the experimental setup and utilized dataset. The chapter also includes the results of the previous works that are using the same datasets.

Chapter 5 presents experimental results of the proposed solution in different datasets. Inertial measurements results, VO results, data fusion and discussion of the results are presented respectively.

Chapter 6 gives a brief conclusion of the thesis. It has been accompanied with future work suggestions.

CHAPTER 2

LITERATURE SURVEY

Indoor localization is a multi-disciplinary problem which includes topics from both machine vision and dynamics. Before attempting to propose a solution to this problem, it is important to give an in-depth survey of the literature of previous related works. Having dealt with thoroughly, however, the literature reviews has been compiled mostly in machine vision field. Accordingly, this chapter has been organized as follows: First, a general thought of the main topic called *Structure from Motion (SfM)* is explored. Then, detailed literature of VO along with its components and its comparison with *Simultaneous Localization and Mapping (SLAM)* problem is presented. VO is divided into three subtopics for convenience. First, monocular VO, which uses a single camera for path extraction, is explained and its literature is revealed. Then, stereo VO and its state-of-art are explored. Furthermore, previous works about *Inertial Measurement Unit (IMU)* integration to VO are investigated. Finally, most influential works are explored in the closure part in order to be able to show similarities and differences and to make a comparison in the end.

2.1. Structure from Motion

Taking a set of image correspondences as the only input, SfM attempts to deduce or estimate the 3D structure of the viewed scene and/or 6D camera locations where the images have been captured. SfM has been a hot topic in machine vision and robotics after the Millennium, and it can be affirmed that it has advanced to a mature level with

commercial applications. 2d3 Sensing [8], acquired by Boeing in 2015, is one of the first companies that uses SfM in their products for film industry and aerial imagery. Likewise, inspired by Photo tourism [9], Microsoft revealed the software Photosynth [10], which can take 200 photos and turn them into a 3D photo in less than ten minutes on an average laptop. A detailed categorization of other SfM applications is available in [11].

Origins of SfM can be traced back to so-called photogrammetry, which is an old technique used for mapping [12]. Measuring topographic lengths and angles from images, photogrammetry reached to an important level, forming the fundamental theory of SfM. Machine vision community has been spurred by the developments in the field not only for its implementations, but also accuracy, speed and stability. The researchers have mostly focused on the automation of SfM, which is thought to enable its usage in mobile robotics. The study on SfM can be divided into three sub-categories. First of which are the assumptions done in the geometry, formulations and the calibration parameters as mentioned by Hartley and Zisserman [13]. Second one is the salient feature detection/description and tracking even in the presence of changes and invariances. Some of the most famous feature detectors and key point descriptors are presented in [14-25]. For the last category, autonomous feature matching from the structure is enabled, and in-frame optimization is made, which is often called rejection of the wrong data or outlier rejection [24]. Several methods have been proposed using these three subroutines in different combinations. Prior knowledge about the camera parameters and constraints on the environment have yielded better results. Using an optimization technique like *Bundle Adjustment (BA)* [25], estimates are refined by minimizing the reprojection error.

2.2. Visual Odometry

Estimation of the egomotion of a mobile platform, which is the motion part of the SfM problem, has also been tackled by robotics community from a slightly different point of view. The so-called VO, a term coined in 2004 by David Nistér [26], uses one or more cameras placed on a platform as the main input, and then calculates the motion of the platform. Optionally, other sensors such as encoders, *Micro Electro-Mechanical*

Systems (MEMS) gyro and accelerometers, laser range finders, and the like are used along with the visual input to calibrate or enhance the estimation.

Contrary to SfM, VO aims to determine the motion from the structure, in other words, from the features/objects in the environment. Camera sensor captures consecutive images with sufficient overlap so that a structure or a feature can be tracked and the motion in six *degree-of-freedom (DOF)* can be calculated. This topic has been studied vigorously since 1980s. The most propelling reason for the development of VO is the NASA Mars exploration program, which consists of designing all-terrain rovers needing to eliminate slipping of the wheels. The studies started with landmark papers [27, 28], continued with various other works [29, 30], and then were summarized and concluded with field reports [31].

VO can be considered as a sub-problem of *Visual Simultaneous Localization and Mapping (V-SLAM)* problem. While VO aims to extract the local camera path incrementally, Visual SLAM maps the environment, discloses global camera path, and optimizes map and path using the loop closure technique. Since V-SLAM somehow encompasses VO, it is important to take a look at corresponding literature. There are mainly two methods used in V-SLAM: One is uses filtering methods which utilize probability distribution to fuse information gathered from images. the other method uses key frame methods which apply bundle adjustment to selected frames. Durrant-Whyte and Bailey [32, 33] explain the methodology used in V-SLAM in a two-part-tutorial and a survey paper by Fuentes et al. [34] presents a brief and comprehensive review of the state-of-the-art by dividing V-SLAM into two as visual-only and visual-inertial.

2.2.1. Monocular Visual Odometry

Using only one camera to estimate the motion of the platform has gained a great amount of importance and interest following the widespread use of the smartphones [3]. For users' localization and navigation purposes in indoor environments, monocular scheme should be used since smartphones usually have one camera on each side. However, using only one camera instead of two has a drawback of estimating the

position as a ratio due to unknown absolute scale. This can be compensated by knowing the environment in advance or using external sensors such as proximity sensors or IMUs.

Monocular VO, considered to be one of the active areas of research in computer vision society, can be divided into two main groups as feature-based methods and appearance-based methods. While salient consistent features are found and compared to other frames in the former approach, the latter one examines the intensity information of the pixels. Moreover, there are studies combining these two methods to bring out a more stable algorithm.

One of the first and well-known studies of monocular VO is by Nister et al. [26], in which improved Harris corner detector and *Kanade-Lucas-Tomasi (KLT) tracker* have been utilized. The paper suggests the use of five-point *Random Sample Consensus (RANSAC)* [35] for outlier rejection, which is then supervened by other works [36, 37]. A long run (about 2.5 km) is accomplished by Tardif et al. [37], where an algorithm is introduced to separate translation and rotation estimates.

Another study in which the Harris corners have been employed is by [38] which have improved to real-time odometry in 1 km test setup. The results are refined with local bundle adjustment and compared to a ground truth of accurate GPS system. Later, Scaramuzza et al. [39] have used non-holonomic constraints and minimized the number of necessary feature points to track the camera motion and utilized a one-point RANSAC for the rejection of the wrong data.

Although appearance-based methods were not used at the beginning of the VO development stage, it has recently favored by most. In one of the earlier works, [40] uses not only appearance for rotation estimation but also features for translation and scale estimations. Preceeding works are more successful using only appearance-based methods such as in [41], which is called *Semi-direct Visual Odometry (SVO)*. Without using the features, the researchers recommend the use of pixel intensities to obtain sub-pixel accuracy in high frame rates. They also use probabilistic mapping methods for

outlier rejection at very high speeds up to 300 fps. It is said to add robustness to the estimation.

2.2.2. Stereo Visual Odometry

The idea of using dual cameras to combine the depth information and motion estimation has a longer history than monocular systems. From 1977 to 2003, researchers focused specifically on the problem of navigation on the Mars' surface as mentioned before, in spite of the inadequate and insufficient number of feature detection algorithms. All the works up to 2007 have been analyzed and summarized by NASA scientists in [42, 31]. Final version of Mars rover localization algorithm uses Harris corners with error covariance matrix of each feature point and 6-point RANSAC for outlier rejection. Depending on the development of fast corner detectors and invariant feature descriptors, accuracy of stereo VO has increased comparatively.

Howard [43] experiments a neat method combining previous knowledge of VO with novel key points by using dense disparity images. The algorithm makes no prior assumptions on the environment and utilizes standard feature detectors Harris or FAST. It constructs a score matrix and computes the matching features in a repetitive manner. The key step of the algorithm is pointed out as finding the largest set of consistent features. Without using a RANSAC scheme, algorithm detects the inliers in the feature sets rather than rejecting the outliers. Achieving an error of 0.25% at 50 Hz in a 400 m run, this study shows that the better accuracies could be accomplished.

Without reconstructing the 3D world, Geiger et al. [44] uses trifocal geometry between image triplets (which is called "bucketing", suggesting that majority of the features lie on the static background) in combination with RANSAC to estimate the motion. *Iterative Sigma Point Kalman Filter (ISPKF)* is employed to fuse the data and to deal with the nonlinearities. Although giving similar results, it is shown that ISPKF is superior to both *Iterated Extended Kalman Filter (IEKF)* and *Unscented Kalman Filter (UKF)* because of considerably low execution times.

Konolige et al. [45], on the contrary, uses a specifically developed, feature detector called *Center Surround Extrema (CenSurE)* [46], which is claimed as complex as SIFT

but as fast as Harris corner detector. Corresponding features are then found in the second image using dense stereo. Their algorithm performs a 3-point RANSAC and *sparse bundle adjustment (SBA)* to N recent frames. In the final step, IMU data is fused for refinement using *Extended Kalman Filter (EKF).*

With the help of IMU fusion, [47] matches the egomotion with key-frame based VO using an increased performance *Semi-Global Matching (SGM)* [48]. This work is important due to two aspects:

- High computing rates and real time applications are possible with the use of *Field Programmable Gate Arrays (FPGAs).*
- VO systems can be as small as a handheld device under a kilogram.

In a more recent work, Siegwart et al. [49] also uses visual-inertial system and FPGA pre-processing for SLAM of Micro Aerial Vehicles *(MAVs).*

Consequently, "multi-frame feature integration" technique is used in [50] suggesting the utilization of the whole history of the tracked points to correct the tracking error. Badino et al. [50] claim that this statistical analysis of the feature tracking error increases the accuracy up to 65% while adding only 3.8% computational cost. The authors also indicate that their algorithm outperforms previous VO methods with a translational error of 1.62% and rotational error of 0.0062 °/m.

2.2.3. Visual-Inertial Odometry

GPS is a highly accurate, absolute and widespread system that is being used by almost everyone in the world. However, GPS cannot be used in other planets; its usage is constrained by thick walls for indoors and by trees, valleys for outdoors. For the indoor applications, which are the primary concern of this work, other sensors such as IMUs can be utilized. Standing for inertial measurement unit, IMUs basically consist of integrated accelerometers and gyros, providing a 6-DOF. Optionally, some IMUs include 3-axis magnetometers and barometric pressure sensors, making the sensor system 10-DOF. Precise, high-end and very high-cost mechanical IMUs have been used for navigational purposes of aerial and naval vehicles for years. Nevertheless,

low-cost integrated IMUs quickly degrade over time unless corrected. If used with VO, position and orientation information taken from IMU, which has been a part of indoor positioning research for years, can be corrected.

In one of the studies, Konolige et al. [45] use 6-DOF IMU in combination with stereo cameras and determines the global orientation and rotation at each frame. It is shown that even a very low-cost IMU dramatically increases the accuracy up to 10-14 times. IMU is fused via loose coupling in every frame. Then, simple, small-sized EKF procedure is used as a second stage of fusion. Weiss et al. [51] uses a 20-gram monocular camera - IMU setup to estimate 6-DOF pose of MAVs in real-time by utilizing optical-flow based positioning. Schmid and Hirchmüller [47] combined stereo camera system and IMU into an 830-gram handheld device consisting of CPU, FPGA and ARM-processor boards. Key frame based position estimation in 15 Hz and less than 1% positioning error are achieved. Use of tightly coupled visual-inertial framework with ARM-processor and FPGA for preprocessing have also been worked by Nikolic et al. [49], showing that under 1% positioning error can be obtained in real-time indoor positioning. It is showed that their system is more accurate and robust compared to visual only or loosely coupled systems.

There is an approach that exploits ground plane constraints while triangulating the 3D data by Panahandeh et al. [52]. The study is done indoors and imposes visual constraints to inertial measurements via *Sigma-Point Kalman Filter*. Martinelli [53] also develops a closed-form solution to indoor fusion of IMU and monocular camera, showing that the result will either have a unique solution, two distinct solutions or infinite solutions depending on the trajectory and the feature point locations.

Following the widespread use of smartphones and camera-equipped mobile devices, researchers have focused on developing VO algorithms using built-in IMU and camera of these devices. Since almost every phone has one camera on each side, most of the research has focused on using monocular VO and internal calibration and fusion of built-in sensors. One of the first attempts of using cameras of mobile phones is in 2005 by Nokia engineers [54], where a feature-based tracking algorithm facilitates user interaction for phone applications. Goldberg and Matthies [55] use *OMAP3530*

system-on-a-chip (SoC) processors with stereo camera and IMU to show that VO can be run real-time with very compact, low-power, yet high performance processors. Engel et al [56] present a direct monocular VO scheme rather than feature (key point) based method for *Augmented Reality (AR)* applications. In a direct method, they track the new images using direct image alignment and the geometry is preserved as a semi-dense map as shown in Figure 2.1 Unlike the previous works, Mourikis et al. [57] models rolling-shutter camera rather than global shutter camera since most of the smartphones are equipped with this type. They claim that 0.8% error accumulation of the distance travelled is possible by fusing camera and IMU of a smartphone by EKF based *Visual-Inertial Odometry (VIO)*. Tomazic and Skrjanc [58] also use VIO on a smartphone with KLT and RANSAC and claims less than 1-m error over 27-m run.

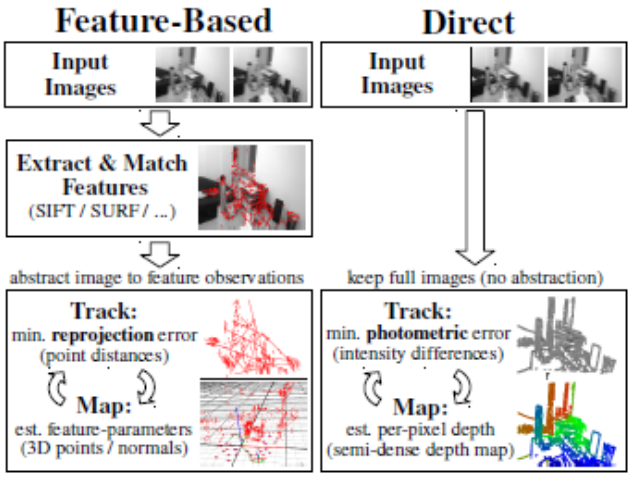

Figure 2.1 Feature-based methods vs. Direct methods [59]

2.3. Closure

Due to the advancements in technology, studies on VO have gained momentum. There are many studies on pose estimation and visual-inertial odometry. The literature indicates that these studies focus on monocular camera systems since outdoor navigation is mostly explored. In this study, a stereo camera system is used for trajectory estimation in indoor environments.

CHAPTER 3

BACKGROUND AND PROPOSED METHODOLOGY

In this chapter, the theory of VO is given in detail with the specific numerical analysis of the used algorithm. Next, inertial navigation and the usage of IMU in the study is described. Then, the fusion of accelerometer and gyroscope using complementary filter and vision sensors and the inertial sensors is explained. Finally, proposed methodology is presented including inertial navigation, VO and fusion of the inertial data to the visual position estimation.

3.1. Visual Odometry

Using only camera images, VO estimates the motion of a platform which could vary from a human to a robot [3]. When Moravec used stereo cameras for estimating robotic motions back in 1980s, VO simply refers to the estimation of the camera motion using image sequences. That is, it is one of the means of finding a robot's trajectory. Nister et al. [26] described the generation of VO through the estimated motion from visual input. It was concluded in their study that to estimate a vehicle's position, VO is an invaluable means. Therefore, VO is employed in a number of fields ranging from robotics to automotive [3].

For VO to operate effectively, there are a number of requisitions. To begin with, the environment is required to be illuminated adequately. Sufficient texture is also needed

so that the motions can be extracted. Moreover, without adequate scene overlapping, subsequent images may not be used for feature matching and tracking. Since the aim of this thesis is to work in indoor environments, it is assumed that there are sufficient features present in the images; however, illumination remains a challenge and a limitation of the study.

In capturing the camera motions, there are two different image acquisition sensors available in today's technology: namely *Charged Coupled Device (CCD)* and c*omplementary metal–oxide–semiconductor (*CMOS*)* cameras. These two devices can be compared based on their pros and cons [60]. One advantage of CCD lies in the aspect of noise. While CCD sensors create high-quality and low-noise images, CMOS sensors are more susceptible to noise. Another fundamental difference between them is their readout architecture [61]. CCD sensors employ interline charge transfer and have photo detectors as well as vertical and horizontal CCDs. Upon immediate exposure, charge is readout through charge transfer from each photo-detector to vertical and horizontal CCDs for all pixels. CMOS sensors; however, use horizontal line for pixels and vertical line for charge, resulting in transformation of one row to capacitors. Other advantages of CCD are related to sensitivity and power. CCD offers greater sensitivity and fidelity whereas CMOS offers lower sensitivity. The former consumes 100 times more power than its counterpart. On the other hand, CCD entails specialized assembly lines to produce, yet the production of CMOS is relatively easier. The first one is older and requires more developed technology but the latter is more inexpensive. CCD is larger in size. In contrast, the other is smaller. CCD can be less responsive; however, CMOS provides more signals per unit. While CCD is akin to blooming, CMOS possesses natural anti-blooming.

In addition, different lenses and configurations are available. Camera types range from passive to active or hybrid types as follows:

- **Passive cameras**
    - ***Monocular***: Monocular cameras are single ones. Their trajectory estimation is only possible up to a scale using minimum of three consecutive frames [3].

- ***Stereo***: Stereo cameras refer to two camera systems. Its biggest appeal is that exact trajectory can be estimated by triangulating feature points and calculating depth information [3]. More data in the scene makes stereo VO more robust; but when the distance between the stereo system and scene is greater than that of stereo camera baseline, stereo vision converges to monocular vision.
- ***Trinocular***: Trinocular cameras refer to three camera systems, which are more robust than stereo but wide baseline trinocular systems are not useful, because they take up more space than monocular or stereo systems. Camera trajectory can be obtained using trifocal tensor as mentioned in [13].
- ***Omnidirectional***: Omnidirectional cameras can see a view of all 360 degrees on a horizontal plane or almost the whole sphere. Catching the light in all dimensions that see the focal point, omnidirectional cameras capture the sphere. Nevertheless, practically it is usually not a full sphere of 360 degrees. They play a critical role in robotics or panoramic photography which requires a vast visual field. Particularly in VO, omnidirectional cameras are frequently employed concerning the SLAM problems.

- **Active cameras**
  - ***LIDAR***: Lidar cameras include a laser, scanners, optics, photodetector, and navigation systems and measure distances using a laser light, which is ultraviolet and visible. They brighten the target with a laser, so are used for high-resolution operations; even a narrow beam resulting in high resolutions. Thus, Lidar cameras are widely used for surveying and mapping. However, they are very costly systems.
  - ***Time-of-flight (ToF):*** These cameras are range imaging systems. They determine distance using the speed of light and functions as follows: Time-of-flight of a light between the subject and camera is measured for each image point. They include an illumination unit, optics, image sensors, driver electronics, and interface. These are similar to Lidar cameras without scanners. ToF cameras work fast; almost 160 images

are captured each second [62]. They are also simple and efficient but costly. In robotics, they are preferred to create a map quickly without obstacles and to spend less power in computation. However, there are issues to be considered such as time measurement accuracy, multiple reflections, background lights, disturbance, and cost.

- ***RGB-Depth***: RGB image and its depth image, in which each pixel correlates with a distance between the image and object, are combined to form a point cloud [63]. With an RGB-D sensor, depth information for each pixel can be captured. Microsoft's Kinect and ASUS's Xtion can be used for that purpose.

- **Hybrid cameras**
  - ***Using multiple sensors:*** Active and passive sensors can be combined to make a more advantageous system for position estimation.

In this study, a stereo camera setup has been used. Stereo images converge to monocular when the features are far away from the camera. Because this study is only concerned with indoor VO, features in the environment are very close to the camera system. As a result, it enables calculation of the depth and scale which cannot be obtained using monocular method without the use of external sensors or initial information about the scene [3]. In other words, stereo cameras are more advantageous for practical purposes. On the other hand, the rationale behind the preference on CCD can be traced on its certain assets. As Litwiller [60] discussed, CCD is more favorable in terms of noise advantage. In addition, the uniformity of response in both imagers for different pixels makes CCD desirable over CMOS which are worse despite a great amount of effort. CCD also enjoys the advantage of lower power dissipation to other circuits on the device. Furthermore, it is easier to modify the binning, speed, range, depth, and other modes for different goals. For an improved pose estimation, inertial sensors are utilized. Since gyroscope data drifts over time, it is combined to an accelerometer for drift compensation. A 6-axis IMU is as a primary sensor for attitude estimation and a secondary sensor for position estimation.

For the formulation of the problem, inputs and outputs are defined as follows:

- Input: For the VO system, the input is grayscale image streams from stereo camera system. Camera intrinsic parameters and baseline distance between stereo camera pair are calculated using stereo calibration algorithms such as [64] and [65].
- Output: 6 axes motion of the platform is required at the end of the VO algorithm. This will be given as 3x3 rotation matrix and 3x1 translation vector with respect to the initial pose of the platform.

Drawing upon Scaramuzza and Fraundorfer's work [3], basic VO process begins with taking images using one or multiple cameras attached to a moving platform. This stage is called the stereo image sequence or stereo image acquisition. Images taken from different cameras, namely left and right cameras, in different time instants, k, can be denoted by $\boldsymbol{I_{l,0:n}} = \{\boldsymbol{I_{l,0}}, \boldsymbol{I_{l,1}}, \dots, \boldsymbol{I_{l,n-1}}, \boldsymbol{I_{l,n}}\}$ and by $\boldsymbol{I_{r,0:n}} = \{\boldsymbol{I_{r,0}}, \boldsymbol{I_{r,1}}, \dots, \boldsymbol{I_{r,n-1}}, \boldsymbol{I_{r,n}}\}$. Transformation of the camera at successive frames, **T**, can be represented by rotation matrix **R**, and translation vector *t* in the following form:

$$\mathbf{T}_{k+1,k} = \left[\begin{array}{c|c} \mathbf{R}_{k+1,k} & \mathbf{t}_{k+1,k} \\ \hline \mathbf{0} & \mathbf{1} \end{array}\right] \tag{3.1}$$

Transformation matrix, ***T***, contains motions in adjacent time instants, while the set of vectors $\boldsymbol{C_{0:n} = \{C_0, C_1, \dots, C_{n-1}, C_n\}}$ denotes the cumulative transformation of the camera pose with respect to the initial camera location as shown in Figure 3.1.

In Figure 3.1, $\boldsymbol{C_0}$ is the initial camera pose and the VO algorithm results with a trajectory estimation with respect to this very point. On the other hand, $\boldsymbol{C_n}$ can be calculated by using the product of all the previous transformations T_0 to T_n. In other words, $\boldsymbol{C_n}$ can be computed as $\boldsymbol{C_n = C_{n-1}T_n = C_0T_1T_2 \dots T_{n-1}T_n}$. Main purpose of VO problem is to calculate the rotation and translation components between successive frames as in *T* and recover estimated trajectory between the initial and final poses as in ***C***.

Following this first phase comes the image processing step which includes preprocessing algorithms on the taken images. Images are first undistorted while stereo images are rectified to have a same image plane which makes stereo feature detection and tracking easier. Salient and robust features are found in stereo image sequence using the using feature detector. Found features are matched and tracked in the ensuing images to estimate or extract the motion. Using an optimization technique, results are refined to minimize the localization error.

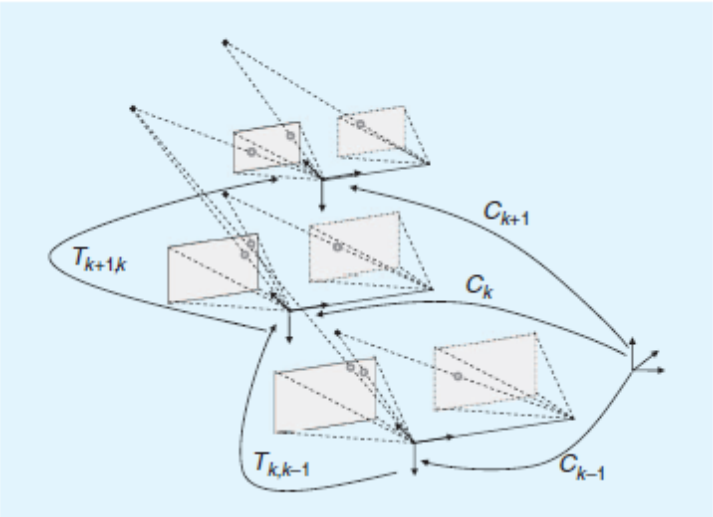

Figure 3.1 VO problem in a nutshell [3]

This pipeline is explained in detailed subsections. First, camera basics are explained in terms of camera model, camera sensor and lens technology, camera parameters, distortion and calibration. Next, feature detectors and descriptors are described. Finally, feature tracking algorithms, motion estimation, outlier rejection and optimization subsections are included.

3.1.1. Camera Basics

Camera model, camera distortion, and camera calibration must be understood before image processing and machine vision algorithms. Details of camera basics can be found in Appendix A. In this study, perspective projection is utilized to project 3D

scene features on an image plane which is more favorable in machine vision algorithms [3].

Camera calibration refers to the intrinsic and extrinsic camera parameters. Intrinsic parameters consist of three different parameters as explained in Appendix A. Focal length representing the distance between the image plane and the lens of the camera, principal point representing the image center point, and the skew coefficient which non-zero if the image axes are not perpendicular as in most CCD cameras. Extrinsic parameters, on the other hand, relate the 3D world point to camera image plane by means of rotation and translation components which is to be found as a result of VO.

CCD cameras provide non-ideal images which must be corrected before machine vision algorithms for better results. Distorted camera images can be formulated using radial and tangential distortion parameters of the camera as showed in Appendix A. Image distortion can be removed using well-known camera calibration algorithms such as [64].

3.1.2. Feature Detection

Computer vision (CV) algorithms are based on local feature detectors and descriptors. Feature detection is a method of image processing in which abstractions of image information are computed so that the possibility of image feature in a given point at each image can be found. As a result, the image domain subsets can be formed. A feature can be considered as a starting point for algorithms in a low-level processing. As such, consequent algorithms can be as good as its feature detection. In other words, repeatability is an essential characteristic in feature detection. Local features are distinct patterns in an image. Features may represent a number of things, but the important thing about them is that they are distinctive from their surroundings. Therefore, it can be inferred that being repeatable, distinctive, and localizable to assign exact points are the three distinguishing merits of a good local feature.

Feature detectors should be distinguished from feature extractors in that the former determines the regions of an image using corners, edges, or blobs whereas the latter computes the descriptors on regions of detected features. To put it in another way,

feature detectors enable selection of the points for processing. However, feature extractors allow the transformation of local pixel neighborhood to a compact vector representation. Descriptors including SIFT, SURF, BRISK, or FREAK (the last two being the binary ones) can be employed.

There are several types of feature detection such as edges, corners, blobs, and ridges. In this study, corner detection has been focused and used, but also blobs are taken into consideration for feature descriptors for reference and comparison.

3.1.2.1. Corner Detection

Corner detection, also referred as interest points, connotes a point-like characteristic in an image with two dimensional structure. Corner detection is used to extract particular features to deduce the properties of an image. A corner, in this sense, refers to the neighboring point where the intersections of two different edges meet. An interest point, similarly, refers to a robustly defined point which could be a corner, line endings, or an isolated point. However, as seen in the literature that these two terms, are interchangeably used.

Some of the corner characteristics are as follows: Corners

- can be defined as junctions of contours or intersection of edges,
- have large changes in intensity in neighborhood points,
- are generally stable and robust over viewpoint and illumination changes,
- are not invariant to image scale.

The history of corner detection lies back to Moravec [66], who viewed corners as low self-similarity points. Whether every pixel views any corners is tested in algorithms. The focus is on the similarity between the nearby patches. Calculating the sum of squared differences between the matching pixels (a low score signifies a high similarity) similarity between the overlapping patches can be tested. However, the problem arises when the edge is out of the neighbors' directions, because it causes to obtain a large sum of squared difference, meaning to select an incorrect edge. Figure

3.2 demonstrates that a corner can be easily recognized by shifting a window over the frame.

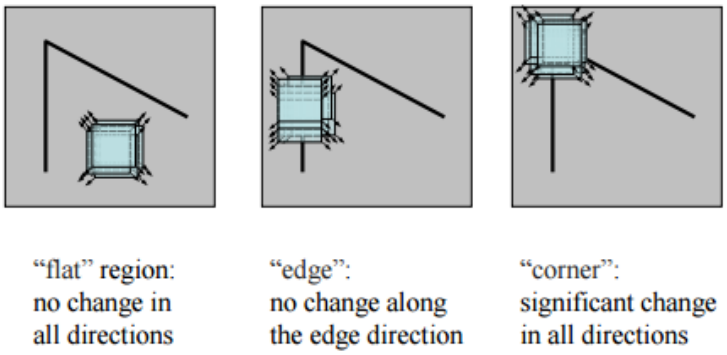

Figure 3.2 Shifting a pre-defined window to determine a corner [17]

If there is a major change in intensity in all directions, the point can be defined as a corner according to Moravec's algorithm. Drawing upon Moravec's work, Harris and Stephens [17] advanced the algorithm using gradients to find the response of the shifted window. In mathematical representation, corner function around an interest point can be expressed as the square of the differences between the current intensity and the shifted intensity of the point.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \tag{3.2}$$

where

$E(u,v)$ is the corner function,
$w(x,y)$ is the window function which can be a rectangular or Gaussian window,
$[u,v]$ is shifting property,
$I(x,y)$ is the current intensity,
$I(x+u,y+v)$ is the shifted intensity.

According to the above expression, if the difference between shifted intensity and the current intensity is close to zero, there is no change in the scene. However, if the difference is large, it means that the score function is large and a corner is located at that point. Writing the intensity change using Taylor Series expansion:

$$\sum w(x,y)[I(x+u,y+v)-I(x,y)]^2$$

$$\approx \sum w(x,y)[I(x,y)+uI_x+vI_y-I(x,y)]^2$$

$$= \sum w(x,y)[u^2{I_x}^2+2uvI_xI_y+v^2{I_y}^2] \quad (3.3)$$

$$= [\,u\ v\,]\left(\sum_{x,y} w(x,y)\begin{bmatrix}{I_x}^2 & I_xI_y\\ I_xI_y & {I_y}^2\end{bmatrix}\right)\begin{bmatrix}u\\ v\end{bmatrix} = [\,u\ v\,]\ \mathbf{M}\ \begin{bmatrix}u\\ v\end{bmatrix}$$

where

I_x is the image derivate in x-direction,

I_y is the image derivative in y-direction,

M is called Harris matrix for corner detection.

If eigenvalues of **M** are described as λ_1 and λ_2, a score function can be defined using the determinant and the trace of the eigenvalues as follows: $det(\mathbf{M}) = |\mathbf{M}| = \lambda_1\lambda_2$ and $tr(\mathbf{M}) = \lambda_1 + \lambda_2$. Thus *cornerness* measure can be made according to below formulation:

$$R = \lambda_1\lambda_2 - \mathrm{k}\,(\lambda_1 + \lambda_2)^2 = |\mathbf{M}| - k(tr(\mathbf{M}))^2 \quad (3.4)$$

where $\mathrm{k} \in [0.04, 0.06]$ is empirically tunable parameter. Since eigenvalue decomposition is computationally expensive because of the computation of square root, it is convenient to use determinant and trace of the function **M** instead of the eigenvalues λ_1 and λ_2. A point can be specified using a score function as follows:

- if λ_1 and λ_2 are small, $|R|$ is small, then the region is flat,
- if λ_1 is larger than λ_2 or vice versa, $R<0$, then the region is an edge,
- if λ_1 and λ_2 are both large and R is also large, then the region is a corner. λ_1 and λ_2 values of each pixel in an image can be sorted to obtain the corner strength and most strong n point can be selected. Exact corner point is the local maxima in that region Figure 3.3 shows how eigenvalues corresponds to a corner, an edge or a region:

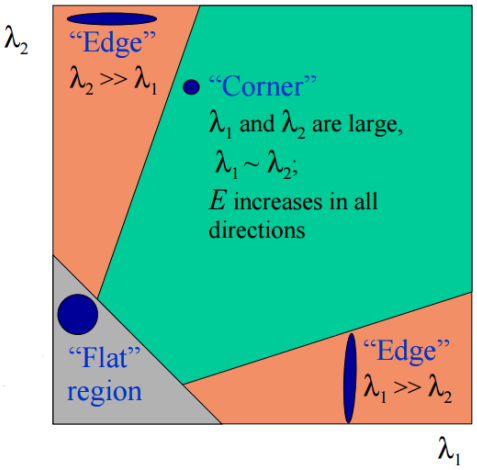

Figure 3.3 Graphical representation of Harris corner detector [17]

Improving upon Harris Corner Detector, Shi and Tomasi [16] suggest using the minimum of the eigenvalues as the score function, thus setting a threshold value for corner quality. Shi-Tomasi corner detector in graph representation can be seen in Figure 3.4.

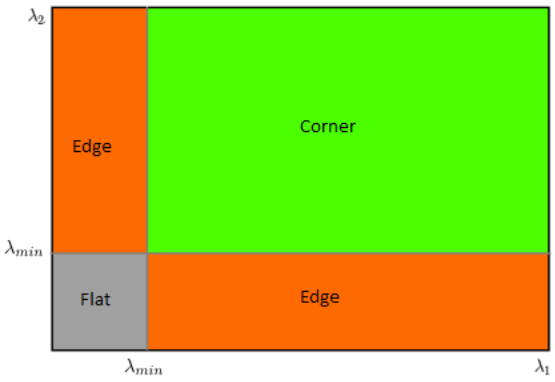

Figure 3.4 Graphical representation of Shi-Tomasi corner detector [16]

There are also FAST [19] and FASTER [67] single scale corner detectors for faster detection of corners. They have their own advantages and disadvantages over Harris and Shi-Tomasi corner detectors. However, in this study Harris and Shi-Tomasi corner detectors are utilized due to their stability in feature tracking.

3.1.2.2. Feature Description

To be able to compare and relate two feature sets of consecutive or stereo images, it is important to attribute a description to each feature point. This can be done using feature vectors and is called feature description. A good feature descriptor has to be robust, and should be able to extract features even under different conditions such as noise, and scale/illumination change. There are many different descriptors in computer vision. Since it is impossible to explain all of them in detail, a comprehensive comparison of famous feature descriptors is given in Table 3.1.

Table 3.1 Comparison of some of the famous feature descriptors

Descriptor	**Invariance**			**Binary**	**Usage**		**Patent**
	Scale	Rotation	Affine		Matching	Class	
SIFT [18]	✓	✓	✓		✓	✓	✓
SURF [23]	✓	✓	✓		✓	✓	✓
BRISK [22]	✓	✓		✓	✓		
FREAK [68]	✓	✓		✓	✓		
ORB [21]	✓	✓		✓	✓	✓	
MSER [69]	✓	✓	✓		✓	✓	

The most well-known feature descriptor is *Scale Invariant Feature Transform (SIFT)* by Lowe [18]. SIFT is not only scale invariant but also rotation, illumination and viewpoint invariant, which makes the algorithm robust and stable. SIFT can be described in six steps as follows:

1. Scale space construction: A filter called scale space is used to take different scales of an image in order to be able to detect different key points. Blob detection is made by finding local maxima using *Laplacian of Gaussian (LoG)* for different σ values.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) = \frac{1}{2\pi\sigma^2} e^{-\left[\frac{(x^2+y^2)}{2\sigma^2}\right]} * I(x, y) \quad (3.5)$$

2. Finding extremes: Since LoG is computationally costly, an approximation, *Different of Gaussians (DoG)*, is used and local extremes on different layers are found. Local extrema points are marked as potential key points for future description process. Figure 3.5 shows the DoG method for SIFT descriptor.
3. Key point localization: Using Taylor series expansion, location points in terms of x, y and σ is found with subpixel accuracy. A 2×2 Hessian window is then used to eliminate edges.

$$D(x) = D + \frac{\partial D^T}{\partial x} x + x^T \frac{\partial^2 D}{2\partial x^2} x$$
$$\rightarrow \text{ Differentiating} \rightarrow \hat{x} = \frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x} \rightarrow (x, y, \sigma) \quad (3.6)$$

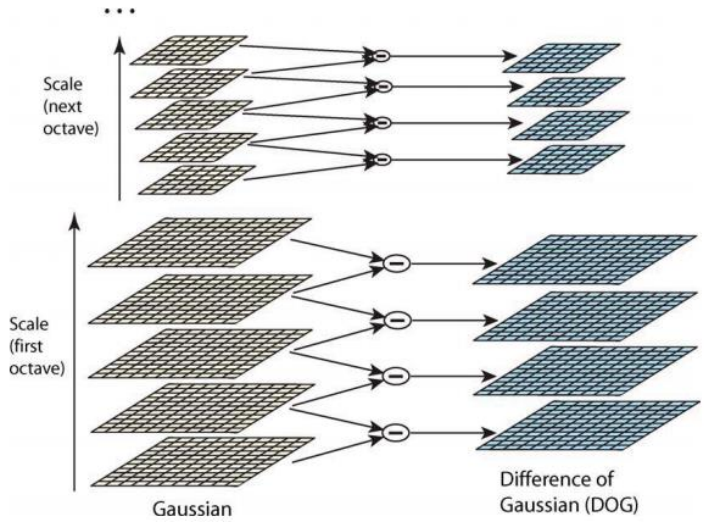

Figure 3.5 Difference of Gaussian method is used in SIFT [18]

4. Orientation estimation: For rotation invariance, each key point is associated with and orientation vector. In the Gaussian smoothed image $L(x, y)$, orientation $\theta(x, y)$ and magnitude $m(x, y)$ is found around each key point and

a histogram for orientation is created. The histogram is weighted by magnitude and $\sigma = 1.5 \times scale$ and peaks larger than 0.8 of key point is used to calculate the orientation.

$$m(x,y) = \sqrt{[L(x+1,y) - L(x-1,y)]^2 + [L(x,y+1) - L(x,y-1)]^2} \quad (3.7a)$$

$$\theta(x,y) = tan^{-1}\left[\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right] \quad (3.7b)$$

5. Key point descriptor: A 4x4 block is created around the key point where each block also divided into a 4x4 sub-block so that histogram of orientation can be represented as key point descriptor vector.
6. Key point matching: Nearest neighbors of key points are matched between two images. If there are two very close matches, second closest is eliminated if their ratio is higher than 0.8.

3.1.3. Feature Tracking

Feature tracking and matching can be done using key point descriptors or optical flow which can be defined as the motion of the features within consecutive frames. Since feature descriptors are explained in section 3.1.2, feature tracking with optical flow will be described here.

One can assume that object in a frame sequence does not change its pixel intensities over time and its connecting pixels also move with it. Mathematically, this can be shown that

$$I(x,y,t) = I(x+dx, y+dy, t+dt) \quad (3.8)$$

Using Taylor series expansion,

$$f_x u + f_y v + f_t = \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t} + f_t = 0 \quad (3.9)$$

Image gradients in x and y directions are indicated by f_x and f_y and time gradient is shown by f_t. There are two unknowns for one equation so the problem cannot be solved without further assumptions.

Kanade-Lucas-Tomasi (KLT) feature tracker [70, 71] is one of the most famous methods that deals with the under-determined optical flow problem. KLT assumes that neighboring pixel around the feature point also travels the same distance, thus having 9 points and 9 equations instead of 1 point and 1 equation to solve for 2 unknowns. Since this is only valid for small motions, tracking is done for different pyramid scales in order to track for large motions. KLT algorithm with a pseudocode is provided in Appendix B. Still steps of the simple KLT algorithm are given in Table 3.2:

Table 3.2 KLT algorithm steps

1) Detect features (Harris corners can be used)
2) For each feature, compute motion between respective frames
3) Link motion vectors in adjacent frames for tracking
4) Since most of the features will be lost after some time, assign new feature points for instance in every 10 frames
5) Track from the beginning.

While KLT uses sparse features for optical flow, it is possible to use all pixels for a dense optical flow as in [72]. Optical flow vectors are calculated using magnitude and direction of pixel intensities. In this study, however, sparse features are tracked via KLT feature tracker.

3.1.4. Motion Estimation

Motion estimation is the core step of VO as the translation and rotation component of the motion is calculated in an additive manner. Assume that a stereo camera system captures images, $\boldsymbol{I}$, at certain time instances, k, and successive images are $\boldsymbol{I_{l,k}}$ and

$\boldsymbol{I_{l,k+1}}$ for the left camera and $\boldsymbol{I_{r,k}}$ and $\boldsymbol{I_{r,k+1}}$ for the right camera. Motion is estimated between these consecutive frames for the whole sequence and then concatenated to build the whole path from the beginning to the end of the motion. There are different methods in the literature [3] to solve the problem as explained in the following sections.

3.1.4.1. Using Essential Matrix and Singular Value Decomposition

Motion between two successive camera frames is related by essential matrix E defined in [13]. Let the locations of the matched feature points in adjacent images are $\boldsymbol{p}$ and $\boldsymbol{p'}$ respectively, essential matrix for a calibrated camera can be expressed as follows:

$$\mathbf{p'E\,p = 0} \tag{3.10}$$

Using Nister's 5-point algorithm [35] which requires a minimum of five feature correspondences or Longuet-Higgins' 8-point algorithm [28] requiring minimum of 8 point correspondences, essential matrix can be calculated efficiently. Essential matrix consists of translation and rotation parameters.

$$\mathbf{E_k \sim \hat{t}_k R_k} \tag{3.11}$$

where $\boldsymbol{\hat{t}_k}$ is the skew symmetric form of the translation vector.

Scale factor is calculated using the same technique among stereo image pairs and then calculated translation components are refined. However, calculation of translation and rotation components from the essential matrix is not straightforward as a decomposition method is needed. Here, *singular value decomposition (SVD)* from [73] is utilized to obtain:

$$\mathbf{R = U(\pm W^T)V^T} \tag{3.12a}$$

$$\mathbf{\hat{t} = U(\pm W^T)SU^T} \tag{3.12b}$$

The detailed explanation of SVD calculation is given in the Appendix A. Step-by-step description of the algorithm is summarized in Table 3.3.

Table 3.3 Motion estimation algorithm using essential matrix and SVD

1) Capture two frames $\boldsymbol{I_k}$ and $\boldsymbol{I_{k+1}}$ 2) Match features between them 3) Compute $\boldsymbol{E}$ 4) Use SVD to get $\boldsymbol{R_k}$ and $\boldsymbol{t_{k+1}}$ 5) Repeat from 1 for stereo image pair and calculate the scale 6) Repeat 1-5 and multiply all transformations

3.1.4.2. Using Perspective from n Points

This approach is generally known as 3D-to-2D image correspondence method for calculating the ego-motion. 3D feature location $\boldsymbol{X_k}$ can be found using stereo data. 2D feature point is p_k, which is the projection of $\boldsymbol{X_k}$ using the transformation $\boldsymbol{T_k}$. The aim of the algorithm is to minimize the reprojection error:

$$arg\min_{\boldsymbol{T_k}} \sum_i \left\| \boldsymbol{p_k^i} - \boldsymbol{T_{k-1} X_{k-1}^i} \right\|_2 = arg\min_{\boldsymbol{k}} \sum_i \left\| \boldsymbol{p_k^i} - \boldsymbol{p_{k-1}^i} \right\|_2 \tag{3.13}$$

Direct Linear Transformation (DLT) is the simplest solution to this problem as suggested in [13]. To apply DLT, 2×9 matrices $\boldsymbol{A_i}$ are computed using a minimum of 6 correspondences:

$$\begin{bmatrix} \boldsymbol{0^T} & \boldsymbol{-z_i' x_i^T} & \boldsymbol{y_i' x_i^T} \\ \boldsymbol{z_i' x_i^T} & \boldsymbol{0^T} & \boldsymbol{-x_i' x_i^T} \end{bmatrix} = \begin{bmatrix} \boldsymbol{p^1} \\ \boldsymbol{p^2} \\ \boldsymbol{p^3} \end{bmatrix} \tag{3.14}$$

where $\boldsymbol{x_i^T} = [x_i \quad y_i \quad z_i \quad 1]$ and $\boldsymbol{p^{iT}}$ is a 4-vector (jth row of P_k). Stacking n 2×9 matrices $\boldsymbol{A_i}$ into a single 2n×9 matrix $\boldsymbol{A}$, SVD can be used to extract translation and rotation components from $\boldsymbol{P} = [\boldsymbol{R} \mid \boldsymbol{t}]$.

There is also a neat minimal case solution to PnP problem using only 3 correspondences which is called P3P [74]. Each step of perspective from n points solution is summarized in Table 3.4.

Table 3.4 Motion estimation using perspective from n points

1) Match stereo image features. Triangulate to get 3D correspondences
2) Track features for several frames reject outliers (chapter 3.1.5)
3) Repeat step 2 for robustness and compute camera pose
4) Triangulate for new matches for both left and right images
5) Repeat from step 2

3.1.4.3. Using 3D-to-3D Point Correspondences

This method is for stereo case only; however, there are issues of being not as accurate as the previous algorithms because of minimizing the feature position error instead of image reprojection error. As the name suggests, 3D point locations from stereo cameras are found in this case and matched with the successive 3D points. $\boldsymbol{X_k^i}$ and $\boldsymbol{X_{k+1}^i}$ representing the 3D point coordinates, algorithm aims to minimize the distance between them as follows:

$$arg \min_{\boldsymbol{T_{k+1}}} \sum_i \left\| \boldsymbol{X_{k+1}^i} - \boldsymbol{T_{k+1} X_k^i} \right\|_2 \tag{3.15}$$

As in the first two algorithms, SVD is used to decompose transformation matrix into rotation and translation components. The algorithm steps can be found in Table 3.5.

Table 3.5 Motion estimation using 3D-to-3D point correspondences

1) Capture 2 stereo images and match features
2) Triangulate matched features to get 3D point set
3) Match features for the left image
4) Compute transformation for the left image
5) Compute transformation for the right image and minimize the error
6) Repeat 1-5 and sum up all transformations

3.1.5. Outlier Rejection

An outlier is a point in a dataset that is distant from the other points, i.e. noisy data. In pose estimation, outlier features show different motion characteristics than the estimated motion. If the feature set contains outliers, the estimates gets more erroneous. Since the odometry is done frame-by-frame, error in the motion estimation accumulate and the drift increases. There are lots of reasons of outliers. Some of them are listed below:

- Image noise
    - Camera movement during shooting
    - Illumination change
    - Motion blur
- Occlusions and environmental changes
- Measurement errors
- Uncertainties in mathematical assumptions, etc.

There are several methods in the literature for rejecting outliers or taking inliers, but the most famous one is using *Random Sample Consensus (RANSAC)* [24]or RANSAC-like outlier rejection schemes. RANSAC assumes that the data points contain inliers with an explainable model and outliers that cannot be explained with that model. It takes a small set of inliers and iteratively finds the correct data set rejecting the outliers. For the VO problem, data points are detected features from successive frames. Procedure is illustrated in Table 3.6.

Since RANSAC offers a non-deterministic probabilistic solution to the outlier rejection problem, it may give different but very similar solutions in different runs especially when the iteration is large. However, when the number of iterations is high, the execution time of the algorithm grows exponentially. Nevertheless, there are algorithms devised to decrease the run time of RANSAC as in Nister's 5-point minimal solver with RANSAC [35]. There are also constrained algorithms to decrease the number of iterations needed to estimate the motion as in [75].

Table 3.6 RANSAC pipeline

1) Take full set of feature matches from successive frames
2) Choose a small set of features are taken from
3) Fit a model to the small feature points set
4) Compute feature distances to the fitted model
5) Find inliers whose distance is smaller than a specified distance
6) Repeat from 2 to cover all feature points
7) Choose the set with most inliers and reject the outliers

3.1.6. Optimization

While RANSAC optimizes the feature matches, this step optimizes the camera pose over last N frames. Since motion estimation is made between adjacent frames, uncertainties accumulate over time. However, it is possible to estimate the motion between the current frame and the n frame before to improve the estimation performance. For this purpose *pose-graph optimization* [76] or *bundle adjustment (BA)* [13] is used. Pose-graph optimization improves the camera parameters only while BA optimizes 3D feature points that are tracked over last n frames. For a 3D feature point $\boldsymbol{X^i}$, the corresponding image feature point p^i, camera pose $\boldsymbol{C_k}$ and the reprojection $\boldsymbol{g}(\boldsymbol{X^i}, \boldsymbol{C_k})$, bundle adjustment tries to minimize this function:

$$arg \min_{\boldsymbol{X^i}, \boldsymbol{C_k}} \sum_{i,k} \left\| \boldsymbol{p_k^i} - \boldsymbol{g}(\boldsymbol{X}^i, \boldsymbol{C_k}) \right\|^2 \tag{3.16}$$

Since the function g is non-linear, Levenberg-Marquardt method is utilized to compute the optimized pose. Pseudocode of sparse Levenberg-Marquardt algorithm is presented in Appendix D.

The only deficiency of the BA optimization is that it is computationally expensive. Complexity of bundle adjustment in general is $O((qm + ln)^3)$ where m is the number of feature points; n is the number of camera poses; q is the number of parameters for

feature points and l is the number of parameters for camera poses. For real time applications, n is being kept small by optimizing only a group of frames. Since offline processing is made in this study, number of frames to be optimized are kept slightly large.

3.2. Inertial Navigation

Inertia is basically a property that keeps the velocity of translation and rotation steady, if not altered by forces or torques. A self-contained technique for navigation, inertial navigation utilizes accelerometer and gyroscope measurements to determine the location and orientation of the objects with respect to their starting point, velocity, and orientation. That is, inertial navigation requires the information on initial point, attitude, and velocity.

Inertial Navigation systems (INS) are autonomous systems that can function underwater or in tunnels, which means that they do not need external aids or visibility since they are only based on Newton's laws of motion. They are also resistant to jamming. INS are quite appropriate for integrated navigation for which IMUs calculate the other variables such as velocity. An INS involves inertial measurement units, instruments support electronics, and navigation computer(s). *Inertial Measurement Units (IMUs)* or *Inertial Reference Units (IRUs)* involve a set of three or more accelerometers and gyroscopes to preserve a constant orientation. But usually, an IMU includes three orthogonal rate-gyroscopes to gauge angular velocity and three orthogonal accelerometers to measure linear acceleration.

Basically, if the acceleration of a vehicle is measured, its velocity and position in regard to time can be found by integration. If navigation in regard to inertial reference frame is desired, then the directions that the accelerometers point should be traced. In addition, IMUs can be utilized to solve the problem of lost visual tracking due to unexpected alterations in illumination, quick motions, or occlusions. In the following section, accelerometers and gyroscopes have been described.

3.2.1. Accelerometers

These sensors measure acceleration, yet not gravitational acceleration. In other words, without gravitational acceleration an accelerometer in freefall does not detect input. There are mechanical, solid-state and MEMs accelerometers. Detailed descriptions of different types of accelerometers can be found in Titterton and Weston's work [77]. In this work, MEMs accelerometers are utilized. Advantages and disadvantages of MEMs accelerometers over traditional systems are presented in Table 3.7.

Table 3.7 Comparison of traditional- and MEMS accelerometers

Traditional Accelerometers	**MEMS Accelerometers**
High performance (Accurate)	More prone to drift and error. (Despite the fact that performance of MEMS accelerometers are increasing rapidly)
Heavy and bulk	Lightweight and small
High power consumption	Low power
Long start-up time	Fast start
Very high cost	Low-cost when mass produced

3.2.1.1. Error Characteristics

Output of an accelerometer without error components can be modeled as:

$$\boldsymbol{a_m} = \frac{1}{m}\left(\boldsymbol{F} - \boldsymbol{F}_g\right) = \boldsymbol{a_B} - \boldsymbol{R_I^B}\begin{bmatrix}0\\0\\g\end{bmatrix} \tag{3.17}$$

where

- $\boldsymbol{a_m}$: measured acceleration
- m: body mass
- $\boldsymbol{F}$: total force acting on the body expressed in the sensor coordinate system
- $\boldsymbol{F}_g$: gravitational force expressed in the sensor coordinate system

$\boldsymbol{a_B}$: Actual acceleration

$\boldsymbol{R_I^B}$: rotation matrix between the inertial frame of reference and sensor body frame

g: gravitational acceleration.

However, different error sources affect the acceleration measurements. Constant bias, white noise, temperature errors, alignment errors and scale factor is explained in detail:

Constant Bias: This error corresponds to the constant offset of the measured value from the correct value. Since position is found by double integration, sensor bias, ϵ, grows with square of the time.

$$\beta_a(T) = \in \cdot \frac{t^2}{2} \tag{3.18}$$

To correct the error, orientation controlled sensor measurements have to be averaged and the bias must be estimated. However, gravity compensation should be done very accurately in order not to include a second bias to the system.

White Noise and Random Walk: White noise in the system results in a huge amount of random walk when double integrated. Let N_i be random variable in the white noise sequence with $E(N_i) = E(N) = 0$ and $(N_i) = \sigma^2$, then the error characteristics can be calculated as

$$\iint_0^t \epsilon(\tau)\, d\tau\, d\tau = \delta t \sum_{i=1}^{n} \delta t \sum_{j=1}^{i} N_j = \delta t^2 \sum_{i=1}^{n} (n - i + 1)\, N_i \tag{3.19}$$

where *n* is number of samples and δt is the sampling period. Expected error in position becomes

$$E\left(\iint_0^t \epsilon(\tau)\, d\tau\, d\tau \right) = \delta t^2 \sum_{i=1}^{n} (n - i + 1)\, E(N_i) = 0 \tag{3.20}$$

Variance is:

$$Var\left(\iint_0^t \epsilon(\tau)\, d\tau\, d\tau\right) = \delta t^4 \sum_{i=1}^{n} (n-i+1)^2\, Var(N_i) \qquad (3.21)$$

$$= \frac{\delta t^4\, n\,(n+1)\;(2n+1)}{6}\, Var(N) \;\cong\; \frac{1}{3}\;\delta t\;\; t^3\;\; \sigma^2$$

It is shown that a second order random walk is formed due to the white noise with a standard deviation growing proportional to $t^{3/2}$.

Temperature Errors: Nonlinear effects of temperature causes quadratic errors [77] on the sensor which can then be compensated using temperature sensor.

Alignment Errors and Scale Factor: Misalignment of the sensor and scale factor introduces a temperature independent bias to the sensor measurements when an acceleration is present. This can be calibrated during the calibration stage of the sensor by putting the sensor body stationary and observing the gravitational acceleration.

3.2.1.2. Velocity and Position Estimation

Velocity and position can be deduced from inertial-frame acceleration by using single and double integration, respectively.

$$v = \int a(t)\, dt \qquad (3.22a)$$

$$x = \iint a(t)\, dt = \int v(t)\, dt \qquad (3.22b)$$

However, data is not gathered continuously but in discrete periods so that velocity and position can only be estimated between small time intervals, *T*, using

$$v[i+1] = v[i] + T\, a[i] \qquad (3.23a)$$

$$x[i+1] = x[i] + T\, v[i] + \; T^2\, a[i] \qquad (3.23b)$$

Block diagram of velocity and position estimation is shown in Figure 3.6.

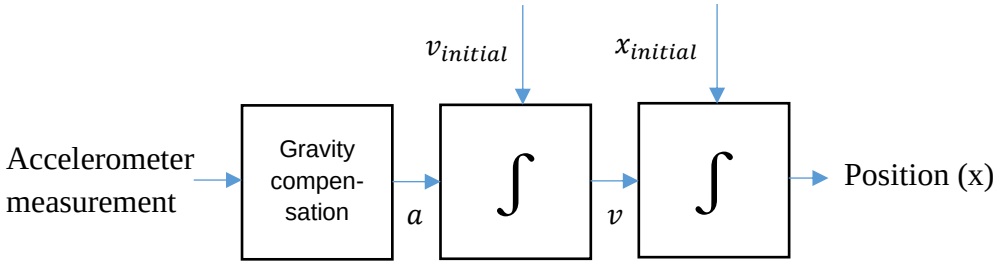

Figure 3.6 Velocity and position estimation block diagram

3.2.2. Gyroscopes

These sensors measure rotation. While rate-gyroscope measures the rate of rotation, whole-angle (integrating) gyroscopes measure the angle of rotation. In a conventional gyroscope, there is a spinning wheel on two gimbals enabling it to turn in three axes. There are mechanical, optical and MEMS gyroscopes. Difference between MEMS gyroscopes and traditional gyroscopes are very similar with Table 3.7 as in accelerometers. Error sources are very similar to those in MEMS accelerometers. While the effect of the double integration is large in position estimation of accelerometer results, single integration yields less error in orientation estimation of gyroscopes. Description and the effect of each error source is shown in Table 3.8:

Table 3.8 Error sources of MEMS gyroscopes

Error Type	Error Description	Effect in orientation
Bias	Constant bias ϵ	Proportional with the time
White Noise	Standard deviation σ	Proportional with square root of the time
Temperature	Bias due to temp. change	Residual bias is proportional with time
Alignment, Scale	Non-random calibration errors	Proportional with the angular velocity rate and the time

3.2.2.1. Orientation Estimation

As in accelerometer, angular velocity signals are measured in fixed time periods in a gyroscope. Several methods and notations are used to calculate the orientation from angular velocity like Euler angles and quaternions. In this study, direction cosines representation is used to calculate the rotation matrix **C** with respect to the initial attitude.

$$\mathbf{v_{global}} = \mathbf{C}\,\mathbf{v_{body}} \tag{3.24}$$

where $\boldsymbol{v_{global}}$ is vector in global frame while $\boldsymbol{v_{body}}$ denotes vector in body frame. Rate of change of **C** can be calculated as

$$\dot{\mathbf{C}}(\mathrm{t}) = \lim_{T\to 0}\left(\frac{\mathbf{C}(\mathrm{t}+\mathrm{T}) - \mathbf{C}(\mathrm{t})}{\mathrm{T}}\right) \tag{3.25}$$

where $\mathbf{C}(\mathrm{t}+\mathrm{T})$ can be written using the rotation matrix $\mathbf{A}(\mathrm{t})$:

$$\mathbf{C}(\mathrm{t}+\mathrm{T}) = \mathbf{C}(\mathrm{t})\,\mathbf{A}(\mathrm{t}) = \mathbf{I} + \boldsymbol{\delta\psi} \tag{3.26}$$

In Eqn. (3.26),

$$\boldsymbol{\delta\psi} = \begin{pmatrix} 0 & -\delta\psi & \delta\theta \\ \delta\psi & 0 & -\delta\varphi \\ -\delta\theta & \delta\varphi & 0 \end{pmatrix} \tag{3.27}$$

Consequently,

$$\begin{aligned} \dot{\mathbf{C}}(\mathrm{t}) &= \lim_{T\to 0}\left(\frac{\mathbf{C}(\mathrm{t}+\mathrm{T}) - \mathbf{C}(\mathrm{t})}{\mathrm{T}}\right) \\ &= \lim_{T\to 0}\left(\frac{\mathbf{C}(\mathrm{t})\mathbf{A}(\mathrm{t}) - \mathbf{C}(\mathrm{t})}{\mathrm{T}}\right) \\ &= \lim_{T\to 0}\left(\frac{\mathbf{C}(\mathrm{t})(\mathbf{I} + \boldsymbol{\delta\psi}) - \mathbf{C}(\mathrm{t})}{\mathrm{T}}\right) = \mathbf{C}(\mathrm{t}) \lim_{T\to 0} = \boldsymbol{\Omega}(\mathrm{t}) \end{aligned} \tag{3.28}$$

where $\boldsymbol{\Omega}(t)$ is the matrix representation of angular velocity vector $\boldsymbol{\omega}_b(t)$ as

$$\boldsymbol{\Omega}(\mathrm{t}) = \begin{bmatrix} 0 & -\omega_{bz}(t) & \omega_{by}(t) \\ \omega_{bz}(t) & 0 & -\omega_{bx}(t) \\ -\omega_{by}(t) & \omega_{bx}(t) & 0 \end{bmatrix} \tag{3.29}$$

Finally, orientation with respect to the initial attitude is found as:

$$\dot{\mathbf{C}}(\mathrm{t}) = \mathbf{C}(\mathrm{t})\boldsymbol{\Omega}(\mathrm{t}) \tag{3.30a}$$

$$\mathbf{C}(\mathrm{t}) = \mathbf{C}(0)\, e^{\int_0^t \boldsymbol{\Omega}(t)} \tag{3.30b}$$

3.3. Sensor Fusion

In this section, data fusion is explained. First, fusion of accelerometer and gyroscope readings is done using so-called complementary filter to obtain orientation and position estimations. Second, inertial and visual data is fused using Extended Kalman Filter which is a non-linear version of standard Kalman Filtering method.

3.3.1. Fusion of Accelerometer and Gyroscope Data

As mentioned in section 3.2, outputs of MEMS accelerometer and gyroscope are very noisy due to white noise and sensor bias. However, their error characteristics are slightly different from each other in different conditions. On one hand, accelerometers are highly sensitive to dynamic conditions while providing decent results in slight movements. On the other hand, gyroscopes have poor static characteristics, but do better in dynamic conditions. Thus a filter mechanism should be applied to each sensor output in order to utilize the best output overcoming the shortcomings of each sensor. To fuse the noisy inertial sensor measurements, complementary filter can be utilized. Numerically integrated gyroscope data is high-pass filtered and fused with low-pass filtered accelerometer data in order to obtain the orientation. Low-pass filtered acceleration only allow long-term changes while filtering the short-term fluctuations. Since gyroscope data drifts over time, high-pass filter only trust the sensor in the short term while filtering the steady signals. Figure 3.7 presents basic complementary filter.

Mahony's explicit complementary filter [78], as illustrated in Figure 3.8, presents an enhanced estimation of the orientation. quaternion notation (see Appendix E) is used for orientation estimation to eliminate the singularity of Euler angles.

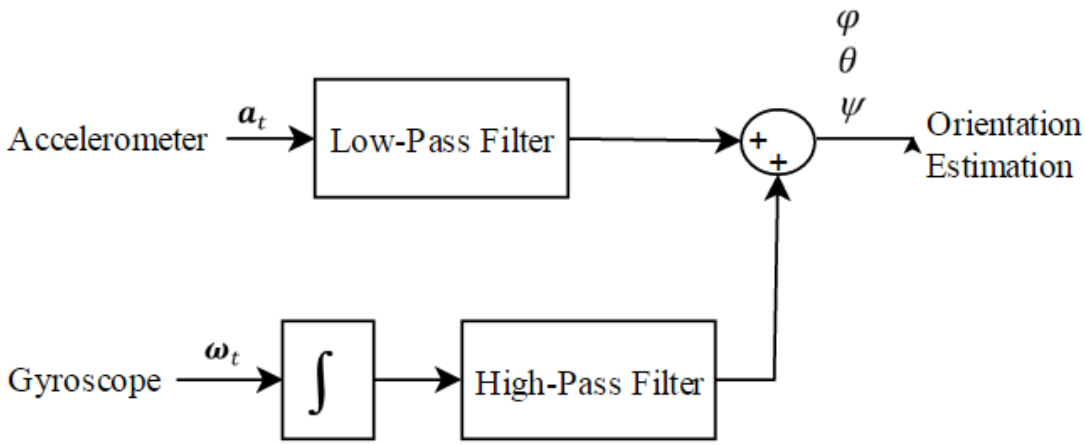

Figure 3.7 Basic complementary filter for orientation estimation

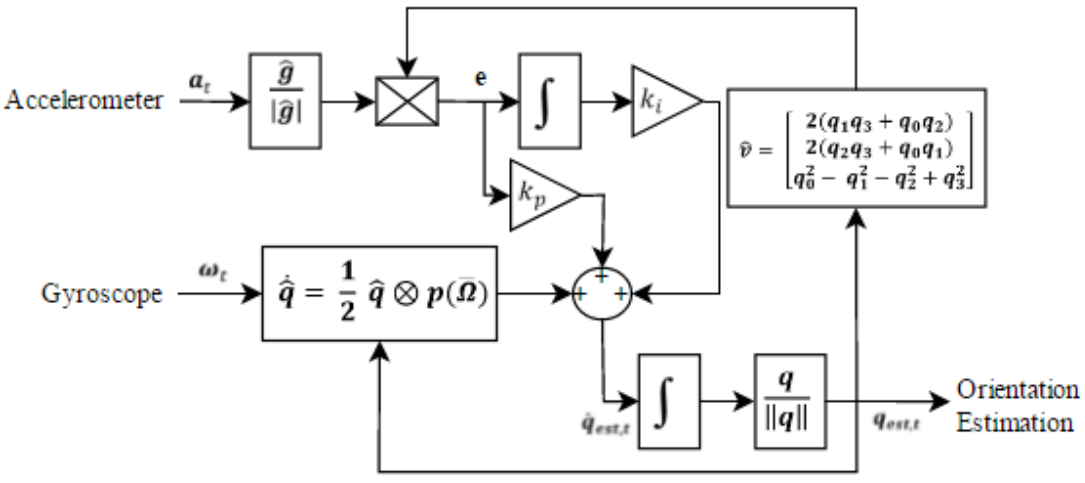

Figure 3.8 Explicit complementary filter for orientation estimation [78]

Algorithm steps for the complementary filter is described as follows:

1. First, data is gathered from the inertial sensors. Initial pose is assumed to be zero meaning that there is no rotation. Gain parameters and sampling rate is defined.

2. Direction of the gravity vector is estimated using the output quaternion:

$$\hat{\boldsymbol{v}} = \begin{bmatrix} 2(q_1 q_3 + q_0 q_2) \\ 2(q_2 q_3 + q_0 q_1) \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.31}$$

3. Acceleration data is normalized. Cross-product of normalized acceleration and the estimated gravity direction is combined to form the error function.

$$\overline{\boldsymbol{v}} = \frac{\hat{\boldsymbol{g}}}{|\hat{\boldsymbol{g}}|} \quad \boldsymbol{e} = \overline{\boldsymbol{v}} \times \hat{\boldsymbol{v}} \tag{3.32}$$

4. Error in estimated direction of gravity, **e**, is fused to gyroscope data using adjustable proportional and integral gain values k_p and k_i, respectively.

$$\overline{\boldsymbol{\Omega}} = \boldsymbol{\Omega} + \mathbf{k_p e} + \mathbf{k_i} \int \mathbf{e}\, \mathbf{dt} \tag{3.33}$$

5. Formula of quaternion differentiation gives:

$$\dot{\hat{\mathbf{q}}} = \frac{\mathbf{1}}{\mathbf{2}} \hat{\mathbf{q}} \otimes \mathbf{p}(\overline{\boldsymbol{\Omega}}) \tag{3.34}$$

6. Taking the integral of $\dot{\hat{q}}$ and normalizing, orientation estimation for one iteration is completed. For a long run, these steps are followed from 1 to 6 and the orientation estimation is updated in each step.

Position is estimated by taking the double integral of the acceleration and filtering in every iteration. Since there is random walk in accelerometer readings, the instances when the system has zero velocity are estimated. Thus, the system resets its velocity and drift is reduced. Zero velocity can be easily observed from IMU readings. When

the system is not in motion, velocity is zero and the angular rate is zero. However, accelerometer also reads zero when the velocity is not zero but a constant value. However, it is assumed that the base system that holds the inertial sensors cannot make odometry without slight vibrations, thus a high-pass filter can solve this problem. Figure 3.9 shows the complete block diagram for IMU pose and orientation estimation.

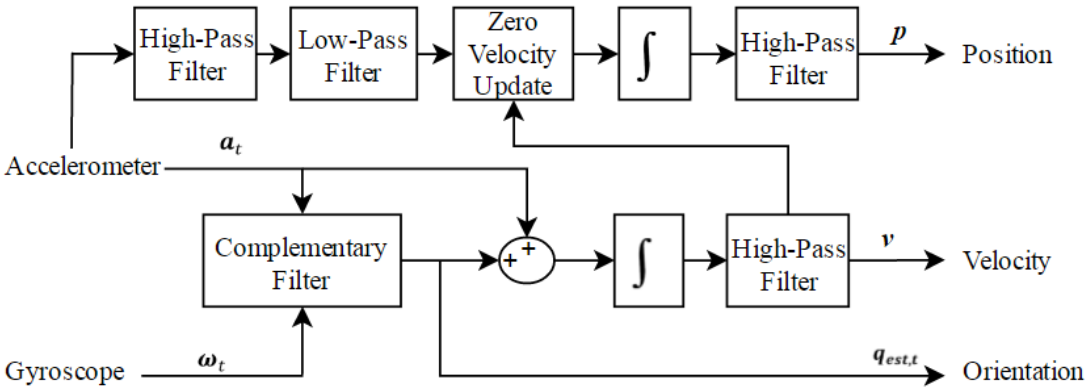

Figure 3.9 IMU pose and orientation estimation block diagram

Algorithm steps are described as follows:

1. Velocity is found by taking the integral of the cross product of the accelerometer readings and complementary filter output quaternion.
2. To filter the zero velocity, first a high-pass filter is applied for smoothing. Second, a low-pass filter is applied to filter zero velocity. If filtered acceleration is below a predetermined threshold value, it is marked as stationary.
3. Filtered velocity values are integrated and high-pass filtered to obtain the position estimation.

3.3.2. Fusion of Visual and Inertial Data

Visual and inertial data is fused using Kalman filter which is a probabilistic estimation method that minimizes the estimation error using state representation and transition. For an optimal estimate, Kalman filter combines two estimates of position or velocity and creates an enhanced estimate with minimal uncertainty. The system uses state-space representation, and probability density function with Gaussian distribution is utilized for the stochastic estimation algorithm. However, linear state propagation and correction estimates are a limitation for visual and inertial navigation which can be considered as a nonlinear problem. Thus, an EKF is utilized for data fusion. For the formulation of the EKF, [79] is followed.

For a general definition, EKF takes two different predictions for the state vector as inputs. In our case, they are VO and inertial navigation outputs. Then, EKF consists of two stages. First a prediction is made using inertial measurements and then prediction is corrected (updated) using VO results. Therefore, EKF is explained in three different sections: state representation, prediction stage and update stage.

3.3.2.1. State Representation

The estimates from two different sensors form the state transition vector. It consists of 3-DOF visual input representing the camera position in world frame, $\boldsymbol{\tau_n}$, 1-DOF scale factor, $\boldsymbol{\lambda}$, and 4-DOF quaternion of the camera orientation in world frame, $\boldsymbol{q_n}$. Equations are as follows:

$$\boldsymbol{\mu_n} = [\boldsymbol{\tau_{x,k}} \quad \boldsymbol{\tau_{y,k}} \quad \boldsymbol{\tau_{z,k}} \quad \boldsymbol{\lambda} \quad \boldsymbol{q_{s,k}} \quad \boldsymbol{q_{a,k}} \quad \boldsymbol{q_{b,k}} \quad \boldsymbol{q_{c,k}}]^{\boldsymbol{T}} \tag{3.35}$$

In extended Kalman filter, state transition and update equations are non-linear. Thus, an approximation of the probability density function is used instead of a Gaussian distribution. Taylor expansion is utilized for the linearization of Gaussian distribution.

3.3.2.2. Prediction Stage

Prediction stage generally consists of a motion model in navigation systems. Given the previous state, expected motion is represented in motion model. This can be done using velocity motion model in which the velocity is assumed to be unchanged. However, this method cannot estimate the state vector truly when there are sudden changes in velocity. Thus, odometry motion model [79] is used in this study and prediction is done using inertial sensor measurements. State equation can be written as follows:

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{k}} = \boldsymbol{g}(\boldsymbol{\mu}_{\boldsymbol{k-1}}, \boldsymbol{v}_{\boldsymbol{k}}) \tag{3.36}$$

where $\hat{\hat{\mu}}_k$ consists of

$$\hat{\boldsymbol{\tau}}_{\boldsymbol{k}} = \tilde{\boldsymbol{q}}_{\boldsymbol{v,k}} \tilde{\boldsymbol{\tau}}_{\boldsymbol{k-1}} \tilde{\boldsymbol{q}}^{*}{}_{\boldsymbol{v,k}} + \boldsymbol{\lambda}_{\boldsymbol{k-1}} \hat{\boldsymbol{\tau}}_{\boldsymbol{k}} \tag{3.37a}$$

$$\hat{\boldsymbol{\lambda}}_{\boldsymbol{k}} = \boldsymbol{\lambda}_{\boldsymbol{k-1}} \tag{3.37b}$$

$$\hat{\boldsymbol{q}}_{\boldsymbol{k}} = \tilde{\boldsymbol{q}}_{\boldsymbol{v,k}} \tilde{\boldsymbol{q}}_{\boldsymbol{k-1}} \tag{3.37c}$$

In the equations, $\hat{\boldsymbol{\mu}}_{\boldsymbol{k}}$ is the predicted state vector, $\boldsymbol{v}_{\boldsymbol{k}}$ is the inertial measurement vector, $\boldsymbol{\mu}_{\boldsymbol{k-1}}$ is the previous state vector. To linearize the non-linear function $\boldsymbol{g}(\hat{\boldsymbol{\mu}}_{\boldsymbol{k-1}}, \boldsymbol{v}_{\boldsymbol{k}})$, Taylor expansion around the input and the current state is used. Necessary Jacobians are presented below:

$$\mathbf{J_g}(\mu_{\mathbf{k-1}}) = \begin{bmatrix} \mathbf{R}_{\tilde{\mathbf{q}}_\mathbf{v}} & \boldsymbol{\tau}_\mathbf{v} & \mathbf{0}_{3\times4} \\ \mathbf{I}_{1\times3} & \mathbf{1}_{4\times1} & \mathbf{1}_{1\times4} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times1} & \mathbf{J}_{\mathbf{q_v q_{k-1}}}(\mathbf{q_{k-1}}) \end{bmatrix} \tag{3.38a}$$

$$\mathbf{J_g}(\mathbf{v_k}) = \begin{bmatrix} \mathbf{v}.\mathbf{I}_{3\times3} & \mathbf{J}_{\tilde{\mathbf{q}}_\mathbf{v}\tilde{\boldsymbol{\tau}}_{\mathbf{k-1}}\tilde{\mathbf{q}}^{*}{}_\mathbf{v}}(\tilde{\mathbf{q}}_\mathbf{v}) \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times4} \\ \mathbf{0}_{4\times3} & \mathbf{J}_{\mathbf{q_v q_{k-1}}}(\mathbf{q_v}) \end{bmatrix} \tag{3.38b}$$

$$\mathbf{J}_{\tilde{\mathbf{q}}_\mathbf{v}\tilde{\mathbf{t}}_{\mathbf{k-1}}\tilde{\mathbf{q}}_\mathbf{v}}(\tilde{\mathbf{q}}_\mathbf{v}) = [\mathbf{R}_{\mathbf{s},\tilde{\mathbf{q}}}\tilde{\mathbf{t}}_{\mathbf{k-1}} \quad \mathbf{R}_{\mathbf{a},\tilde{\mathbf{q}}}\tilde{\mathbf{t}}_{\mathbf{k-1}} \quad \mathbf{R}_{\mathbf{b},\tilde{\mathbf{q}}}\tilde{\mathbf{t}}_{\mathbf{k-1}} \quad \mathbf{R}_{\mathbf{c},\tilde{\mathbf{q}}}\tilde{\mathbf{t}}_{\mathbf{k-1}}] \tag{3.38c}$$

$$\mathbf{J}_{\mathbf{q_v q_{k-1}}}(\mathbf{q_{k-1}}) = \begin{bmatrix} q_{v,s} & -q_{v,a} & -q_{v,b} & -q_{v,c} \\ q_{v,a} & q_{v,s} & -q_{v,c} & q_{v,b} \\ q_{v,b} & q_{v,c} & q_{v,s} & -q_{v,a} \\ q_{v,c} & -q_{v,b} & q_{v,a} & q_{v,s} \end{bmatrix} \tag{3.38d}$$

$$\mathbf{J}_{\mathbf{q_v q_{k-1}}}(\mathbf{q_v}) = \begin{bmatrix} q_{k-1,s} & -q_{k-1,a} & -q_{k-1,b} & -q_{k-1,c} \\ q_{k-1,a} & q_{k-1,s} & -q_{k-1,c} & q_{k-1,b} \\ q_{k-1,b} & q_{k-1,c} & q_{k-1,s} & -q_{k-1,a} \\ q_{k-1,c} & -q_{k-1,b} & q_{k-1,a} & q_{k-1,s} \end{bmatrix} \tag{3.38e}$$

where $\boldsymbol{R}_{\tilde{q}_v}$ is the rotation matrix.

It is important to present the error covariance matrix of the current state which will reveal how much the state estimation is trustable. Smaller error covariance matrix means that the current estimated state is more trustable. The equation for covariance with respect to the previous estimation and the error is as follows:

$$\widehat{\mathbf{M}}_\mathbf{k} = \mathbf{J_g}(\widehat{\boldsymbol{\mu}}_{\mathbf{k-1}})\mathbf{M_{k-1}}\,\mathbf{J_g^T}(\widehat{\boldsymbol{\mu}}_{\mathbf{k-1}}) + \mathbf{J_g}(\mathbf{v_k})\,\boldsymbol{\gamma}_\mathbf{k}\,\mathbf{J_g^T}(\mathbf{v_k}) \tag{3.39}$$

where $\boldsymbol{\gamma_k}$ is inertial measurement noise vector.

3.3.2.3. Update Stage

Unlike the prediction stage, this stage is linear. As the name suggests, this stage updates the prediction made in the previous step. First, difference between the measurement and the priori state estimation, which is called innovation, is found as

$$\tilde{\mathbf{y}}_\mathbf{k} = \boldsymbol{\varphi}_\mathbf{k} - \boldsymbol{\beta}\,\boldsymbol{\mu}_\mathbf{k} \tag{3.40}$$

where $\boldsymbol{\varphi_k}$ is a 7×8 matrix and $\boldsymbol{\beta}$ is the measurement model vector defined as

$$\boldsymbol{\varphi_k} = \boldsymbol{\beta}\,\boldsymbol{\mu_k} = \begin{bmatrix} \mathbf{I_{1\times3}} & \mathbf{0_{3\times1}} & \mathbf{0_{3\times4}} \\ \mathbf{0_{4\times3}} & \mathbf{0_{4\times1}} & \mathbf{I_{4\times4}} \end{bmatrix} \boldsymbol{\mu_k} \tag{3.41}$$

Innovation covariance is calculated as

$$\mathbf{S_k} = \boldsymbol{\beta}\,\widehat{\mathbf{M}}_\mathbf{k}\boldsymbol{\beta}^\mathbf{T} + \mathbf{R} \tag{3.42}$$

Then Kalman gain can be calculated. Kalman gain states the trust ratio to the innovation.

$$\mathbf{K_k} = \mathbf{M_k}\,\boldsymbol{\beta}^\mathbf{T}\mathbf{S}_\mathbf{k}^{-\mathbf{1}} = \mathbf{M_k}\,\boldsymbol{\beta}^\mathbf{T}(\boldsymbol{\beta}\,\widehat{\mathbf{M}}_\mathbf{k}\boldsymbol{\beta}^\mathbf{T} + \mathbf{R})^{-\mathbf{1}} \tag{3.43}$$

If innovation is not trustable, innovation covariance $\boldsymbol{S}$ will be high and if the estimation is trustable, error covariance matrix $\boldsymbol{M_k}$ will be small, thus the Kalman gain. On the contrary, if $\mathbf{S}$ is high and $\boldsymbol{M_k}$ is low, then Kalman gain will be large stating that innovation is trustable. Now, corrected estimate can be written as:

$$\hat{\boldsymbol{\mu}}_\mathbf{k} = \hat{\boldsymbol{\mu}}_\mathbf{k} + \mathbf{K_k}\tilde{\mathbf{y}}_\mathbf{k} = \hat{\boldsymbol{\mu}}_\mathbf{k} + \mathbf{K_k}(\boldsymbol{\varphi_k} - \boldsymbol{\beta}\,\hat{\boldsymbol{\mu}}_\mathbf{k}) \tag{3.44}$$

Finally, corrected estimation error covariance matrix is updated as:

$$\mathbf{M_k} = (\mathbf{I_{8\times8}} - \mathbf{K_k}\,\boldsymbol{\beta})\,\widehat{\mathbf{M}}_\mathbf{k} \tag{3.45}$$

3.4. Proposed Methodology

Theory of VO, inertial navigation and fusion of inertial and visual data is given in previous sections. In this section, a methodology is proposed for the implementation of stereo visual inertial odometry. First, VO implementation details are presented. Second, IMU attitude estimation is given. Third, fusion of IMU data and visual data is implemented.

3.4.1. Implementation of Stereo Visual Odometry

Stereo VO is implemented mainly from the works of Howard [43] and Nister [35]. For the implementation, MATLAB's computer vision and optimization toolboxes are utilized. Inputs and outputs of each step of the stereo VO implementation is described as:

Input: Stereo image pairs captured at times k and k+1 as defined in Chapter 3.1. Successive stereo image pairs can be denoted as $\boldsymbol{I} = \{\boldsymbol{I_{l,k}}, \boldsymbol{I_{l,k+1}}, \boldsymbol{I_{r,k}}, \boldsymbol{I_{r,k+1}}\}$

Output: Rotation matrix, **R**, and translation vector, **t**, as contained in **T** as outlined in Equation (3.1). Homogenous transformation matrix, **T**, contains estimated motions in adjacent time instants. Homogeneous transformation matrices are multiplied to obtain a pose estimation between initial and final frames. Implementation of stereo VO is given in Figure 3.10.

Used datasets provide visual data in .png format which can be read easily with MATLAB. There is also a file for camera parameters (i.e. distortion, intrinsic and extrinsic parameters) which is required for image pre-processing algorithms. Step by step explanation of the implementation is:

1. Input images at time k and k+1 are taken from the dataset
$$\boldsymbol{I} = \{\boldsymbol{I}_{l,k}, \boldsymbol{I}_{l,k+1}, \boldsymbol{I}_{r,k}, \boldsymbol{I}_{r,k+1}\}$$
2. To apply machine vision algorithms to the input images, a preprocessing step is implemented using image processing techniques. First of all, camera matrices for intrinsic and extrinsic parameters are constructed. Using the given lens distortion parameters, tangential and radial distortion effects of the CCD camera is compensated. Stereo image pairs are rectified so that epipolar lines are collinear in each image. This is an important step for further calculations because disparity map, $\mathbf{D}_k$, is computed using horizontal matching blocks. Epipolar geometry and stereo rectification is explained in Appendix A and Appendix C, respectively.
3. Using the left image, $\boldsymbol{I}_{l,k}$, Harris features are found and each feature point is given a score for corner strength. Since homogeneity in features location is

important for VO algorithms, Harris feature detector is improved. Images are divided to 6x8 small images and 10 strongest Harris features are detected in each small image. Thus, features are homogenously distributed over the whole frame.

4. KLT tracker is used to track a feature point in $\boldsymbol{I}_{l,k+1}$. Feature points cannot be tracked in a large tracking framework, because features can be lost due to illumination change, out of plane rotation, or articulated motion of the platform. If the number of tracked features is below a threshold, a new set of

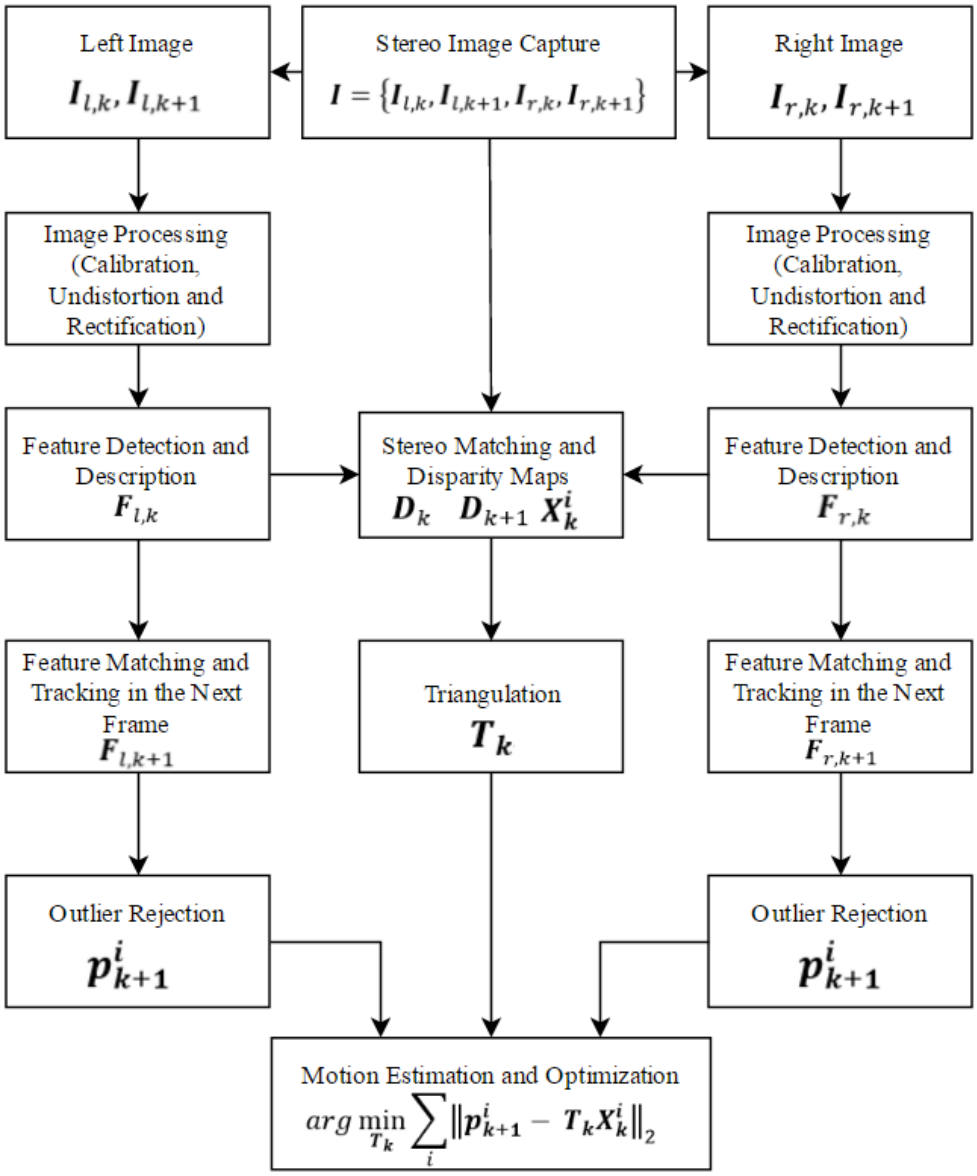

Figure 3.10 Stereo VO block diagram

features are detected and the tracker object is established again. KLT details can be found in Appendix B. In the implementation, a pyramid size of 3 is used to track the features in different levels to account for the scale change. Validity of the tracking is ensured using a backtracking algorithm, so that invalidly tracked points are not used for further calculations. KLT implementation uses a 31x31 block size for the tracking window and number of iterations for the new location of the feature point is limited to 30 for computational reasons.

5. Using stereo image pairs, $\boldsymbol{I}_{l,k}$ and $\boldsymbol{I}_{r,k}$, disparity map, $\boldsymbol{D}_k$, is calculated. Disparity map holds the horizontal pixel location change of a 3D point in stereo image pairs. Since the images are rectified, a feature location in the left image $(x_{l,k}, y_{l,k})$ appears in the right image as $(x_{r,k} + d, y_{r,k})$. Note that location of the feature point does not change in vertical direction because of the rectification. Dense disparity calculation [48] is implemented which minimizes the sum of the absolute differences within the search window size of 15 in our application.
6. Similarly, using $\boldsymbol{I}_{l,k+1}$ and $\boldsymbol{I}_{r,k+1}$, disparity map, $\boldsymbol{D}_{k+1}$, is computed.
7. Using disparity maps, $\boldsymbol{D}_k$ and $\boldsymbol{D}_{k+1}$, and their projections on the image plane feature sets, $\boldsymbol{F}_k$ and $\boldsymbol{F}_{k+1}$, are constructed. Reprojection matrix for triangulation is denoted as **Q**, and the resulting 3D feature set is denoted as $\boldsymbol{W}$.
8. Assuming 3D features at time k and k+1 should have the same motion, outliers can be rejected. In other words, the distance between a feature in $\boldsymbol{W}_k$ and the corresponding feature in $\boldsymbol{W}_{k+1}$ must not change. Furthermore, if a moving object is present in the images, its motion differs from the remaining part of the scene and rejected. A consistency matrix, $\boldsymbol{M}$, is constructed as suggested in [43] to detect the inliers and reject the outliers.
9. Image reprojection error between 3D features and the 2D image correspondences is minimized in the last step to obtain the motion estimation. Sparse Levenberg-Marquardt optimization is used for this purpose. Estimated motion is divided into rotation matrix and translation vector in order to extract the motion in the camera frame.

3.4.2. Implementation of Inertial Pose Estimation

Since the movement of the MAV, described in Chapter 4, is very shaky, orientation estimates from VO is not reliable. Thus, inertial sensors are used for estimating the orientation of the vehicle. Madgwick [80] developed a gradient descent complimentary filter improving the work by Mahony described in 3.3.1. The rate of change of angular rate in quaternion representation is same with Equation (3.34). Orientation at time t, $\boldsymbol{q_{est,t}}$, is computed by numerically integrating estimated orientation rate $\dot{\boldsymbol{q}}_{\boldsymbol{est,t}}$:

$$\boldsymbol{q_{est,t}} = \boldsymbol{q_{est,t-1}} + \dot{\boldsymbol{q}}_{\boldsymbol{est,t}}\,\Delta t \tag{3.46a}$$

where,

$$\dot{\boldsymbol{q}}_{\boldsymbol{est,t}} = \dot{\boldsymbol{q}}_{\boldsymbol{\omega,t}} - \beta\frac{\boldsymbol{\nabla f}}{\|\boldsymbol{\nabla f}\|} \tag{3.46b}$$

$$\boldsymbol{\nabla f} = \boldsymbol{J_g^T(q)f_g(q,a)} \tag{3.46c}$$

$$\boldsymbol{f_g(q,a)} = 2\begin{bmatrix}(q_2q_4 - q_1q_3)\\(q_1q_2 + q_3q_4)\\(0.5 - q_2^2 - q_3^2)\end{bmatrix} - \boldsymbol{a} \tag{3.46d}$$

$$\boldsymbol{J_g(q)} = \begin{bmatrix}-2q_3 & 2q_4 & -2q_1 & 2q_2\\ 2q_2 & 2q_1 & 2q_4 & 2q_3\\ 0 & -4q_2 & -4q_3 & 0\end{bmatrix} \tag{3.46e}$$

In the formulation, $\dot{\boldsymbol{q}}_{\boldsymbol{\omega,t}}$ is the rate of change of the orientation from gyroscope readings, β is the orientation measurement error, $\frac{\boldsymbol{\nabla f}}{\|\boldsymbol{\nabla f}\|}$ is the direction of the estimated error, $\boldsymbol{\nabla f}$ is the gradient function, $\boldsymbol{f_g(q,a)}$ is the objective function of the gradient and $\boldsymbol{J_g(q)}$ is its Jacobian. Detailed explanation of the terms is given in the step by step algorithm description after the block diagram of the improved version of Madgwick's complementary filter [80] used in the implementation in Figure 3.11.

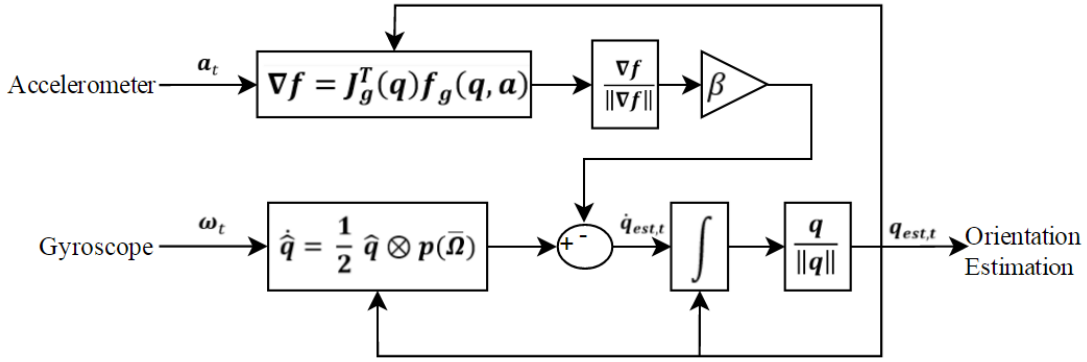

Figure 3.11 Block diagram of implemented complementary filter [80]

1) Initialization: Orientation, sampling time and divergence rate, β, are initialized. Sampling time is 0.005 seconds and β is initialized as 0.1. β is an adjustable parameter and can take a value between 0 and 1. Values larger than 1 can be given in initialization for fast convergence. Orientation initialization is done using quaternions as:

$$\boldsymbol{q} = [1\ 0\ 0\ 0]$$

2) Normalization: Three axes accelerometer data is normalized.
3) Gradient descent algorithm: Gradient descent is calculated as in Equation (3.48) to compensate for the gravity vector with an optimization method.
4) Error direction: The direction of the error is calculated as $\frac{\nabla f}{\|\nabla f\|}$.
5) Data fusion: Using Equation (3.47), rate quaternion correction is performed. In the equation, $\dot{\boldsymbol{q}}_{\boldsymbol{\omega,t}}$ depends on gyroscope data, while $\frac{\nabla f}{\|\nabla f\|}$ depends on accelerometer data.
6) Integration: Final orientation is estimated using numerical integration as in Equation (3.46). Here, $\boldsymbol{q}_{\boldsymbol{est,t-1}}$ is the normalized quaternion estimated at time t-1, $\dot{\boldsymbol{q}}_{\boldsymbol{est,t}}$ is from step (5), and Δt is the sampling time which is 0.005 seconds.
7) Normalization: Estimated quaternion is normalized in the final step.

3.4.3. Implementation of Visual – Inertial Odometry

Odometry sensor fusion is implemented in two different aspects. First, visual position estimate is refined using gyroscope readings. Second, inertial orientation estimate is substituted into visual orientation estimate. Step-by-step description of the stereo visual-inertial odometry is given as:

1. Gyroscope readings are transformed using *Continuous Wavelet Transform (CWT)* using *Fast Fourier Transform (FFT)*. Time-frequency representation of gyroscope signals is revealed as a result of CWT. Unlike standard FFT representation, high frequency components of the signal can be detected in time-domain using CWT. It is seen that high-frequency gyroscope data correspond to blurry images. Using a pre-defined empirical threshold value, high-frequency data points corresponding to blurry images are selected.

2. Since the sampling time of IMU is 10 times larger than stereo camera system, visual and inertial timestamps are synchronized. Next, corresponding blurry images which is a cause of wrong feature tracking are deleted. Therefore, more reliable feature detection, matching and tracking is provided.

3. Blurry images may correspond to fast movements, thus deleted images may introduce larger distances between tracked feature points. Therefore, disparity matching and feature tracking window sizes are enlarged when images are deleted.

4. Since the data is taken by a MAV, vibration affects the measurements. Thus, VO cannot make a good estimation of orientation due to the low sampling rate of the camera system. Therefore, orientation estimates from inertial measurements are substituted for visual orientation estimates in the final step.

Figure 3.12 represents the block diagram of stereo visual-inertial odometry in the next page.

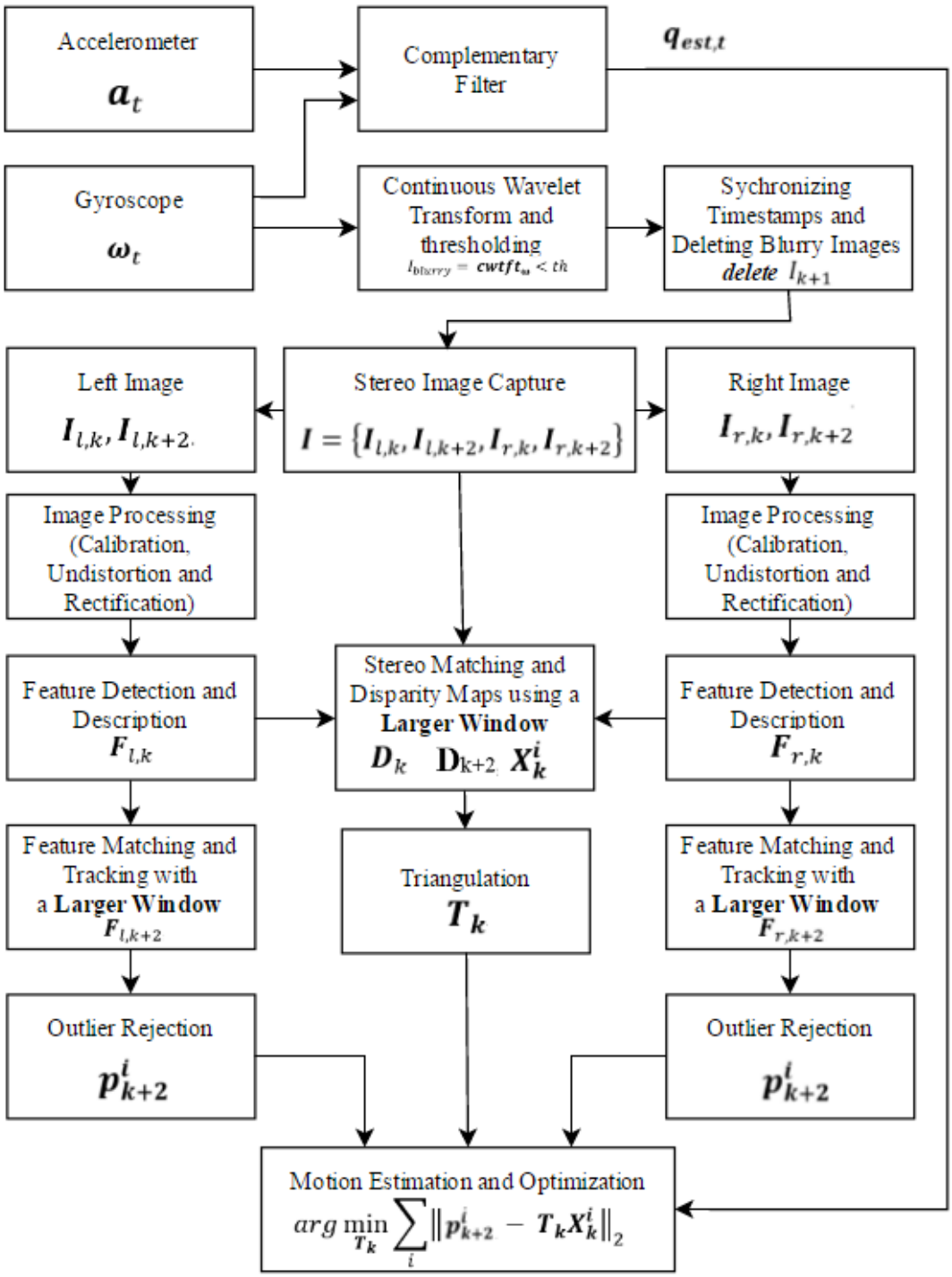

Figure 3.12 Block diagram of fusion of visual and inertial measurements

3.5. Closure

This section summarizes the theory behind the visual-inertial odometry since it is very important to have a broad understanding of machine vision before implementing the algorithms. Various approaches to different problems are presented for both VO, inertial navigation and sensor fusion. Therefore, this section is kept relatively long, explanatory and illustrative. In section 3.4, proposed methodology is presented with the details of the implementation. The results and discussion of the implementation is given in Chapter 5.

CHAPTER 4

EXPERIMENTAL SETUP AND DATASETS USED

For computer vision, there is a great number of datasets available online to verify developed algorithms. For this study, VO or SLAM datasets are utilized. Most popular datasets in this research area are *KITTI* Vision Benchmark Suite [81] and *Middlebury Stereo Datasets* [82]. However, being an outdoor dataset, *KITTI* is not suitable for this research. Outdoor VO is different from indoor VO due to the different characteristics of the features that are located far away from each other. Moreover, stereo vision converges to monocular when the distance between camera and the scene is much larger than the baseline of the stereo camera pair. *Middlebury* dataset, on the other hand, consists of very small movement of the camera which cannot simulate indoor path estimation and lacks IMU data. *Rawseeds Project's Bicocca* dataset [83] is utilized in which various sensors are used to take data including IMU and stereo vision in different indoor environments. However, it is observed that there are leaps and significant disconnections in some parts of the ground truth data. Therefore, *the EuRoC MAV dataset* [84], a more reliable and stable dataset in which high quality sensors are gathered on a MAV to collect data, is utilized throughout this thesis study.

4.1. The EuRoC MAV Dataset

The dataset is recorded in the context of the *European Robotics Challenge (EuRoC),* to evaluate visual-inertial SLAM and 3D reconstruction algorithms on MAVs. EuRoC MAV dataset, published in [84], presents visual-inertial sensor data collected on-board a MAV. Synchronized stereo images, accelerometer and gyroscope measurements and accurate ground truth is included in the dataset. Data is taken in two different environments. The first batch of real flight data is collected in an industrial environment for the evaluation of visual-inertial localization algorithms. The data contains millimeter accuracy position ground truth from a laser tracking system. The second batch of the dataset is taken for 3D reconstruction algorithms and the ground truth is recorded with a motion capture system. A total of eleven datasets are provided ranging from slow flights under good visual conditions to dynamic flights with motion blur and poor illumination.

4.1.1. Sensor Setup

The main features and sensor specifications of *EuRoC MAV* dataset are provided below:

- Sensor system
  - Monochrome *Aptina MT9V034* global shutter stereo camera system at 2×20 FPS
  - *ADIS16448* 6-axis IMU - gyroscope and accelerometer - at 200 Hz
  - Sensors are aligned using shutter-centric based alignment
- Calibration
  - Camera intrinsic parameters are provided
  - Camera-IMU transformation matrices are provided
  - Stereo calibration checkerboard images are provided
  - Spatio-temporally alignment with ground truth is provided
- Ground Truth Verification
  - 6D pose is gathered using *Vicon* motion capture system at 100 Hz
  - 3D position is obtained using *Leica MS50* laser tracker at 50 Hz

- Indoor datasets:
  - *Machine Hall* dataset
  - *Vicon Room* dataset

An AscTec Firefly hex-robot is used to collect data as shown in Figure 4.1. Visual inertial sensor unit is mounted in a front-down looking position to enable an unobstructed stereo view. A respective prism for laser tracker and markers for motion capture system are mounted at the top of the MAV to track the pose of the vehicle. Stereo images and IMU readings are logged and timestamped on-board. However, ground truth data is logged on the base station. Thus, a maximum likelihood estimator is used to align and synchronize the data.

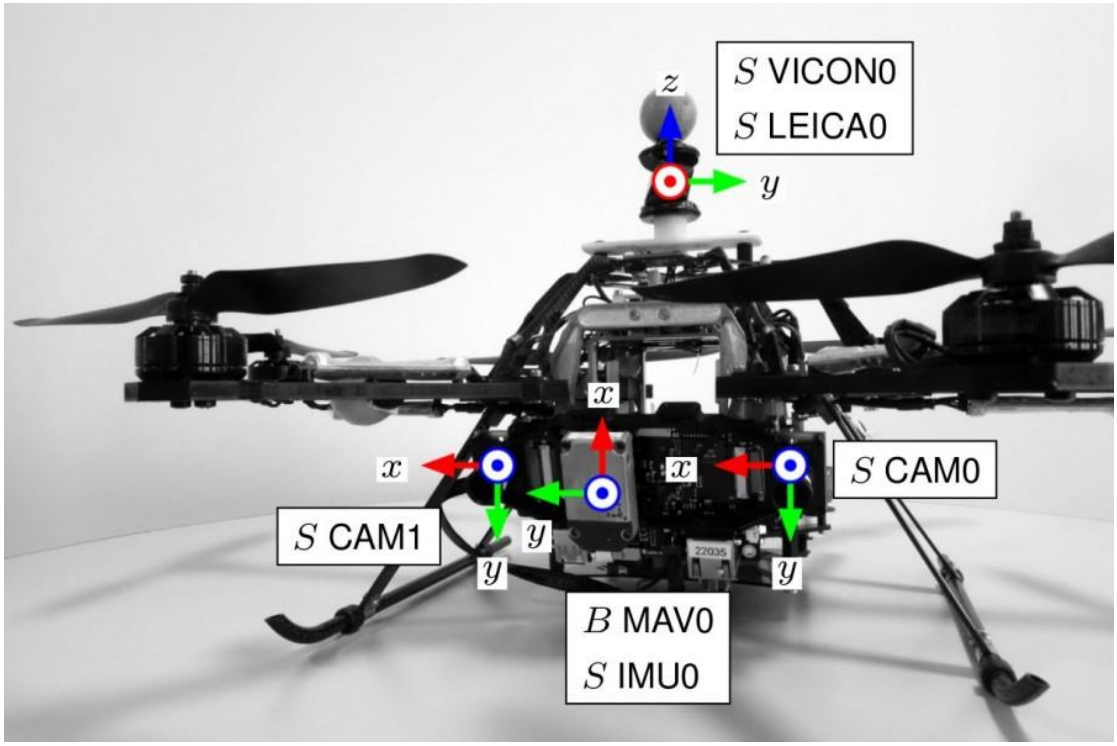

Figure 4.1 EuRoC MAV dataset, data collector micro air vehicle [84]

Each sensor data is logged with respect to its own reference coordinate frame *S*. The sensor system that was used to capture the datasets and the transformations between their coordinate frames are depicted in Figure 4.2. Ground truth data is measured with respect to prism/marker coordinate frame. Body frame of the MAV is located on the IMU coordinate frame and transformation of each sensory system with respect to the base frame is provided with the datasets.

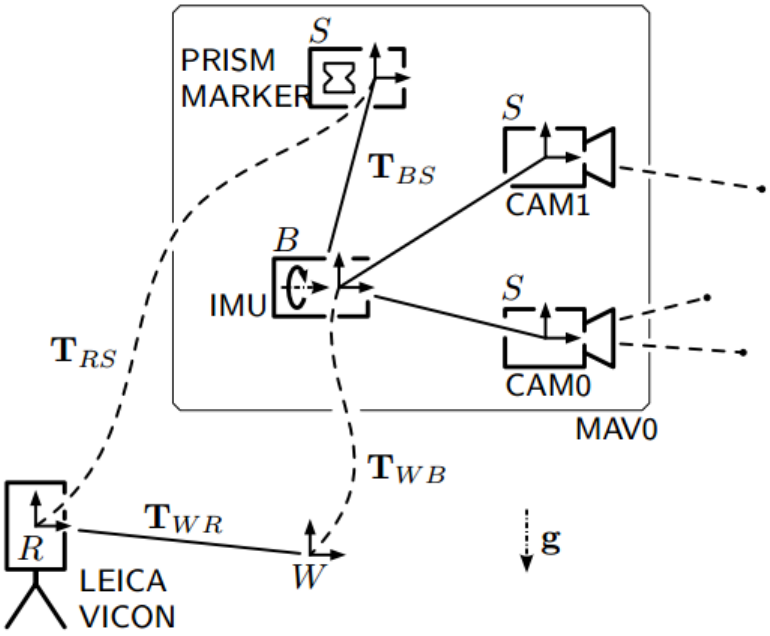

Figure 4.2 The sensor setup that is used to capture datasets [84]

4.1.2. Industrial Machine Hall Dataset

This dataset is recorded in a large machine hall shown in Figure 4.3 which represents a challenging industrial environment for visual-inertial motion estimation or SLAM algorithms. Ground truth data is taken by Leica Nova MS50 laser tracker and recorded on the base station. There are a total of five datasets taken in machine hall from easy to medium. Table 4.1 shows the total length, total duration, average velocities and explanations for each dataset.

Figure 4.3 A representative image from machine hall dataset [84]

4.1.3. Vicon Room Dataset

This dataset is recorded in a specially prepared 8 m x 8.4 m x 4 m room. There are two different configurations for this dataset in different obstacle scenarios. There are also moving curtains in the dataset which makes motion estimation difficult. Ground truth data is taken by Vicon motion capture system. Leica laser scanner is also utilized for point cloud data as shown in Figure 4.4.

4.1.4. Dataset Format

Format and the conventions of the dataset is elaborated in this section. Sensor systems are rigidly attached to the *body frame* as denoted by *B*. Figure 4.1 shows the sensors and Figure 4.2 shows the frames and their convention with respect to each other. *B* moves with respect to a *world frame* as denoted by *W*. Sensor's reference frame is used to represent the raw sensor data of each sensor as denoted by *S*. Extrinsic parameters

showing the transformation between *S* and *B* are provided in the sensor's *yaml* file in the datasets. Ground truth data provides transformation between *B* and *W*.

Each sensor in the datasets consists of two different files. One is *yaml* files containing the sensor type and 4x4 homogenous transformation matrix representing the extrinsic parameters of the sensor with respect to the body frame. The other file is *Comma Separated Value (CSV)* data file containing sensor data with nanoseconds resolution timestamps. First raw of the CSV data file includes the name of the sensor with the SI units in square brackets.

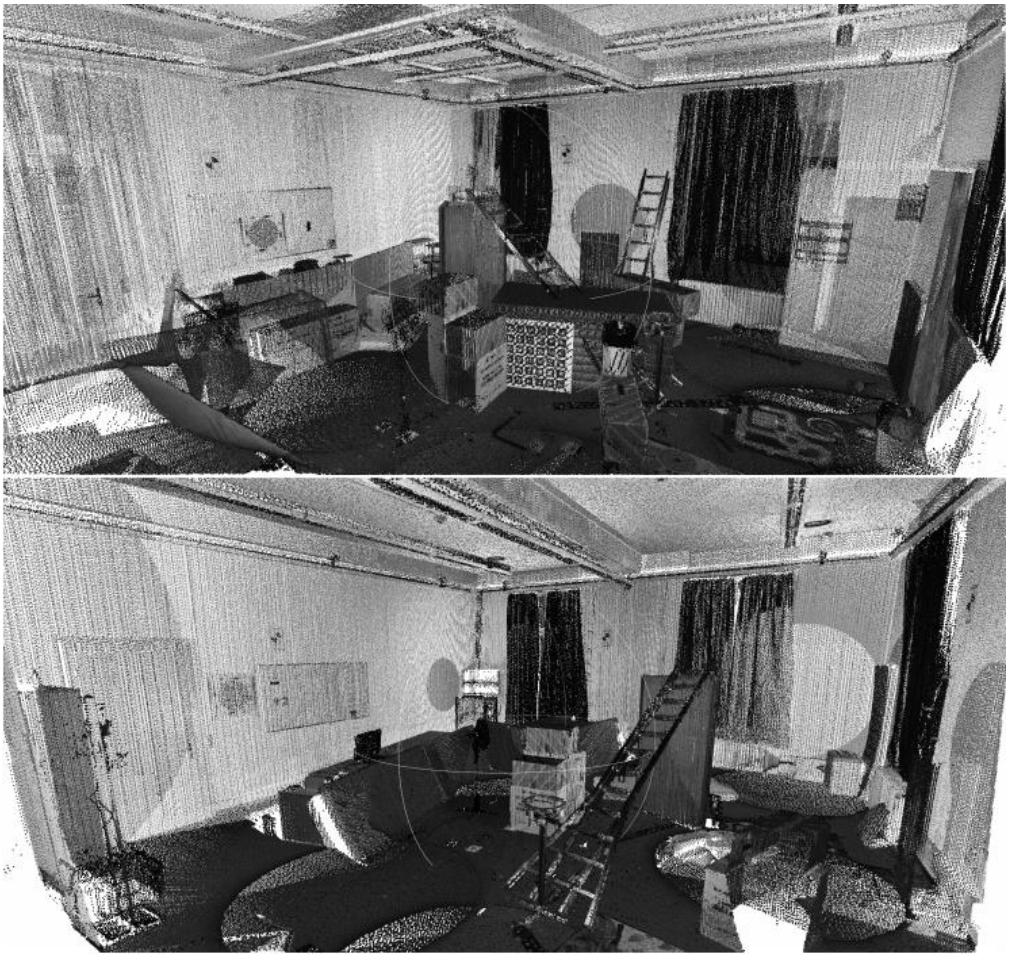

Figure 4.4 Point cloud of Vicon room in 2 different configurations [84]

Table 4.1 Dataset characteristics

Name	Length / Duration	Avg. Vel. / Angular Vel.	Note
MH_01_easy	80.6 m 182 s	0.44 m s $^{-1}$ 0.22 rad s $^{-1}$	good texture, bright scene
MH_02_easy	73.5 m 150 s	0.49 m s $^{-1}$ 0.21 rad s $^{-1}$	good texture, bright scene
MH_03_medium	130.9 m 132 s	0.9 m s $^{-1}$ 0.29 rad s $^{-1}$	fast motion, bright scene
MH_04_difficult	91.7 m 99 s	0.93 m s $^{-1}$ 0.24 rad s $^{-1}$	fast motion, dark scene
MH_05_difficult	97.6 m 111 s	0.88 m s $^{-1}$ 0.21 rad s $^{-1}$	fast motion, dark scene
V1_01_easy	58.6 m 144 s	0.41 m s $^{-1}$ 0.28 rad s $^{-1}$	slow motion, bright scene
V1_02_medium	75.9 m 83.5 s	0.91 m s $^{-1}$ 0.56 rad s $^{-1}$	fast motion, bright scene
V1_03_difficult	79.0 m 105 s	0.75 m s $^{-1}$ 0.62 rad s $^{-1}$	fast motion, motion blur
V2_01_easy	36.5 m 112 s	0.33 m s $^{-1}$ 0.28 rad s $^{-1}$	slow motion, bright scene
V2_02_medium	83.2 m 115 s	0.72 m s $^{-1}$ 0.59 rad s $^{-1}$	fast motion, bright scene
V2_03_difficult	86.1 m 115 s	0.75 m s $^{-1}$ 0.66 rad s $^{-1}$	fast motion, motion blur

4.1.5. Rotations and Transformations

Quaternion representation is used for rotations. Detail explanation of rotation quaternions can be found in Appendix E. The rotation quaternions are shown as:

$$\boldsymbol{q} = \begin{bmatrix} q_w \; q_x \; q_y \; q_z \end{bmatrix}^T = \begin{bmatrix} q_w \\ \overline{\boldsymbol{q}} \end{bmatrix} \tag{4.1}$$

where q_w and $\overline{\boldsymbol{q}}$ denotes real and imaginary parts of the quaternion, respectively. Using **q**, direction cosine matrix transforming between *B* to *W*, $\boldsymbol{C}_{WB}(\boldsymbol{q}_{WB})$, can be computed:

$$\boldsymbol{C}(\boldsymbol{q}) = q_w^2\, \boldsymbol{I}_{3x3} + 2\, q_w\, [\overline{\boldsymbol{q}}\times] + [\overline{\boldsymbol{q}}\times]^2 + \boldsymbol{q}\boldsymbol{q}^T \tag{4.2}$$

where $[a\times]$ is the skew-symmetric matrix of vector a. A position estimation vector $\overrightarrow{KL}$ expressed in *B*, $\boldsymbol{p}_{KL}^{B}$, can be expressed in *W* as:

$$\boldsymbol{p}_{KL}^{W} = \boldsymbol{C}_{WB}\,(\boldsymbol{q}_{WB})\,\boldsymbol{p}_{KL}^{B} \tag{4.3}$$

4x4 homogenous transformation matrix is defined as:

$$\boldsymbol{T}_{WB} = \begin{bmatrix} \boldsymbol{C}_{WB} & \boldsymbol{p}_{WB}^{W} \\ 0 & 1 \end{bmatrix} \tag{4.4}$$

4.2. Dataset Results

In this section, studies that use available datasets are presented in order to give insight about the expected results from the odometry and motion estimation algorithms. First of all, Table 4.2 is presented to show outdoor VO results from KITTI dataset [81]. Stereo VO translational and rotational errors are presented as offset ratio. Contestant methods are tested on all eleven sequences of the benchmark and the errors are calculated for different subsequences (i.e. 5, 10, 50, …, 400 meters). Table 4.2 shows the average of results from all sequences. Translational error is computed as percentages while rotational error is presented in degrees per meter. Ground truth is taken using a GPS aided inertial navigation system with centimeter level accuracy.

EuRoC results not available online, but results of some of the motion estimation algorithms using EuRoC MAV datasets are published recently. In one of the works, Oleynikova et al. [85] shows that an integrated visual-inertial odometry and localization systems can be run in real-time on-board robots. The study utilizes BRISK descriptors and graph filtering to remove outliers. Then, PnP RANSAC is used to estimate motion and further refine the inliers. Finally, bundle adjustment is done for pose estimation optimization. Algorithm is tested on machine hall datasets. For the *MH_01* dataset, for 77 m and 182 sec run, algorithm gives 0.37±0.23 meters pose estimation error in translation and 0.13±0.11 radians rotational errors. Rotational error makes a total of 13 degrees and translational error makes a total of 0.6 meters which corresponds to 0.78 percent of covered distance.

In one of the works by Krombach et al. [87], combination of feature based and direct methods is used which utilizes FAST or Harris features in combination with ORB descriptor. Again a PnP RANSAC scheme is used to estimate the motion. Unlike the VO, this method uses a SLAM approach, extracts the local map of the triangulated 3D points and uses a key frame based pose estimation. *Absolute Trajectory Error (ATE)* is used for the error metric which gives the *Root Mean Square (RMS)* error in each frame location. Vicon room dataset is used for this study. 0.22-meter ATE and 0.35 m maximum error is reported for the *V2_01* dataset which corresponds to 0.61 percent of the covered distance.

Table 4.2 Odometry translation and rotational errors from KITTI [86]

Rank	**Method**	**Translation**	**Rotation [°/m]**
1	MFI	1.69 %	0.0066
2	VoBA	1.77 %	0.0066
3	VISA2-SLANM	1.88 %	0.0152
4	SSLAM	1.87 %	0.0083
5	eVO	1.93 %	0.0076
6	D6DVO	2.10 %	0.0083
7	GT-VO3pt	2.21 %	0.0117
8	VISO2-S	2.27 %	0.0152
9	BoofCV-SQ3	2.27 %	0.0111
10	TGVO	2.44 %	0.0105
11	SVO	2.45 %	0.0109
12	SSLAM-HR	2.45 %	0.0112
13	KPnP	2.73 %	0.0107
14	VO3pt	2.93 %	0.0116
15	VO3ptLBA	3.17 %	0.0180
16	MSD VO	3.50 %	0.0166
17	MLM-SFM	4.07 %	0.0104
18	VOFS	4.21 %	0.0158
19	VOFSLBA	4.35 %	0.0189
20	VISO2-M	13.79 %	0.0372

Fu et al. [88] evaluates their stereo visual-inertial odometry algorithm using Vicon room dataset *V2_01* before implementing their real-world small scale ARM-based stereo vision preprocessing system. Their algorithm tracks a certain number of FAST features with BRIEF descriptor. Finally, they apply a bundle adjustment optimization, selecting the key frames when the tracked feature number is less than a threshold or camera motion is larger than a threshold. Implementation of the algorithm gives 0.43 meter RMS translational error corresponding to 1.2% of the covered distance and 6.5 degrees rotational error.

Table 4.3 summarizes the results of the works on EuRoC MAV datasets. As seen in the table, translational errors are between 0.6 and 1.2 percent of the covered distance. However, orientation estimation errors are high since their estimate only relies on VO estimates.

4.3. Closure

This chapter analyzes the datasets that are utilized throughout the experimental studies of the thesis. There are many datasets available in the literature, however, it is important to make use of a prestigious and reliable datasets in order to prevent later complications and problems. Therefore, EuRoC MAV datasets are chosen for the evaluation of the algorithms. Results of different algorithms using the datasets are also presented in this chapter.

Table 4.3 Odometry translation and rotational errors from various studies

Reference	**Oleynikova et al. [85]**		**Krombach et al. [87]**		**Fu et al. [88]**	
Datasets	***MH_01***	***V2_01***	***MH_01***	***V2_01***	***MH_01***	***V2_01***
Translation error [m]	0.37	-	-	0.22	-	0.43
Translational error [%]	0.78	-	-	0.61	-	1.2
Rotational error [°]	13	-	-	-	-	6.5

CHAPTER 5

EXPERIMENTAL RESULTS AND DISCUSSION

In this chapter, experiments of developed algorithms are described and the results are shown in a proper manner. First, the results of the implementations are shown in three parts: inertial-only, visual-only, and visual inertial. Next, the interpretations of the results are included in the discussion part.

5.1. Experimental Results

Experimental results are examined in three sections. First, inertial odometry and orientation estimates are shown. Second, VO results are presented. Lastly, fusion of stereo images and inertial measurements, titled as *Visual-Inertial Odometry (VIO),* is presented. For the translational components, errors are calculated using quadratic mean known as *Root Mean Square (RMS)* for each frame location. This error metric is standard in VO literature as in [3, 43, 87]. However, it is important to give the percentage of the error with respect to the covered distance to compare the results from different datasets. In this study RMS errors, maximum errors and error percentages are given for translational position estimates. Only RMS errors and maximum errors are used for orientation estimation errors.

5.1.1. Inertial Odometry Implementation

Change in velocity, orientation, and position are estimated using inertial data. Two inertial sensors (namely, accelerometer and gyroscope) combined in an IMU are utilized for this purpose. Accelerometer produces principal components (including gravity vector) by measuring the body forces. Compensating the gravity vector, only actual acceleration data are used. Gyroscope, on the other hand, measures the rotational velocities. Both sensors' data are fused using the gradient-descent-based explicit complementary filter [80] explained in Chapter 3.4.2. Orientation estimate of the fusion is improved using a line-fitting drift elimination technique using the stationary periods. Original implementation is called "*estimate*" while the improved version is called "*improved estimate*" for simplicity.

Accelerometer and gyroscope used in *EuRoC MAV* dataset are calibrated beforehand. Thus, their data are used without any pre-processing. Using *MATLAB R2016a*, raw and filtered readings from the sensors, orientation, velocity and position estimates are plotted for *Machine Hall MH_01* and *Vicon Room V2_01* datasets. Figure 5.1 and Figure 5.2 show raw and filtered accelerometer readings for *MH_01* and *V2_01* datasets, respectively. Stationary periods can be observed easily after filtering the data with a high-pass filter followed by a low-pass filter as explained in Chapter 3.3.1. While MAV is stationary in the first and last three seconds in *V2_01* dataset, it is stationary between the 20^{th} and 40^{th} seconds in *MH_01* dataset. Stationary periods are used in drift elimination of orientation estimates.

Figure 5.3 and Figure 5.4 illustrate the transformed gyroscope readings for *MH_01* and *V2_01* datasets, respectively. *Continuous Wavelet Transform (CWT)* using *Fast Fourier Transform (FFT),* which has the ability to construct a time-frequency representation of a signal, is applied to gyroscope data in order to detect high-frequency components corresponding to the blurry images.

Furthermore, Figure 5.5 and Figure 5.6 present the *Power Spectral Density (PSD)* of gyroscope and accelerometer signals from *MH_01* and *V2_01* datasets, respectively. There are high amplitude components in gyroscope signals at very low frequencies.

Flat PSD of accelerometer signal means that the sensor noise is small [89]; therefore, the direct integration of the acceleration will yield velocity and position without further operation. However, in both datasets, there are obvious high-amplitude frequency components in accelerometer spectral density. Thus, filtering methods are utilized when integrating the acceleration to obtain velocity and position.

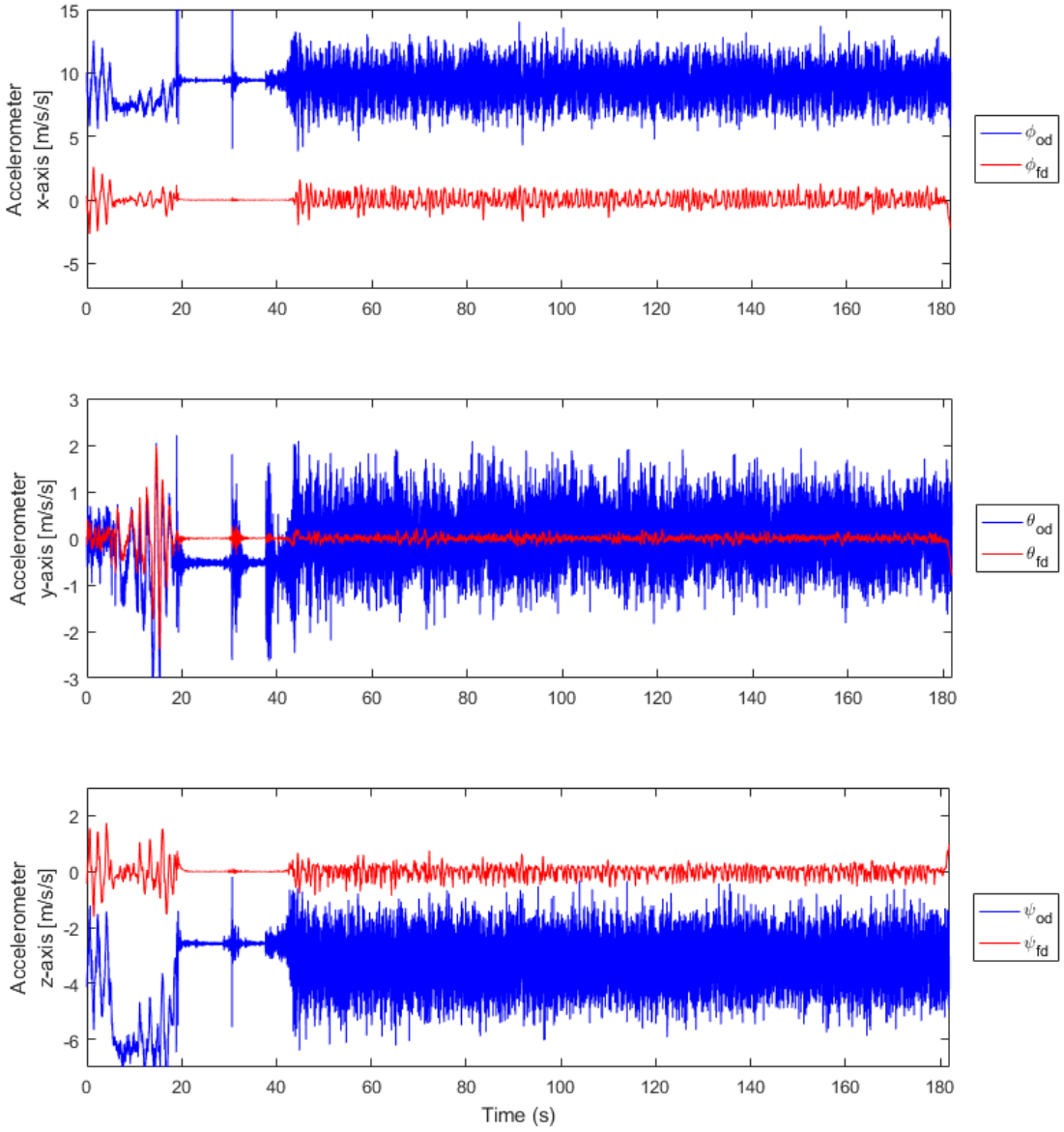

Figure 5.1 Original (blue) and filtered (red) accelerometer data for *MH_01* dataset

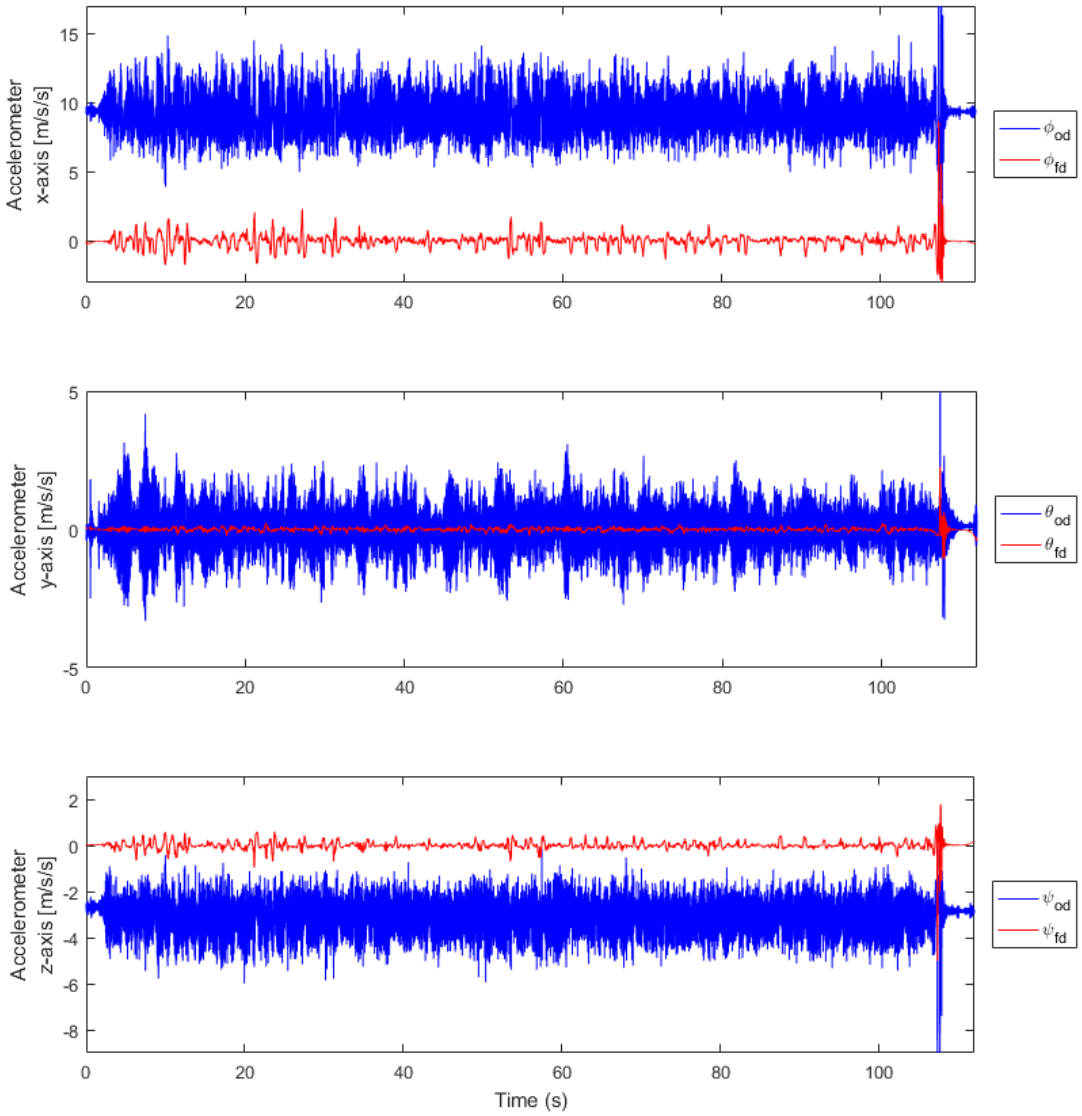

Figure 5.2 Original (blue) and filtered (red) accelerometer data for *V2_01* dataset

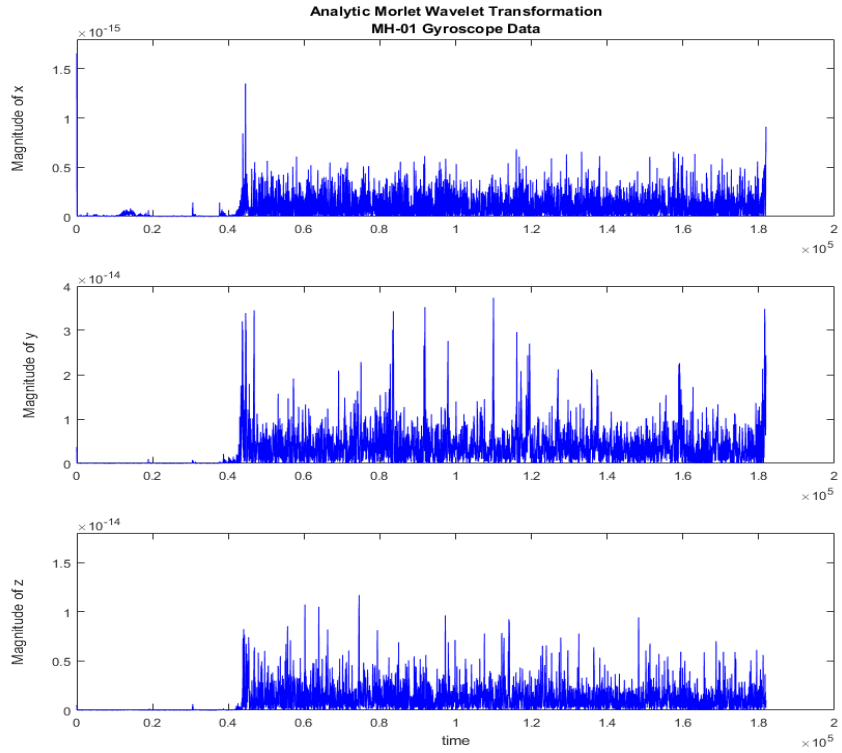

Figure 5.3 Continuous wavelet transform of gyroscope data from *MH_01* dataset

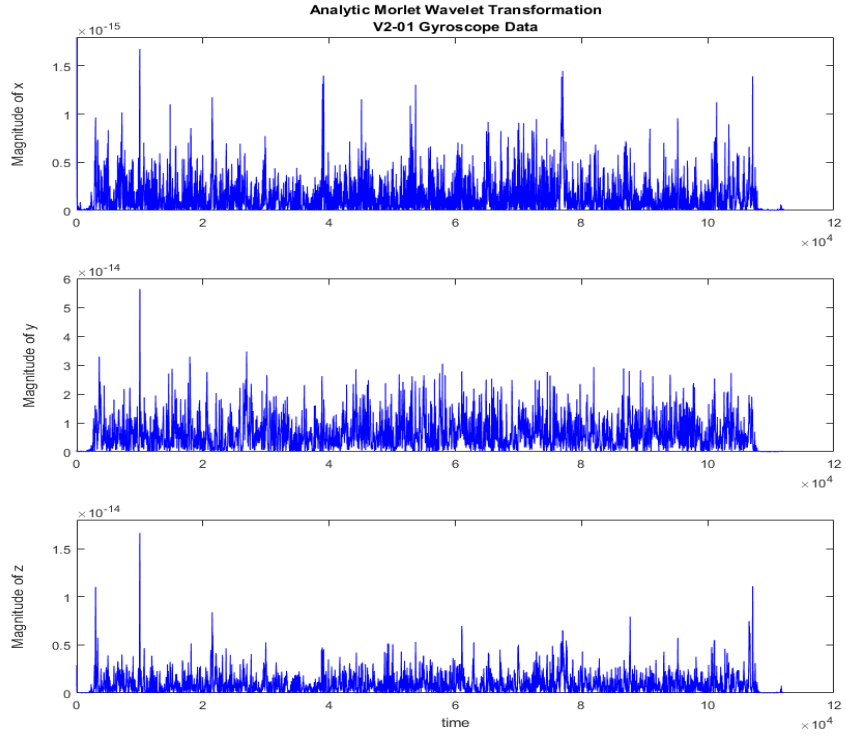

Figure 5.4 Continuous wavelet transform of gyroscope data from *V2_01* dataset

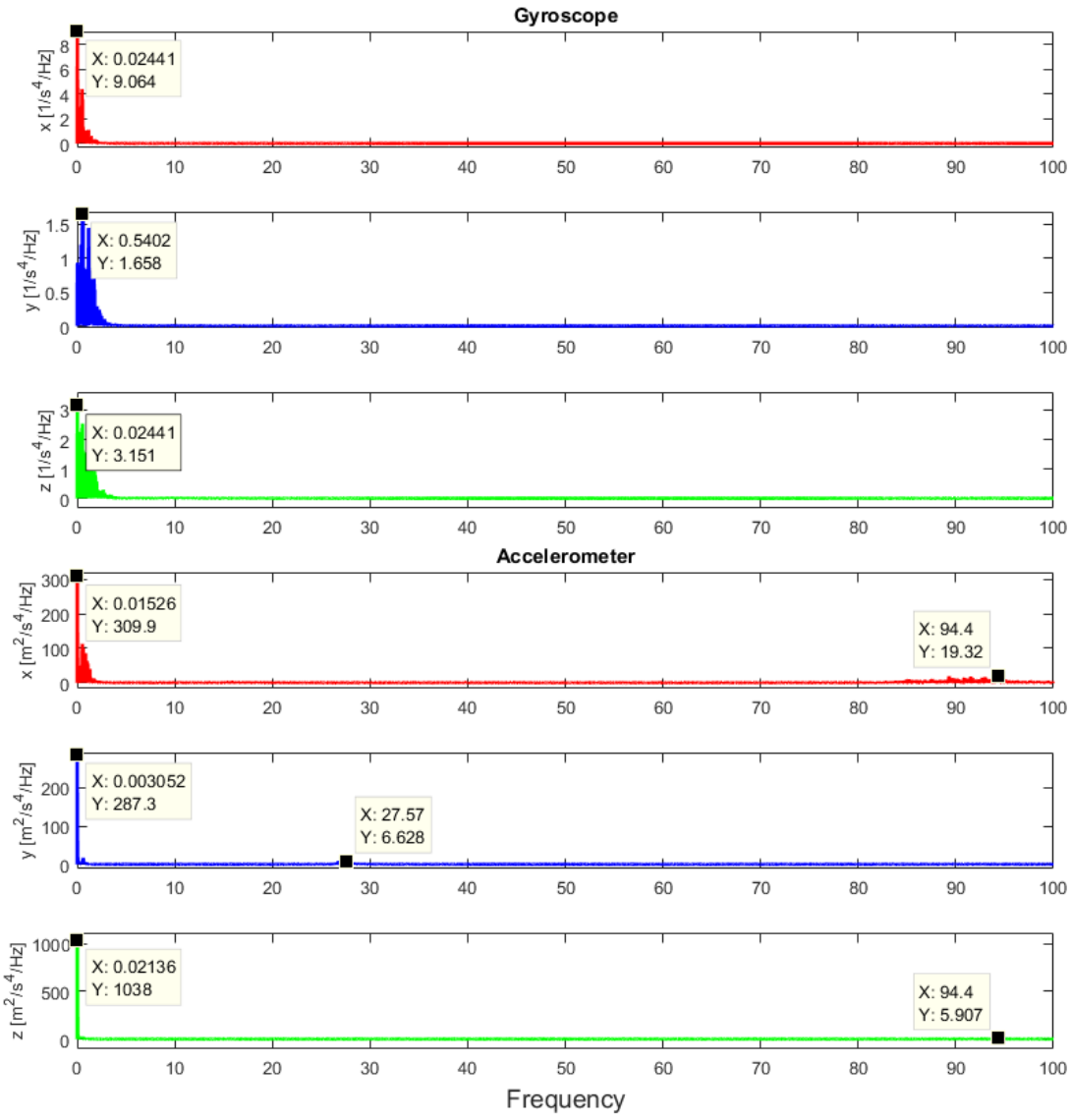

Figure 5.5 PSD of gyroscope and accelerometer signals from *MH_01*

Figure 5.7 presents orientation estimates, using only inertial measurements, represented in the world frame, W, for *MH_01* dataset. The implementation of gradient-descent-based explicit complementary filter is plotted in black whereas an improved version of the same filter based on a line-fitting drift elimination is shown in blue. Since there is no drift in x- and y-axes, these two estimates are overlaid; thus, only improved estimated quantities are visible in the first two figures. However, there is a significant drift in z-axis in the orientation estimate which is eliminated in the

improved estimate. RMS error in z-axis is decreased from 190.7 degrees to 14.4 degrees in the improved estimate. Figure 5.8 presents orientation estimate for *V2_01* dataset. Similar to *MH_01*, only z-axis drifts and it is eliminated in the improved estimate from 128.5 degrees to 16.4. Table 5.1 shows the RMS and maximum errors for these two algorithms.

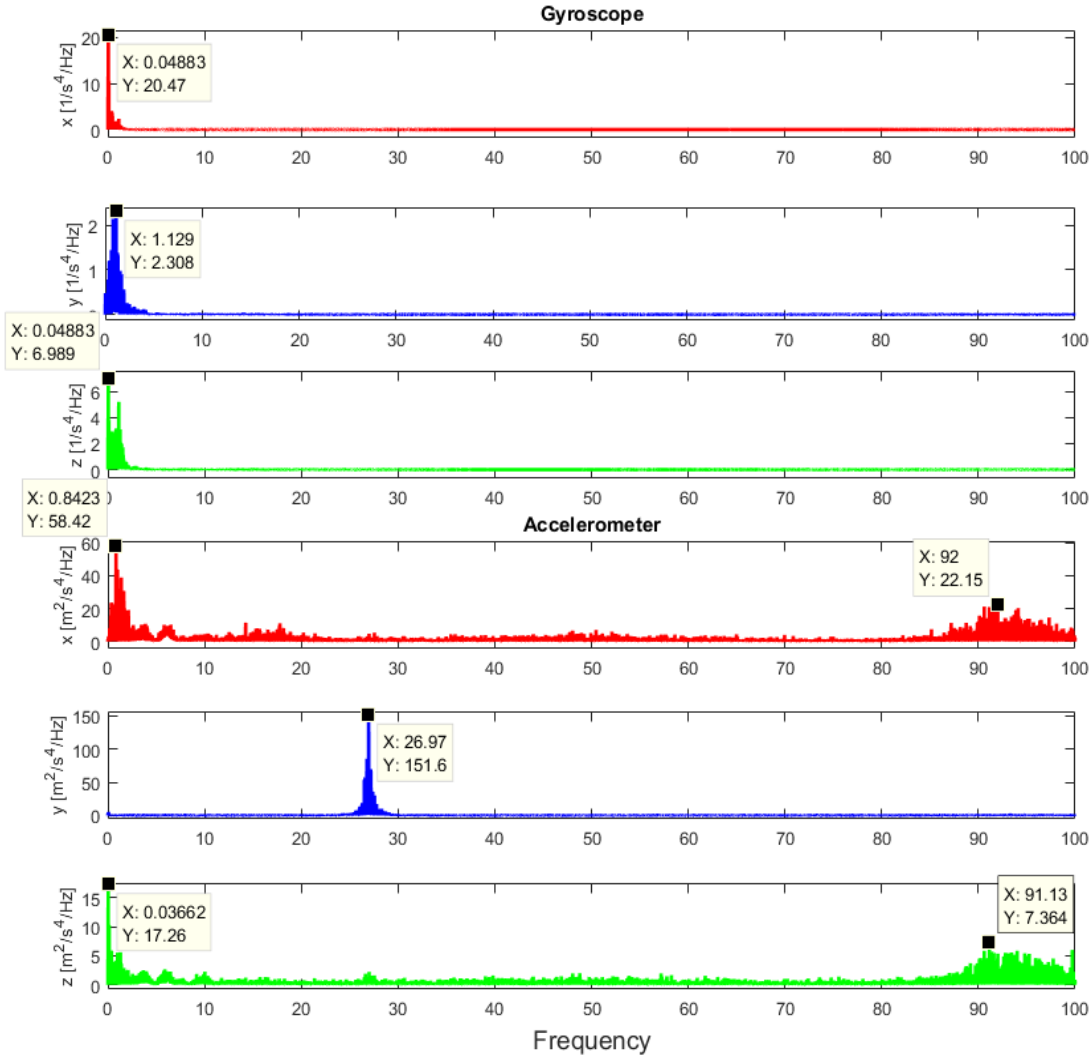

Figure 5.6 PSD of gyroscope and accelerometer signals from *V2_01*

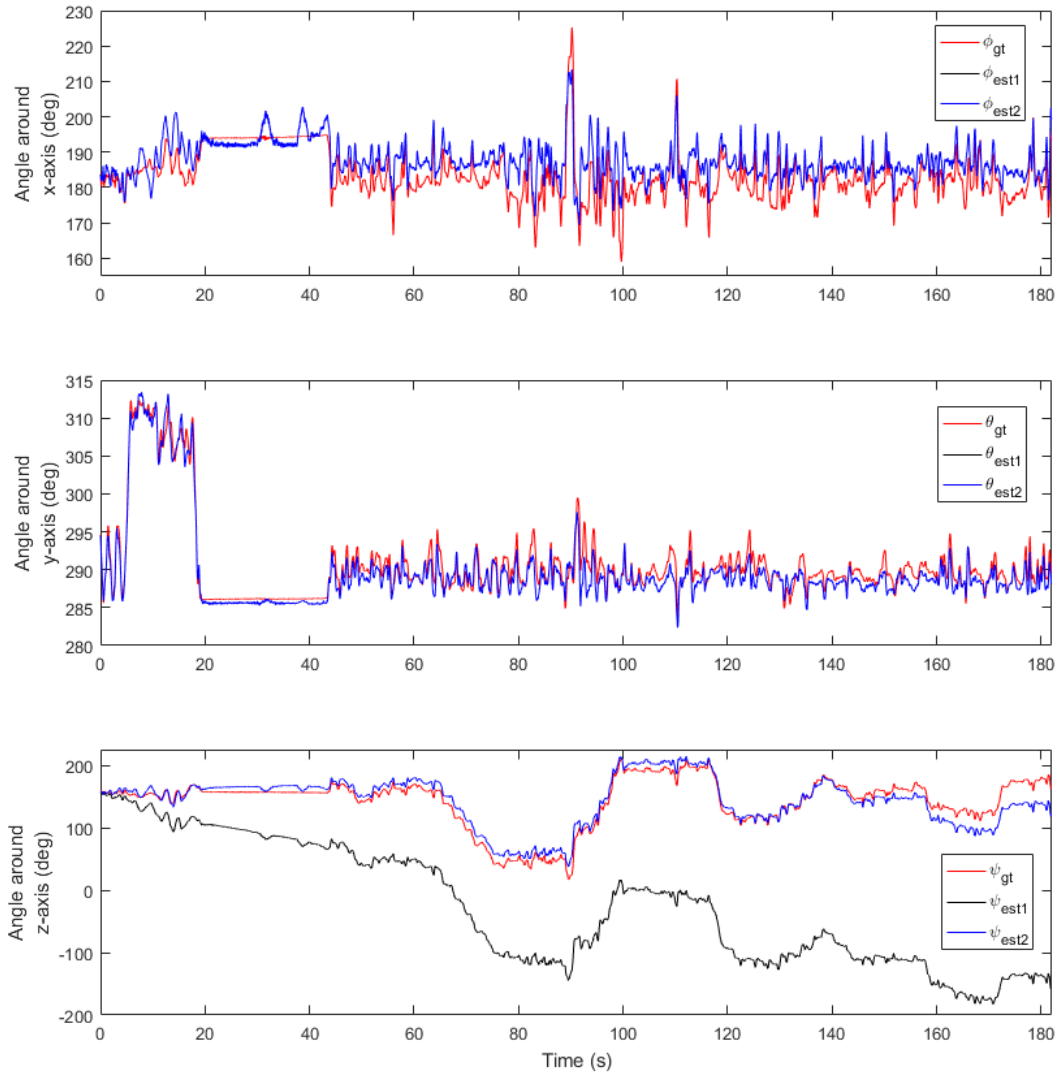

Figure 5.7 GT orientation vs. estimates from IMU readings for *MH_01*. Ground truth (red), estimate (black) and improved estimate (blue) data are plotted.

Similarly, Figure 5.9 and Figure 5.10 demonstrate the velocity estimates for *MH_01* and *V2_01* datasets, respectively. It is obvious from the figures that MAV is stationary between the 20th and 40th seconds in *MH_01*; however, there are small offsets in the estimates due to the drift elimination stage and filtering after the zero velocity update stage. RMS velocity error for *MH_01* dataset is found to be 0.5063 m/s while the maximum error is found as 1.3386 m/s. For the *V2_01* dataset, RMS velocity error is

0.3144 m/s while maximum velocity error is 0.7516 m/s. These error magnitudes can be considered as relatively large if compared to the average MAV velocity which is 0.44 m/s for *MH_01* and 0.33 m/s for *V2_01* respectively. Figure 5.11 and Figure 5.12 present the estimated positions from IMU. Although there are significant drifts in the plots, the results are promising for sensor fusion. Likewise, inertial orientation, velocity- and position estimation errors are summarized in Table 5.1.

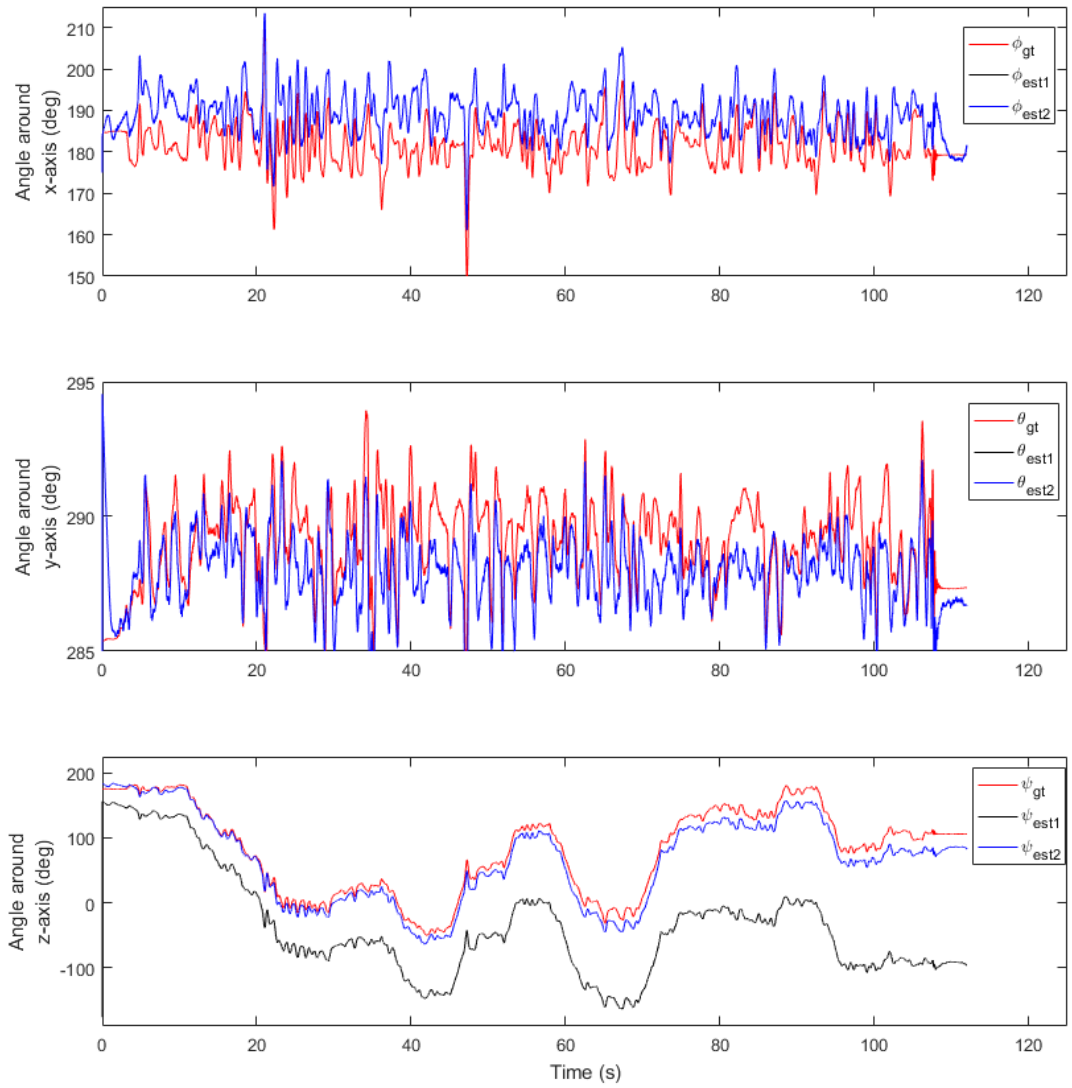

Figure 5.8 GT orientation vs. estimates from IMU readings for *V2_01*. Ground truth (red), estimate (black) and improved estimate (blue) data are plotted.

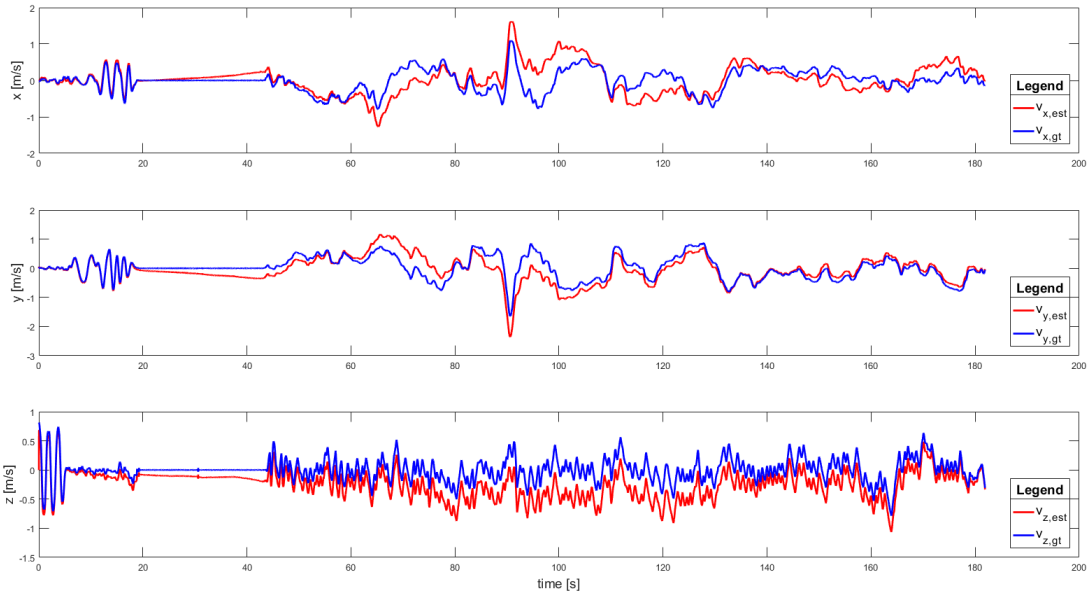

Figure 5.9 GT velocity vs. estimates from IMU readings for *MH_01*

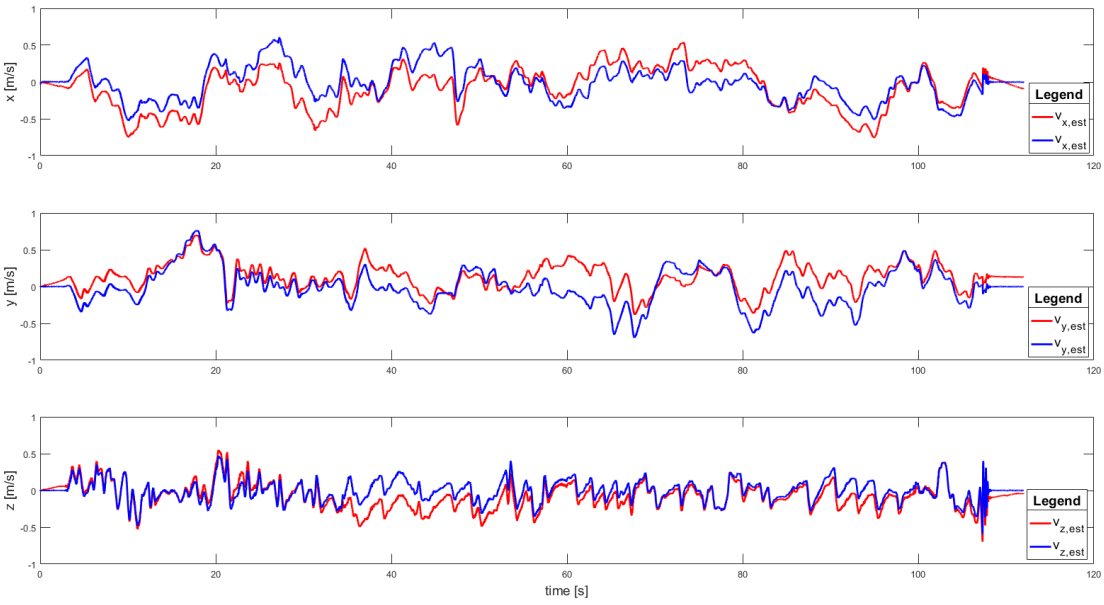

Figure 5.10 GT velocity vs. estimates from IMU readings for *V2_01*

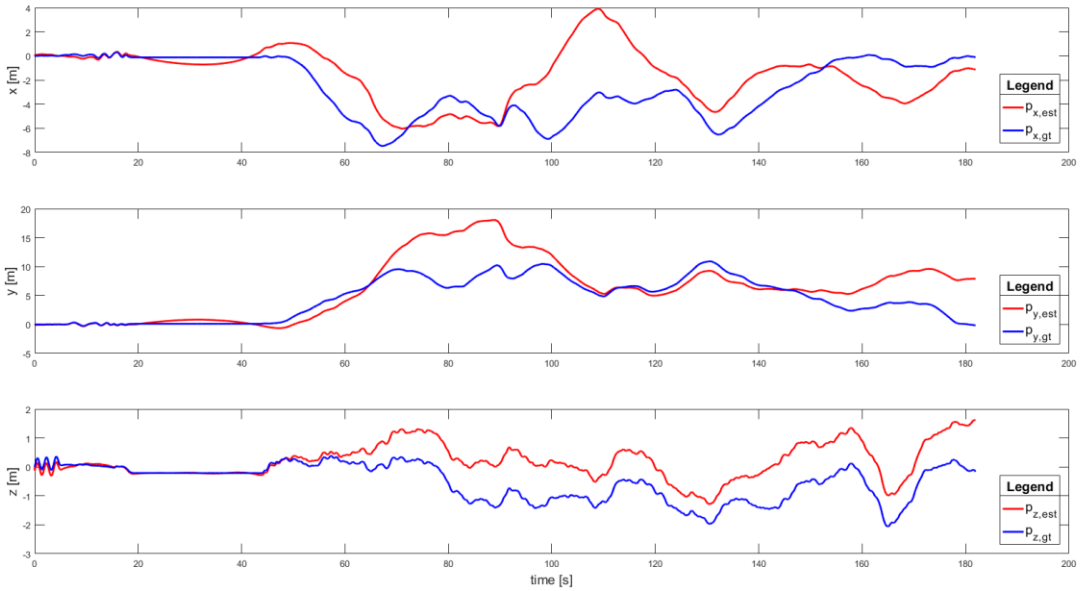

Figure 5.11 GT position vs. estimates from IMU readings for *MH_01*

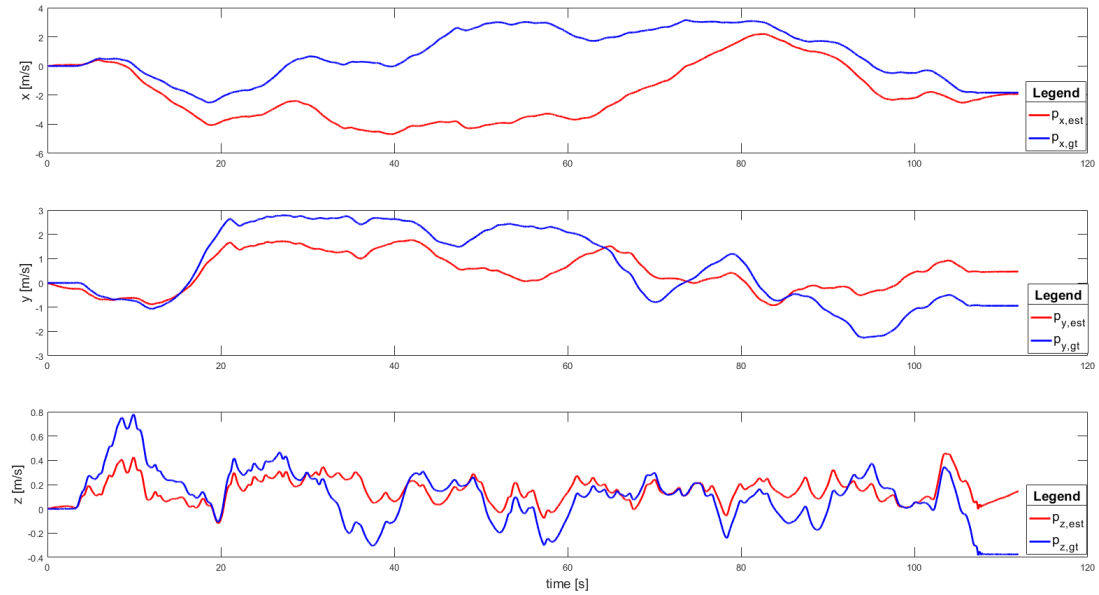

Figure 5.12 GT position vs. estimates from IMU readings for *V2_01*

Table 5.1 IMU pose estimate comparison

		MH_01		***V2_01***	
		RMS	**max**	**RMS**	**max**
Orientation [°]	**x**	5.7213	17.9406	8.211	17.6025
	y	1.3815	5.7455	1.571	4.7835
	z	14.3926	45.4524	16.3872	34.4991
	Σ	15.5496	49.2016	18.3965	38.9804
	Σ [%]	-	-	-	-
Velocity [m/s]	**x**	0.3165	0.9649	0.2022	0.4693
	y	0.2802	0.7860	0.2062	0.4809
	z	0.2786	0.4931	0.1242	0.3367
	Σ	0.5063	1.3386	0.3144	0.7516
	Σ [%]	-	-	-	-
Position [m]	**x**	3.5751	8.8311	3.4801	6.9696
	y	5.8735	10.4296	1.1163	2.2522
	z	0.5626	1.3047	0.1870	0.5215
	Σ	6.8990	13.7283	3.6595	7.3430
	Σ [%]	8.56	17.03	10.03	20.12

5.1.2. Visual Odometry Implementation

VO tests are again conducted on two different datasets, namely *Machine Hall MH_01* and *Vicon Room V2_01*. Note that VO implementations of *MH_01* take approximately 10 hours on an Intel I5, 3.2 GHz desktop computer while *V2_01* implementation lasts for 2 hours. One of the reasons for the long computing time is the number of frames per dataset since 3500 stereo image pairs are processed in *MH_01* dataset whereas 2000 frames are processed in *V2_01* dataset. Another reason is the number of frames tracked in consecutive images. Average processing time for each successive frames is given in Table 5.2.

Table 5.2 Average computing times between successive frames for VO algorithm

	MH_01	***V2_01***
# of processed frames	3500	2000
Disparity map	0.35 seconds	0.28 seconds
KLT and 3D feature location extraction	0.99 seconds	0.31 seconds
Outlier rejection and motion estimation	7.98 seconds	0.62 seconds
Optimization	1.84 seconds	0.94 seconds
Average time per loop	11.16 seconds	2.15 seconds
Average total time	10.8 hours	1.2 hours

As explained before, images are divided into a 6×8 grid and strongest 100 features are found in each grid in order to obtain a homogeneous distribution of features over the whole frame. Since grids are very small, exact number of 100 features cannot always be extracted. The reason why 100 features are specified is to ensure that the maximum number of possible features is found in each grid. Figure 5.13 presents the number of frames tracked over the *MH_01* left image sequence. As can be seen from the figure, the number of features tracked varies from a minimum of 38 to a maximum of 1696. This huge difference is due to the featureless and blurry images. Average number of tracked features for *MH_01* is found to be 821 in each image. Figure 5.14 illustrates the number of frames tracked over the *V2_01* left image sequence. Average number of features tracked between sequential images is 333. It can be observed that the number of tracked features decreases to a minimum of 14 in between 395^{th} and 396^{th} images. This significant difference is due to the fast rotations of the vehicle and the image sampling time. It is also important to note that the average number of features tracked in *MH_01* is more than two times of the average number of features tracked in *V2_01*.

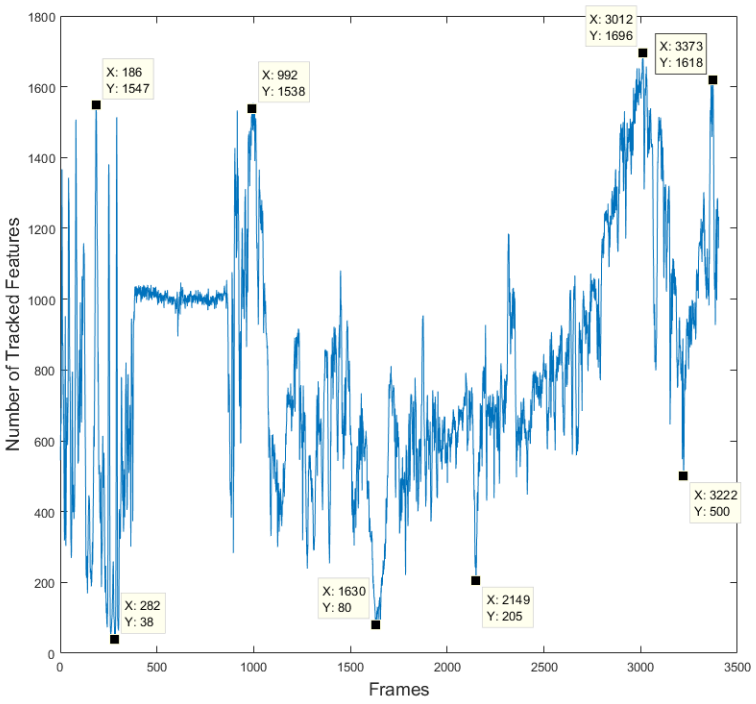

Figure 5.13 The number of features tracked over *MH_01* left image sequence

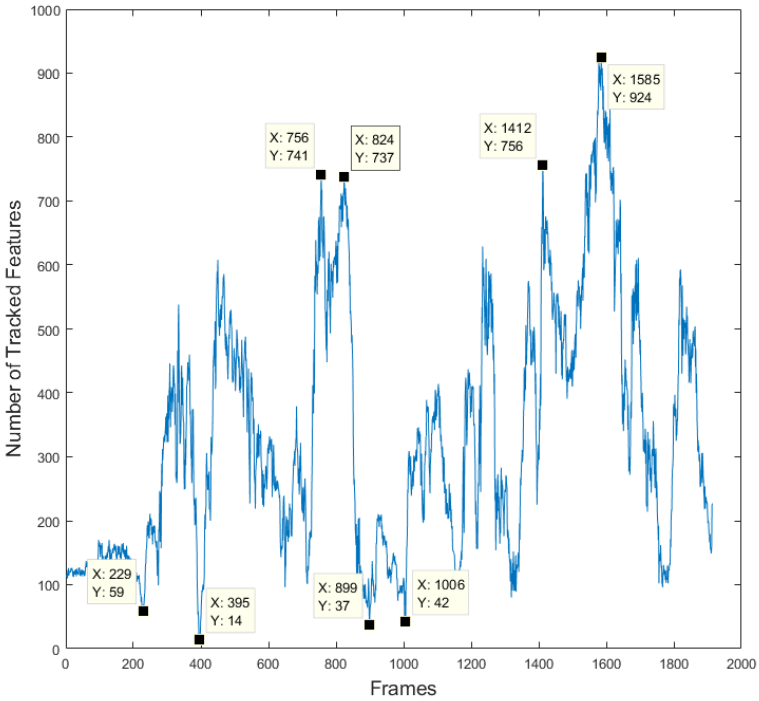

Figure 5.14 The number of features tracked over *V2_01* left image sequence

5.1.2.1. Machine Hall *MH_01* Dataset Results

The EuRoC MAV dataset is recorded in two different indoor places, one of which is a machine hall as described in Chapter 4.1.2. There are a lot of features in the area, yet the vibration of MAV is a big challenge to overcome for VO algorithm as the stereo image data is gathered at 20 Hz.

VO algorithm takes image pairs and uses rectified images in order to ease the manipulation. Rectification process used in the VO algorithm is explained in Appendix C. VO is preformed between consecutive stereo pairs to output transformation in each step. These transformations are then multiplied to form the cumulative transformation between the starting frame and the end frame.

Figure 5.15 illustrates the *ground truth (GT)* trajectory and the estimated VO trajectory on the same plot for xy-plane. It is known from the dataset that in the first 50 seconds, MAV is almost stationary where it takes off (hovers) and lands several times. The figure shows that the drift occurs proportional to the distance covered. Drift in time is almost negligible if compared to the drift on the distance covered. Figure 5.16 presents the sliced data, which indicates that the estimated trajectory is almost identical to the GT data. However, the orientation errors accumulate and generate a big deviation in the end. Likewise, Figure 5.17 shows the comparison in the xz-plane and Figure 5.18 indicates the sliced data. As can be seen, the sliced data have small errors since they consist of the beginning of the movement. Drift in z-axis is obvious in Figure 5.17.

Figure 5.19 presents the comparison of GT data and VO estimates separately at each axis. As can be seen from the figure, a sudden drift occurs around the 100^{th} second in y- and z-axes. Note that the RMS error in x-axis is calculated as 0.5236 m while the RMS error in y- and z-axes are 0.8293 m and 2.9163 m, respectively. Maximum errors are 1.2730 m, 2.0267 m and 5.5888 m for x-, y-, and z-axes, respectively. Similarly, 3D RMS error is calculated as 3.0768 m and 3D maximum error is calculated as 6.0797 m which corresponds to 3.82% and 7.54% of the total distance covered. Comparing the drifts in the graph with the dataset images, it can be observed that the very close flight opposite to a featureless plane caused the corresponding drift in that period.

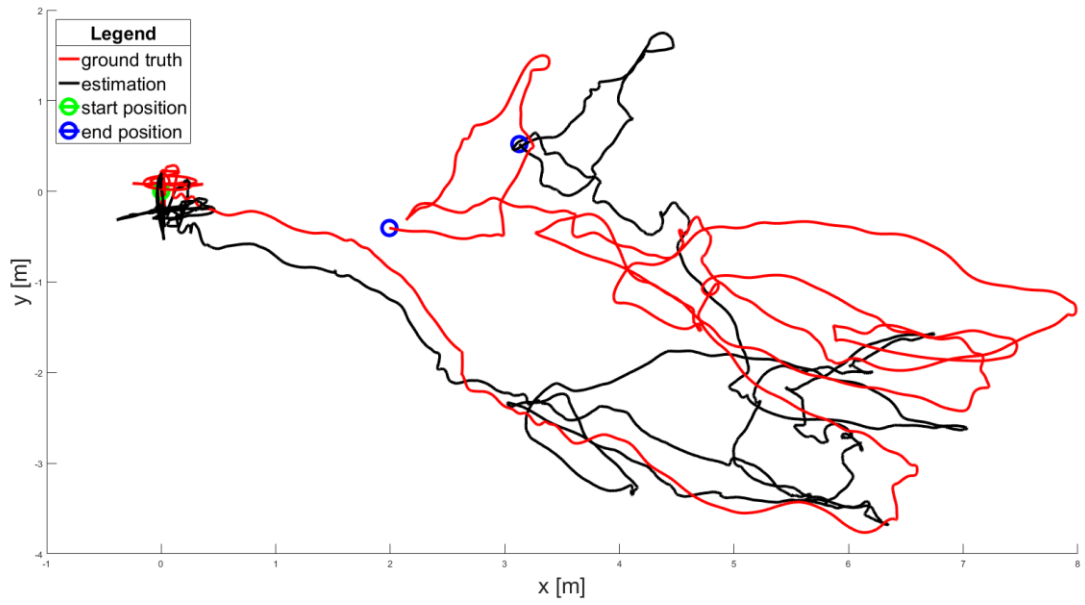

Figure 5.15 GT position vs. VO estimate in xy-plane for *MH_01*

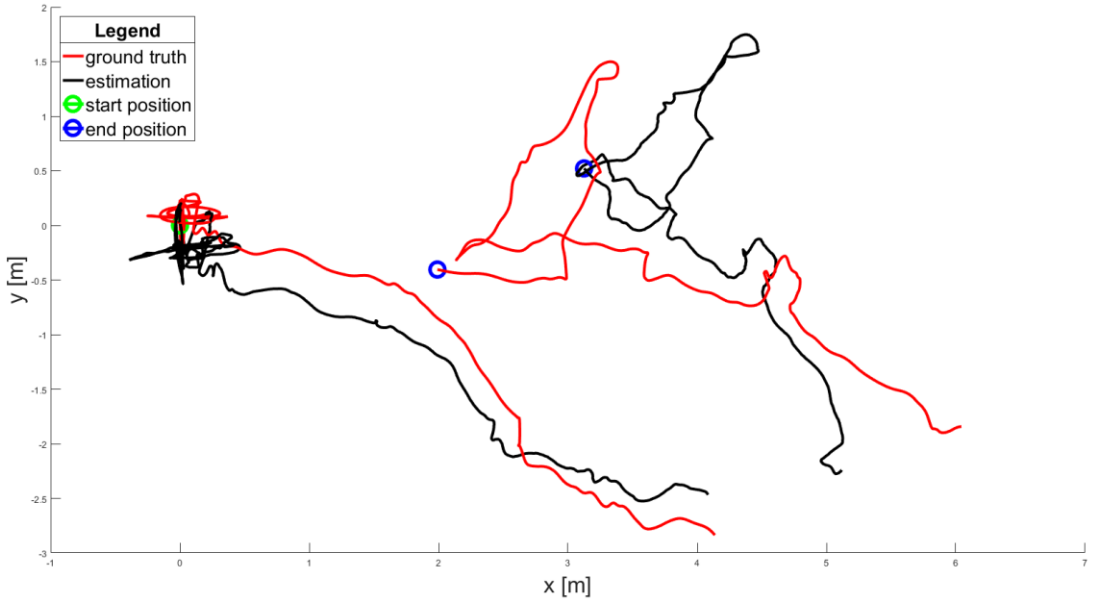

Figure 5.16 GT position vs. VO estimate sliced from frames 0-1550, 2800-3500

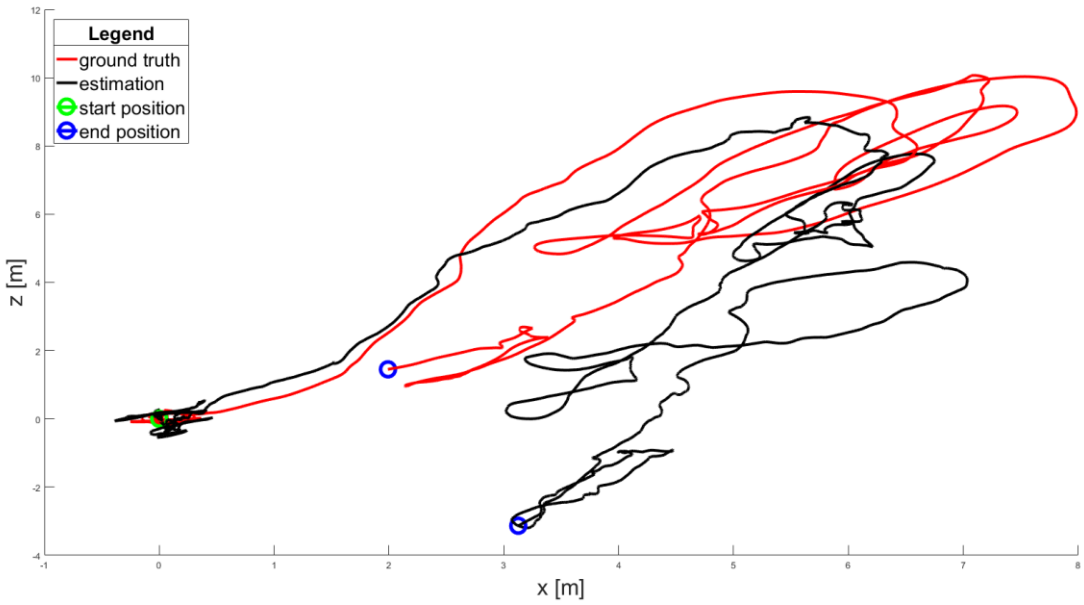

Figure 5.17 GT position vs. VO estimate in xz-plane for *MH_01*

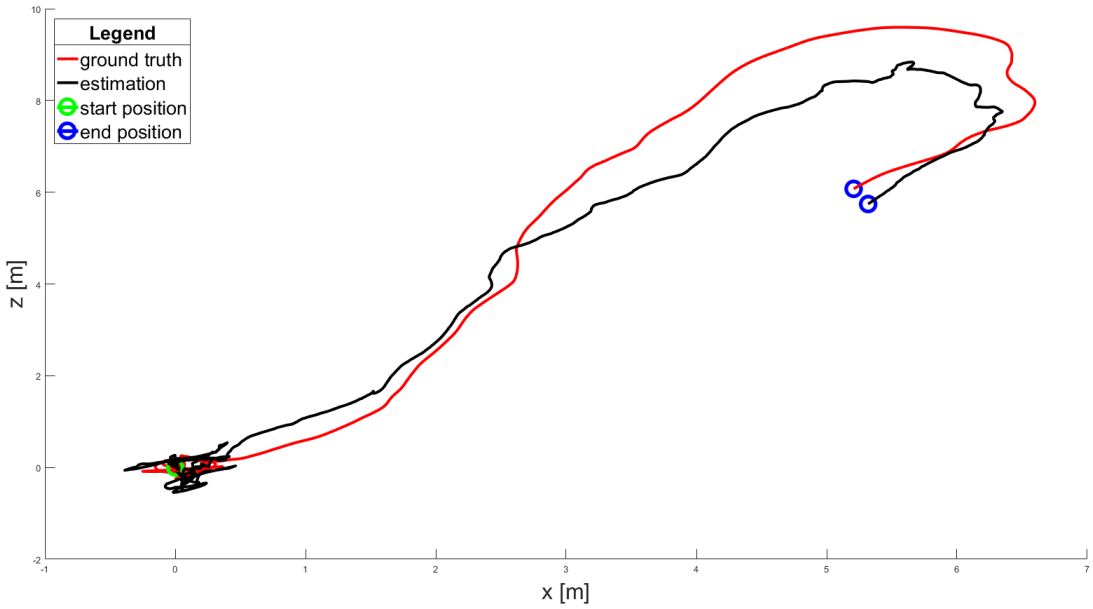

Figure 5.18 GT position vs. VO estimate sliced from frames 0-1550

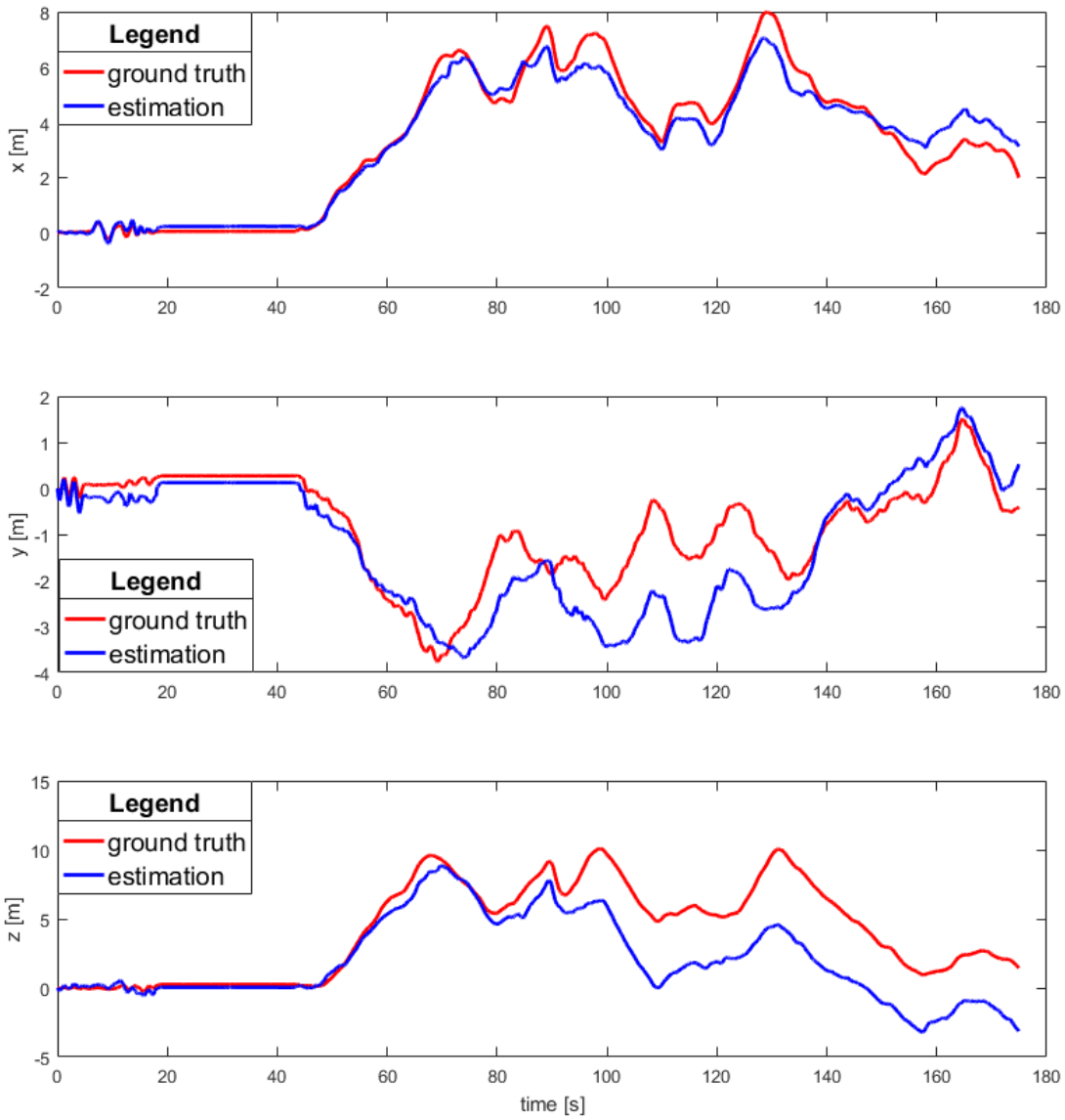

Figure 5.19 GT position vs. VO estimate in x, y and z separately for *MH_01*

5.1.2.2. Vicon Room *V2_01* Dataset Results

This dataset is relatively small compared to its counterpart. Total number of processed frames is 2000 over a 36.5-meter total distance. It is critical to note that the RMS error in x-axis is calculated as 1.49 m while the RMS error in y- and z-axes are 0.8409 m and 0.7918 m, respectively. Maximum errors are 2.6855 m, 2.1.7881 m and 1.7961 m for x-, y-, and z-axes, respectively. Similarly, the 3D RMS error is calculated as 1.8852 m and 3D maximum error is calculated as 3.6926 m which correspond to 5.16% and

10.11% of the covered distance, respectively. Comparing the drifts in the graph with the dataset images, it can be observed that the full turns around the z-axis worsen the estimates. After the full turns, the orientation of the path drifts due to the large errors if compared to the *MH_01* dataset.

Figure 5.20 illustrates VO estimates vs. GT data in xz-plane for *V2_01* dataset. As can be inferred from the figure, the major source of error is in x-direction due to straight and full angle turns of the vehicle. Figure 5.21 presents the sliced version of the previous figure for better visualization. It is observed that the estimated trajectories are very similar despite the drift. Figure 5.22 shows GT vs. VO estimates in xy-plane. It can be inferred that final pose error in x-axis is twice as large as the error in y-axis. On the other hand, Figure 5.23 demonstrates that y-axis errors are large in the middle of the run, but they do decrease to minimum in the end.

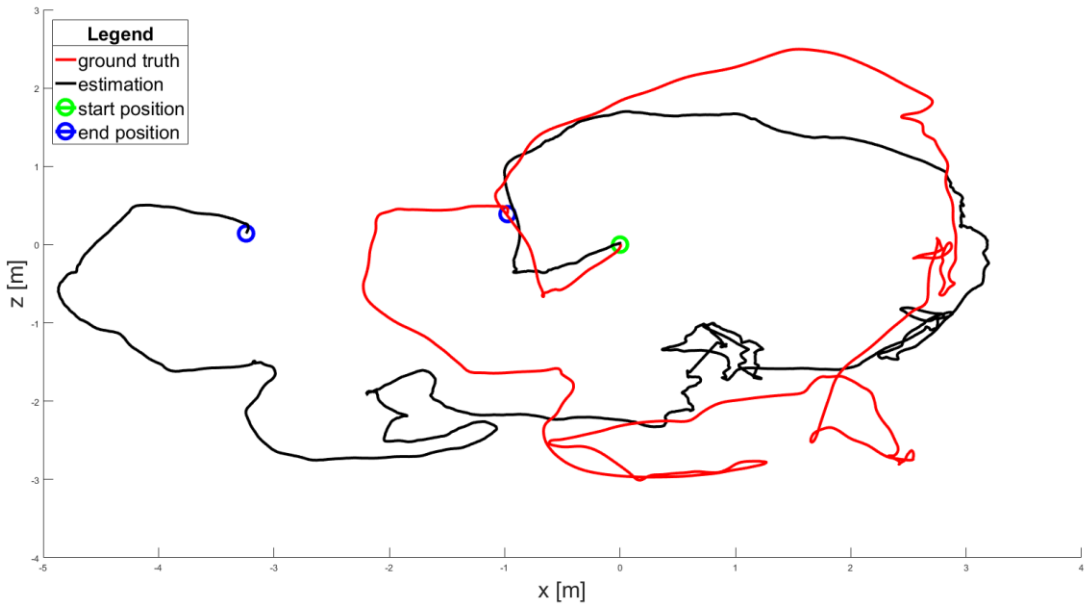

Figure 5.20 GT position vs. VO estimate in xz-plane for *V2_01*

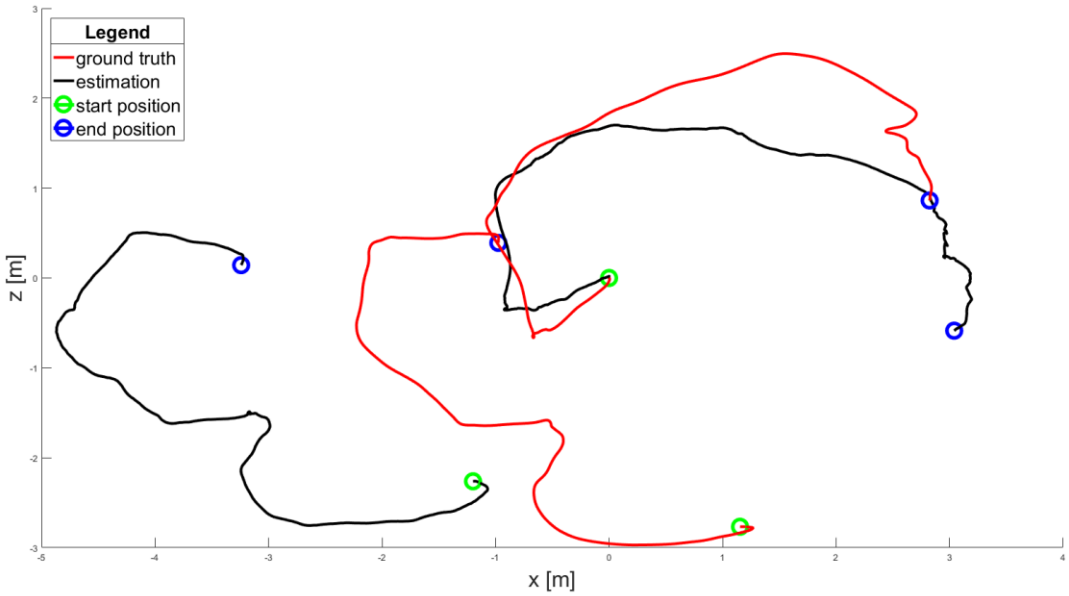

Figure 5.21 GT position vs. VO estimate sliced from frames 0-500, 1550-2000

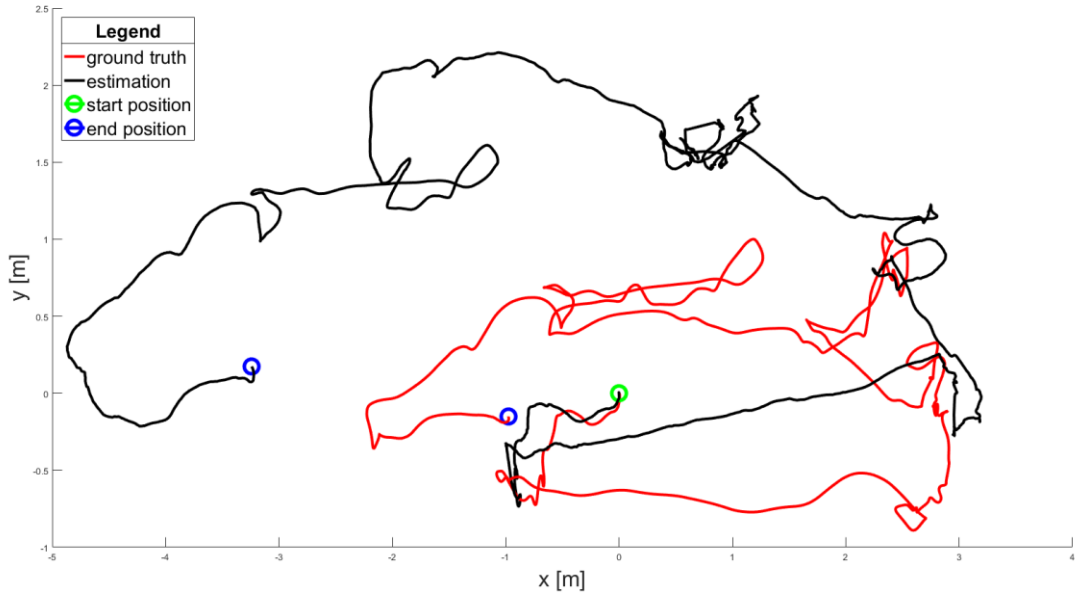

Figure 5.22 GT position vs. VO estimate in xy-plane for *V2_01*

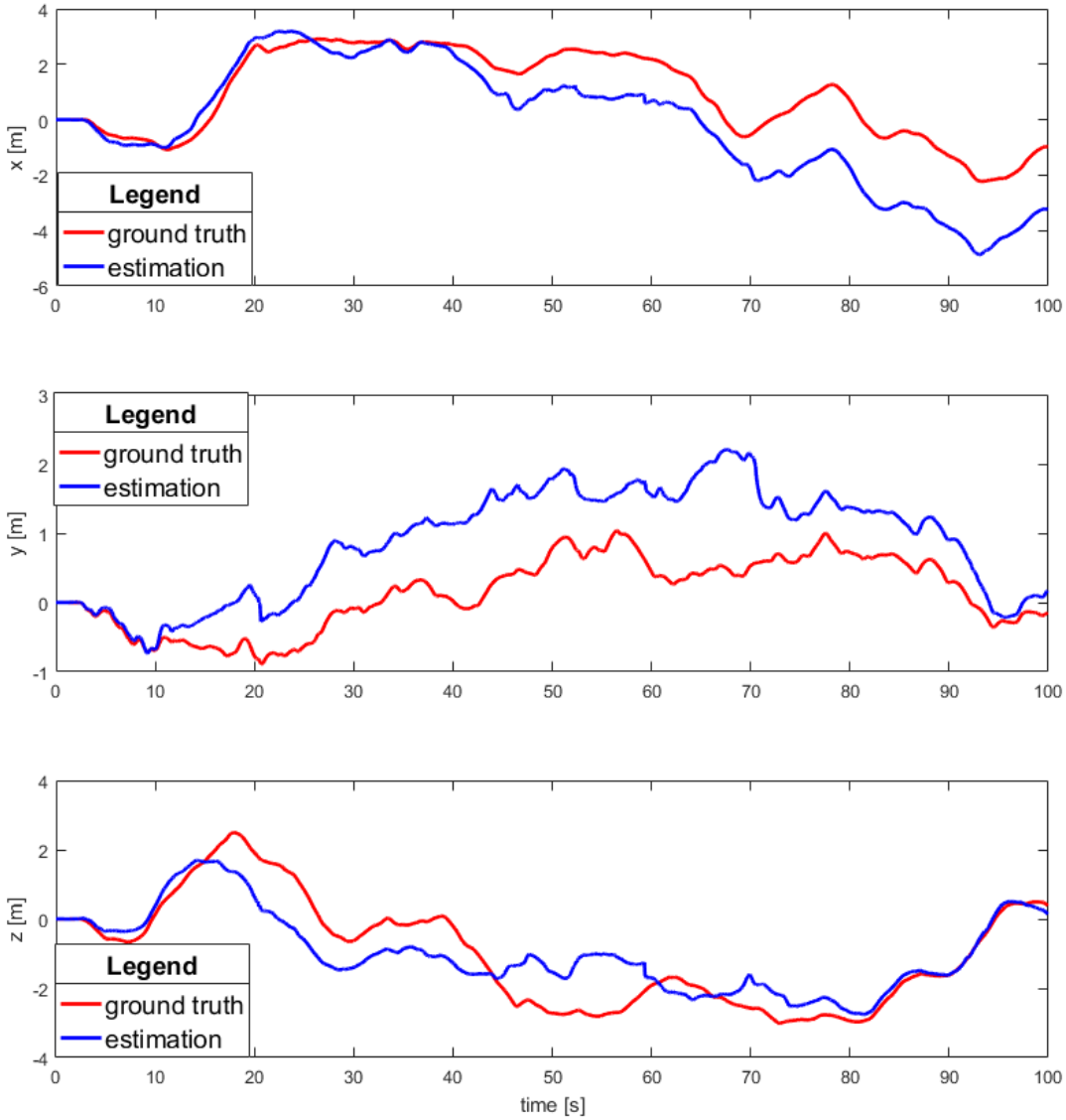

Figure 5.23 GT position vs. VO estimate in x, y and z separately for *V2_01*

5.1.3. Visual-Inertial Odometry Implementations

In this section, the results of VIO implementation are presented. Unlike the complicated filtering processes, a very easy and handful sensor fusion is applied which leads to slight improvements over visual-only estimates when the noise is small. Moreover, pose estimation results are fused via EKF using empirical covariance

matrices for comparison. Figure 5.24 and Figure 5.25 present the comparison of position estimation algorithms implemented on *MH_01* and *V2_01* datasets, respectively. In the figures, red lines represent the ground truth; blue lines represent the visual-only estimates; black dashed lines represent the EKF estimates and yellow dashed lines are representing the proposed filter results. Table 5.3 summarizes the results with the RMS, maximum and percentage errors for both datasets.

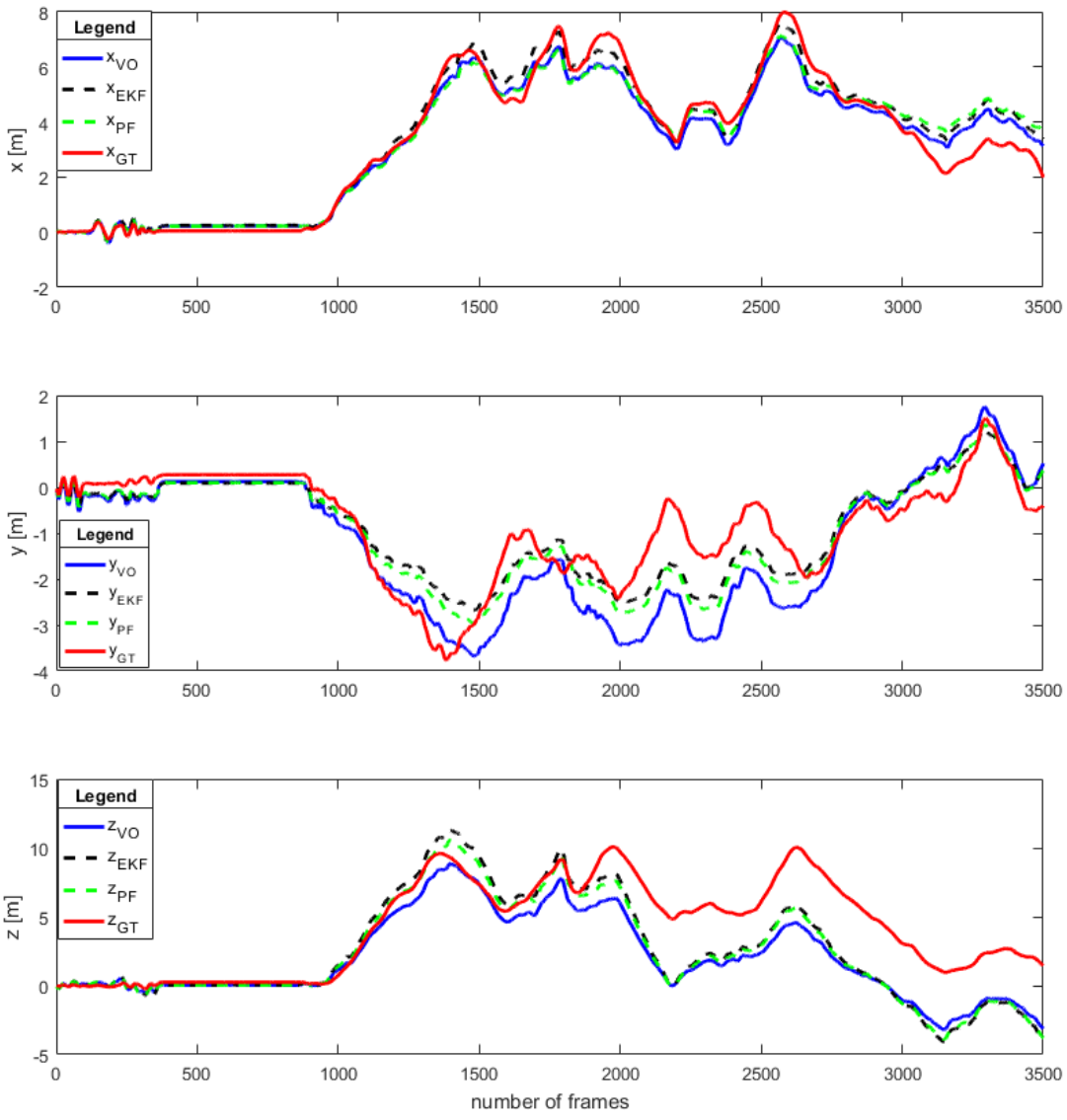

Figure 5.24 Position estimate comparison of implementations for *MH_01*

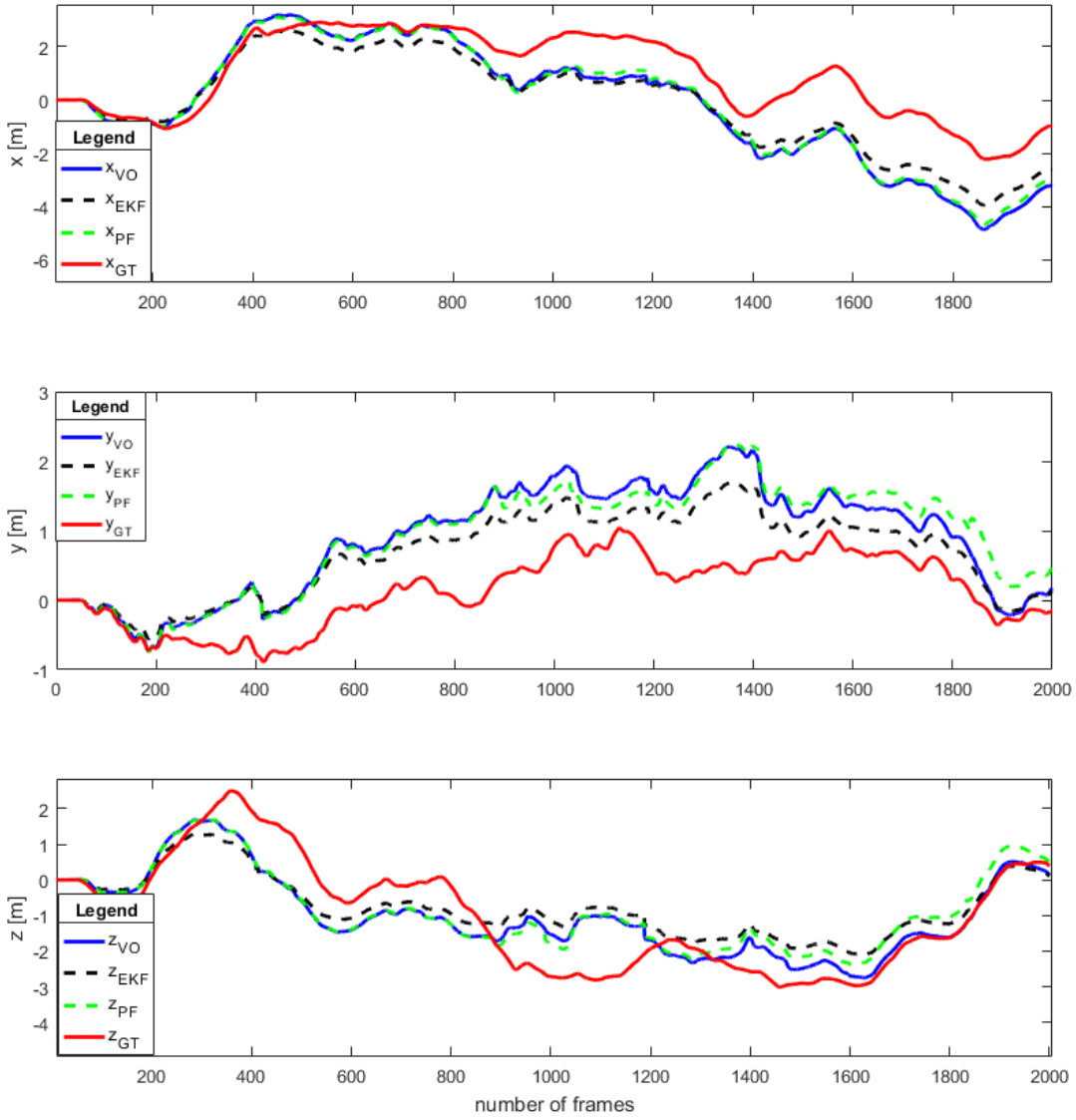

Figure 5.25 Position estimate comparison of implementations for *V2_01*

Table 5.3 VO results from different datasets

Dataset	Error	Visual only	EKF	Proposed Filter
Machine Hall *MH_01* dataset	RMS (max) error in x [m]	0.52 (1.27)	0.52 (1.45)	0.63 (1.73)
	RMS (max) error in y [m]	0.83 (2.02)	0.55 (1.4)	0.58 (1.53)
	RMS (max) error in z [m]	2.92 (5.59)	2.75 (5.46)	2.77 (5.28)
	RMS (max) error in 3D [m]	3.08 (6.08)	2.85 (5.81)	2.90 (5.74)
	RMS (max) error in 3D [%]	4 (7.9)	3.7 (7.55)	3.77 (7.45)
Vicon Room *V2_01* dataset	RMS (max) error in x [m]	1.49 (2.68)	1.29 (2.14)	1.42 (2.55)
	RMS (max) error in y [m]	0.84 (1.79)	0.6 (1.27)	0.85 (1.84)
	RMS (max) error in z [m]	0.79 (1.79)	0.93 (2.03)	0.85 (1.82)
	RMS (max) error in 3D [m]	1.88 (3.69)	1.7 (3.21)	1.86 (3.66)
	RMS (max) error in 3D [%]	5.17 (10.12)	4.65 (8.81)	5.10 (9.96)

5.2. Discussion

The results in Table 5.1 show that orientation, velocity and position can be estimated using only inertial sensors; however, sensor bias and unknown gravity direction introduce errors in the estimates. It is because small errors in orientation estimate produce extremely high erroneous acceleration and gravity measurements. Since gyroscope itself also drifts in long-term, it is not favorable to use only inertial sensors for long-term positioning systems, but they can be utilized in combination with VO because in short term, accelerometer and gyroscope give sufficient accuracy in

estimating the orientation and velocity, which can then be used in predicting the position for sensor fusion.

For the IMU-only orientation estimate, Madgwick's complementary filter is utilized and improved to cope with the drift. Since the motion and the change of the orientation is mostly in z-axis, a significant drift is observed in Madgwick's filter. This drift is eliminated by fitting a line between stationary periods which are detected by filtering accelerometer signals as Figure 5.1 and Figure 5.2 show. Z-axis orientation error decreases dramatically after using a first order line fitting for both datasets. It is important to note that the MAV is stationary between the 20^{th} and 40^{th} seconds in *MH_01* while only first and last three seconds of the movement is stationary in *V2_01*. Therefore, different line fitting periods are used for different datasets. Results show that RMS orientation error is still large in z-axis whereas relatively small errors are present in other two axes.

Velocity estimate is used as a pre-processing step to position estimates. Since integration introduces drift due to the integration constants, high-pass filter corrects the estimates. Average velocities of the datasets are given as 0.44 m/s for *MH_01* and 0.33 m/s for *V2_01* [84] and RMS 3D velocity errors are found as 0.5 m/s and 0.31 m/s, respectively. It is observed from Figure 5.9 and Figure 5.10 that the shape of the estimation signal is very close to the shape of the ground truth, but the error magnitude is large due to the drift.

Inertial position estimate is done by integrating the filtered and drift eliminated velocity vector. Position estimates in the direction of the gravity, z-axis in the world frame, are quite accurate compared to the other two axes. When the motion along an axis is large, estimates become more uncorrelated compared to the ground truth. One can observe that the shapes of the signals in the z-axis of the inertial position estimates are very close to the ground truth except the drift. However, there are obvious diffractions in x- and y-axes for both datasets.

For the VO implementation, only 2 datasets are processed from EuRoc MAV datasets in order to avoid long computation time. Since the VO algorithm is developed for

offline processing, algorithm is not optimized for fast operations. Therefore, pose estimation between two successive frames takes 11.16 seconds on average for *MH_01* dataset while 2.15 seconds is spent for *V2_01*. There are various reasons affecting the computation time. One of the most crucial reasons is that all images in the datasets are taken into large files before the implementation of the main algorithm, which uses a lot of RAM in the computer's memory. Since *MH_01* dataset has 1.75 times more images than *V2_01*, more memory is used to store variables, leaving a limited memory for calculations. The effects of memory can be seen in computation times of disparity maps which is the only part that is not affected by the number of tracked. These computation times are 0.28 seconds for *V2_01* compared to 0.35 seconds for *MH_01*. However, main reason for long execution time is the number of tracked features. The average number of tracked features is 821 for *MH_01* and 333 for *V2_01*. Figure 5.26 illustrates the correlation between the computation times of different parts of the VO algorithm and the number of features tracked between frames. Computation times are multiplied by a constant shown in the legend of the figure in order to visualize the correlation and bring the two data to a comparable scale. It can be seen from the figure that the processing time of costly outlier rejection is multiplied by a small scaling number and shown in black. Although the number of tracked features is high when the MAV is stationary, running time for outlier rejection is very small, which means that the features are located and matched accurately and the need for outlier rejection is minimum. However, it can be easily seen that around the 1000^{th} and 3000^{th} frames, computation duration increases rapidly as the number of tracked features increases. Likewise, KLT and optimization algorithms are also affected by the number of features as expected. However, the computation duration of dense disparity map is not affected from the number of features. Figure 5.27 illustrates the same correlation with constant multipliers for *V2_01* dataset. It is notable that the execution time of the minimization of the image reprojection error in motion estimation algorithm increases exponentially with the number of tracked features. This is because of the computational complexity of the Levenberg-Marquardt algorithm which is $O(n^3)$. This is also the main reason of the difference between the running times of the datasets *MH_01* and *V2_01*. A twice-larger dataset takes eight times longer computational time.

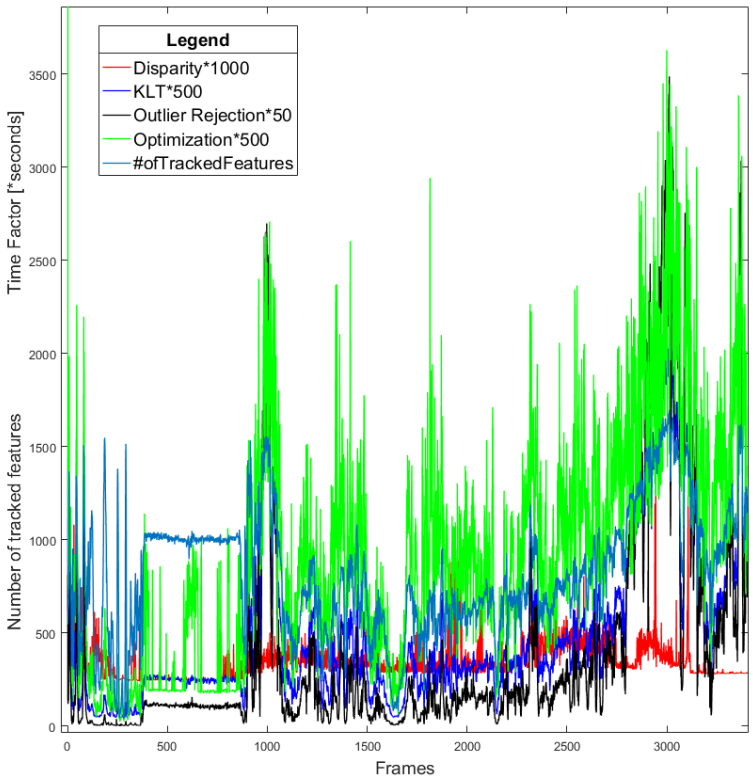

Figure 5.26 Computation times and number of tracked features for *MH_01*

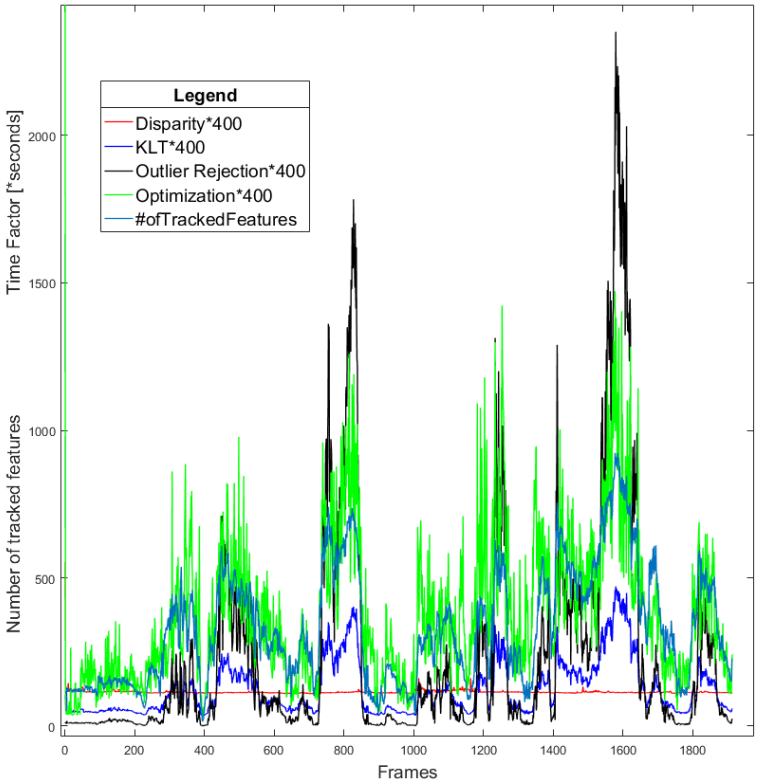

Figure 5.27 Computation times and number of tracked features for *V2_01*

Visual-only position estimates are affected by various factors. The most influential factors are noise, vibration of the MAV, illumination and sudden full turns around an axis. The main reasons of the errors in *MH_01* are vibrations and illumination while of the errors in *VH_01* are full rotations and featureless scenes.

Both proposed filter and EKF for sensor fusion offer slightly better overall results compared to the visual-only results. Table 5.3 shows that using a sensor fusion system, it is possible to succeed less than five percent errors. VO estimates in Machine Hall *MH_01* dataset are improved by 0.3 and 0.23 percent using respectively EKF and proposed filter. In *V2_01*, visual-only position estimates are enhanced by 0.52 percent with EKF and 0.07 percent with the proposed filter. Since the proposed filter identifies the blurry images, delete them and enlarge the window sizes, it gives better results when there is a vibration in the images. Vibratory motion of the camera forms noisy feature points but the locations of the features stay very close. Taking advantage of this characteristic proposed filter provides better improvements in *MH_01* dataset. However, this procedure is not thriving when there are fast orientation changes as in *V2_01*. Since high angular velocity introduces noise and blur, number of deleted blurry images increases. Accordingly, the uncertainty and the distance between tracked features increase, causing the proposed filter to converge to visual-only results. On the contrary, EKF is more resistant to orientation changes as it also utilizes inertial position estimate which is robust in attitude change. Thus EKF gives the best estimate in *V2_01* dataset, improving the visual-only RMS error 0.18 meters.

While position ground truth is taken by high-end laser tracking and motion capture systems in EuRoC MAV dataset, orientation ground truth is obtained only by using 6-axis IMU. Since the visual-only orientation estimates are prone to drift and not reliable, attitude estimate of the MAV is also done using only IMU readings. Thus, inertial orientation estimates are directly substituted for VO orientation estimates to form a 6D position estimate for each frame location.

5.3. Closure

Implementation results using different algorithms and different datasets are presented in this chapter. First, inertial pose estimations are given. It is shown that orientation estimates using IMU readings are very reliable, as it offers less than 20 degrees 3D RMS error. Thus, inertial orientation estimates are taken as the final estimates of the pose. For translational motion estimation, inertial pose estimates are fused to VO results using both EKF and proposed filter. Results show that proposed filter offer minor improvements over visual-only results while EKF introduces slightly more enhancements. Final remarks, conclusion and future works are given in the following chapter.

CHAPTER 6

CONCLUSIONS

This thesis study investigated the camera trajectory estimation for indoor robot odometry using IMU and stereo images. Since GPS is a great solution for outdoor environments, indoor visual inertial odometry is researched. The study first focused on the inertial pose estimation. Second, pose estimation using stereo images are done. Finally, sensor fusion is implemented to attain improved estimates. In other words, the aim was to develop and implement an algorithm for stereo visual-inertial odometry. For this goal, three basic criteria were determined. These are (1) to employ low-cost sensor systems, (2) to obtain less than five percent error, and (3) to create a robust algorithm working in harsh conditions such as noise. A custom sensor system is not constructed because an already available dataset is used in the study. Nevertheless, a very low cost camera and a medium priced IMU is utilized in the dataset [84], which partly satisfies the criterion. Using noisy stereo images and inertial reading less than five percent error is achieved in position estimates. Orientations estimates are shown to be less than 20 degrees.

Inertial measurements are processed in three stages: First, orientation is estimated by implementing Madgwick's gradient-descent based explicit complementary filter. An improvement is suggested to eliminated the drift in z-axis. Second, velocity is estimated by integrating the accelerometer measurements and high-pass filtering the results. Zero velocity positions are used to eliminate the drift. Lastly, velocity is

integrated to obtain position estimates. Again, drift in position estimates is removed by further filtering and line fitting between zero velocity positions. For *MH_01* dataset 15.54 degrees, 0.5 m/s and 6.9 m RMS errors are obtained respectively for orientation, velocity and position as demonstrated in Table 5.1. Algorithm implemented on *V2_01* dataset resulted in 18.4 degrees RMS error for attitude, 0.31 m/s RMS error for velocity and 3.66 m RMS error for position estimates.

Visual-only estimate is only used for translational component of the 6D pose estimate. As explained in Chapter 3.4.1, successive stereo image pairs are captured and the images are undistorted and rectified for further processing. Then, salient corner features are detected and extracted features in both stereo and sequential images. Then, motion is estimated by minimizing the image reprojection error between k^{th} 3D feature and $(k+1)^{th}$ 2D point correspondence after finding the inliers' set. Results in Table 5.3 show that 3.08 m and 4% 3D RMS error is obtained in *MH_01* dataset while 1.88 m and 5.17% 3D RMS error is obtained in *V2_01* dataset.

An original filter is proposed to fuse the inertial and visual data. EKF is also implemented to the pose estimates of IMU and VO for comparison. Results are compared in Table 5.3. It is shown that proposed filter improves visual-only estimates. While proposed filter enhances visual-only estimates as much as EKF in *MH_01*, it is observed that *V2_01* dataset gives only a minor improvement over VO estimate. Chapter 5.2 discusses the performance of the algorithms.

6.1. Future Work

First of all, proposed fusion filter can be improved. Gyroscope aided feature tracking can be added and number of tracked features can be optimized to have a better fusion. It will have a great impact on the camera trajectory estimation results as the MAV vibrates during its motion which can be compensated easily using high rate accelerometer readings. Kalman filter can also be improved by improving the update and measurement error covariance matrices.

Utilizing a magnetometer and barometer together would increase the accuracy of the estimates since error in z-direction is relatively high as indicated by experimental

results. The use of a magnetometer and a barometer would enable system to precisely know its altitude. Moreover, altitude data could be fused with VO to have better position estimates.

It is important to test the algorithms for different indoor vehicles other than micro air vehicles. Due to the challenging motion characteristics of the MAV, implementation results are not completely satisfying, yet sufficient. It is predicted that better results can be obtained with wheeled robotic platforms. Since the motion is usually planar in indoors for wheeled vehicles, change in tilt and roll angle is very small, which will result in increased accuracy in position and orientation estimates. Visual-inertial odometry algorithm should also be tested using real world data that will be taken specifically for the study. This has various advantages in exchange for the cost of reliable data gathering platform. First of all, application specific dataset can be constructed making long-range runs in different indoor environments like malls, large and small rooms, hospitals, warehouses with different circumstances such as illumination change, noise, etc. For instance, EuRoC MAV dataset has a movement only in an 8 m × 8.4 m × 4 m room. There is no room change, no corridor movements in the dataset which should have also been considered for indoor positioning and navigation purposes. It is also important to use different kinds of sensors while creating a dataset. Using 2 different stereo camera system and 5 to 10 different low- and medium-cost IMUs would enable the researchers to evaluate different sensors in different specifications. Since visual-inertial odometry is a future technology that will be used in commercial applications, promising results should have been taken with low-cost sensors as well as high-end sensors.

There is also a room for developing adaptive algorithms for changing conditions in the environment. Using IMU readings noisy movements can be detected and VO algorithm can be changed accordingly. Moreover, room scale can be calculated from stereo data. The algorithm can be adapted so that for long range data a monocular scheme can be used since stereo odometry converges to monocular odometry if the features are far away. Likewise, baseline length can be changed on the fly so that depth estimates of distant objects can be identified easily. What’s more, different feature

detection, feature extraction, feature matching, outlier rejection, motion estimation and optimization algorithms for VO can be combined in a single package and can be compared both for accuracy and efficiency.

Since the electronics and information technologies are developing so rapidly, camera trajectory estimation algorithm is implemented offline, thinking that the technology eventually reach the maturity for online visual-inertial odometry in a short period of time. For the future works, the study can be implemented in real time using parallel processing systems such as FPGAs and high-end DSPs.

REFERENCES

[1] M. Kumru, *Navigation and control of an unmanned sea surface vehicle,* Doctoral Dissertation, Middle East Technical University, 2015.

[2] J. Schipperijn, J. Kerr, S. Duncan, T. Madsen, C. D. Klinker and J. Troelsen, "Dynamic accuracy of GPS receivers for use in health research: a novel method to assess GPS accuracy in real-world settings," *Emerging Technologies to Promote and Evaluate Physical Activity,* vol. 23, 2014.

[3] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics & Automation Magazine,* vol. 18, no. 4, pp. 80-92, 2011.

[4] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part II: Matching, robustness, optimization, and applications," *IEEE Robotics & Automation Magazine,* vol. 19, no. 2, pp. 78-90, 2012.

[5] M. D. Warren, *Long-Range Stereo Visual Odometry for Unmanned Aerial Vehicles,* Doctoral Dissertation, Queensland University of Technology, 2015.

[6] M. Hagele, "Robots Conquer the World [Turning Point]," *IEEE Robotics & Automation Magazine,* vol. 23, no. 1, pp. 118-120, 2016.

[7] B. S. Cho, W. S. Moon, W. J. Seo and K. R. Baek, "A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding," *Journal of mechanical science and technology,* vol. 25, no. 11, pp. 2907-2917, 2011.

[8] "Unmanned Systems," [Online]. Available: https://insitu.com/information-delivery/unmanned-systems. [Accessed 7 9 2016].

[9] N. Snavely, S. M. Seitz and R. Szeliski, "Photo tourism: exploring photo collections in 3D," *ACM transactions on graphics (TOG),* vol. 25, no. 3, pp. 835-846, 2006.

[10] "Photo Tourism, Exploring photo collections in 3D," University of Washington, [Online]. Available: http://phototour.cs.washington.edu/. [Accessed 9 7 2016].

[11] Y. M. Wei, L. Kang and B. Yang, "Applications of structure from motion: a survey," *Journal of Zhejiang University SCIENCE C,* vol. 14, no. 7, pp. 486-494, 2013.

[12] J. Civera, "Real-time EKF-Based Structure from Motion," PhD Thesis, Universidad de Zaragoza, 2009.

[13] R. Hartley and A. Zisserman, Multiple view geometry, Cambridge university press, 2003.

[14] W. Förstner, "A feature based correspondence algorithm for image matching," *International Archives of Photogrammetry and Remote Sensing,* vol. 26, no. 3, pp. 150-166, 1986.

[15] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions,* vol. 6, pp. 679-698, 1986.

[16] J. Shi and C. Tomasi, "Good features to track," *Proceedings of IEEE lnt'l Conference on Computer Vision and Pattem Recognition,* pp. 593-600, 1994.

[17] C. Harris and M. Stephens, "A combined corner and edge detector," *Alvey vision conference,* vol. 15, 1988.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision,* vol. 60, no. 2, pp. 91-110, 2004.

[19] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision–ECCV,* pp. 430-443, 2006.

[20] S. M. Smith and M. J. Brady, "SUSAN—a new approach to low level image processing," *International journal of computer vision,* vol. 23, no. 1, pp. 45-78, 1997.

[21] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *International conference on computer vision,* pp. 2564-2571, 2011.

[22] S. Leutenegger, M. Y. Chli and R. Siegwart, "BRISK: Binary robust invariant scalable keypoints," *IEEE International Conference on Computer Vision (ICCV),* pp. 2548-2555, 2011.

[23] H. Bay, T. Tuytelaars and L. Van Gool, "Surf: Speeded up robust features," *European conference on computer vision,* pp. 404-417, 2006.

[24] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm," *Communications of the ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[25] B. Triggs, P. F. McLauchlan, R. I. Hartley and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," *International workshop on vision algorithms,* vol. Springer, pp. 298-372, 1999.

[26] D. Nistér, O. Naroditsky and J. Bergen, "Visual odometry," *Computer Vision and Pattern Recognition (CVPR),* vol. 1, no. 1, pp. 652-659, 2004.

[27] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1980.

[28] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," in *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, MA Fischler and O. Firschein, 1987, pp. 61-62.

[29] C. F. Olson, L. H. Matthies, H. Schoppers and M. W. Maimone, "Robust stereo ego-motion for long distance navigation," *Computer Vision and Pattern Recognition,* vol. 2, pp. 453-458, 2000.

[30] C. F. Olson, L. H. Matthies, M. Schoppers and M. W. Maimone, "Rover navigation using stereo ego-motion," *Robotics and Autonomous Systems,* vol. 43, no. 4, pp. 215-229, 2003.

[31] M. Maimone, Y. Cheng and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics,* vol. 24, no. 3, pp. 169-186, 2007.

[32] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE robotics & automation magazine,* vol. 13, no. 2, pp. 99-110, 2006.

[33] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics & Automation Magazine,* vol. 13, no. 3, pp. 108-117, 2006.

[34] J. Fuentes-Pacheco, J. Ruiz-Ascencio and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review,* vol. 43, no. 1, pp. 55-81, 2015.

[35] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE transactions on pattern analysis and machine intelligence, ,* vol. 26, no. 6, pp. 756-770, 2004.

[36] P. Corke, D. Strelow and S. Singh, "Omnidirectional visual odometry for a planetary rover," *Intelligent Robots and Systems,* vol. 4, pp. 4007-4012, 2004.

[37] J. P. Tardif, Y. Pavlidis and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," *IEEE/RSJ International Conference on Intelligent Robots and System,* pp. 2531-2538, 2008.

[38] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser and P. Sayd, "Real time localization and 3d reconstruction," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06),* vol. 1, pp. 363-370, 2006.

[39] D. Scaramuzza, F. Fraundorfer and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac," *Robotics and Automation, ICRA'09*, pp. 4293-4299, 2009.

[40] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 1015-1056, 2008.

[41] C. Forster, M. Pizzoli and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15-22, 2014.

[42] Y. Cheng, M. Maimone and L. Matthies, "Visual odometry on the Mars exploration rovers," *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 903-910, 2005.

[43] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3946-3952, 2008.

[44] B. Kitt, A. Geiger and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," *Intelligent Vehicles Symposium*, pp. 486-492, 2010.

[45] K. Konolige, Agrawal, M. and J. Sola, "Large-scale visual odometry for rough terrain," in *Robotics research*, Berlin Heidelberg, Springer, 2010, pp. 201-212.

[46] M. Agrawal, K. Konolige and M. R. Blas, "Censure: Center surround extremas for realtime feature detection and matching," in *European Conference on Computer Vision*, Berlin Heidelberg, Springer, 2008, pp. 102-105.

[47] K. Schmid and H. Hirschmüller, "Stereo vision and IMU based real-time ego-motion and depth image computation on a handheld device," *Robotics and Automation (ICRA)*, pp. 4671-4678, 2013.

[48] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328-341, 2008.

[49] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale and R. Siegwart, "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 431-437, 2014.

[50] H. Badino, A. Yamamoto and T. Kanade, "Visual odometry by multi-frame feature integration," *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 222-229, 2013.

[51] M. Achtelik, S. Weiss and R. Siegwart, "Onboard IMU and monocular vision based control for MAVs in unknown in-and outdoor environments," *IEEE international conference on Robotics and automation (ICRA)*, pp. 3056-3063, 2011.

[52] G. Panahandeh, D. Zachariah and M. Jansson, "Exploiting ground plane constraints for visual-inertial navigation," *Position Location and Navigation Symposium (PLANS)*, pp. 527-534, 2012.

[53] A. Martinelli, "Closed-form solution of visual-inertial structure from motion," *International journal of computer vision*, vol. 106, no. 2, pp. 138-152, 2014.

[54] A. Haro, K. Mori, T. Capin and S. Wilkinson, "Mobile camera-based user interaction," *International Workshop on Human-Computer Interaction*, vol. Springer Berlin Heidelberg, pp. 79-89, 2005.

[55] S. B. Goldberg and L. Matthies, "Stereo and IMU assisted visual odometry on an OMAP3530 for small robots," *CVPR 2011 WORKSHOPS*, pp. 169-176, 2011.

[56] T. Schöps, J. Engel and D. Cremers, "Semi-dense visual odometry for AR on a smartphone," *Mixed and Augmented Reality (ISMAR)*, pp. 145-150, 2014.

[57] M. Li, B. H. Kim and A. I. Mourikis, "Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera," *Robotics and Automation (ICRA)*, pp. 4712-4719, 2013.

[58] S. Tomažič and I. Škrjanc, "Fusion of visual odometry and inertial navigation system on a smartphone," *Computers in Industry*, vol. 74, pp. 119-134, 2015.

[59] J. Engel, J. Stückler and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," Technische Universitat München - Computer Vision Group, 15 07 2016. [Online]. Available: http://vision.in.tum.de/research/vslam/lsdslam. [Accessed 25 7 2016].

[60] D. Litwiller, "CCD vs. CMOS," *Photonics Spectra*, vol. 35, no. 1, pp. 154-158, 2001.

[61] L. Xinqiao, "CMOS Image Sensors Dynamic Range and SNR Enhancement via Statistical Signal," Doctoral Thesis,Department of Electrical Engineering, Stanford University, 2002.

[62] V. Ganapathi, C. Plagemann, D. Koller and S. Thrun, "Real time motion capture using a single time-of-flight camera," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[63] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," *International Symposium on Robotics Research (ISRR)* .

[64] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence* , vol. 22, no. 11, pp. 1330-1334, 2000.

[65] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab," 14 10 2015. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/. [Accessed 17 8 2016].

[66] H. P. Moravec, "Towards automatic visual obstacle avoidance," *Proc. Int. Joint Conference on AI*, pp. 584-586, 1977.

[67] 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*, vol. 32, no. 1, pp. 105-119, 2010.

[68] A. Alahi, R. Ortiz and P. Vandergheynst, "Freak: Fast retina keypoint," *Computer vision and pattern recognition*, pp. 510-517, 2012.

[69] J. Matas, O. Chum, M. Urban and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing*, vol. 22, no. 10, pp. 761-767, 2004.

[70] C. Tomasi and T. Kanade, "Detection and tracking of point features," School of Computer Science, Carnegie Mellon Univ., Pittsburgh, 1991.

[71] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *International Joint Conference on Artificial Intelligence*, pp. 674-679, 1981.

[72] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Springer*, Berlin Heidelberg, 2003.

[73] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137-154, 1992.

[74] L. Kneip, D. Scaramuzza and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," *Computer Vision and Pattern Recognition*, pp. 2969-2976, 2011.

[75] D. Nister, "Preemptive RANSAC for live structure and motion estimation," *Machine Vision and Applications*, vol. 16, no. 5, pp. 321-329, 2005.

[76] E. Olson, J. Leonard and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," *ICRA, IEEE International Conference on Robotics and Automation*, pp. 2262-2269, 2006.

[77] D. Titterton and J. L. Weston, Strapdown inertial navigation technology, IET, 2004.

[78] R. Mahony, T. Hamel and J. M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203-1218, 2008.

[79] S. Thrun, W. Burgard and D. Fox, Probabilistic robotics, MA: MIT Press Cambridge, 2005.

[80] S. O. Madgwick, A. J. Harrison and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," *IEEE International Conference on Rehabilitation Robotics*, pp. 1-7, 2011.

[81] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354-3361, 2012.

[82] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," *German Conference on Pattern Recognition*, no. Springer International Publishing., pp. 31-42, 2014.

[83] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti and P. Taddei, "Rawseeds ground truth collection systems for indoor self-localization and mapping," *Autonomous Robots*, vol. 27, no. 4, pp. 353-371, 2009.

[84] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. Achtelik and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, p. DOI: 10.1177/0278364915620033, 2016.

[85] H. Oleynikova, M. Burri, S. Lynen and R. Siegwart, "Real-time visual-inertial localization for aerial and ground robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3079-3085, 2015.

[86] S. Sırtkaya, "Visual-Inertial Sensor Fusion for 3D Urban Modeling," Ph.D. dissertation, Dept. of Electrical and Electronics Eng., Middle East Technical Univ., Ankara, 2013.

[87] N. D. D. &. B. S. Krombach, "Combining Feature-based and Direct Methods for Semi-dense Real-time Stereo Visual Odometry," *International Conference on Intelligent Autonomous Systems*, 2016.

[88] C. Fu, A. Carrio and P. Campoy, "Efficient visual odometry and mapping for unmanned aerial vehicle using ARM-based stereo vision pre-processing system," *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 957-962, 2015.

[89] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696, 2007.

[90] L. W. Kheng, "Camera Models and Imaging," 19 10 2012. [Online]. Available: https://www.comp.nus.edu.sg/~cs4243/lecture/camera.pdf. [Accessed 15 8 2016].

[91] o. d. team, "Camera Calibration," 10 11 2014. [Online]. Available: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_calibration/py_calibration.html. [Accessed 15 8 2016].

[92] ETH, "ASL Datasets," Autonomous System Lab, [Online]. Available: http://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets. [Accessed 5 8 2016].

[93] "Blackfly 1.3 MP Color USB3 Vision (Sony ICX445)," Point Grey, Inc, [Online]. Available: https://www.ptgrey.com/blackfly-13-mp-color-usb3-vision-sony-icx445. [Accessed 5 8 2016].

[94] "Stereo Calibration App," MathWorks, [Online]. Available: http://www.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html. [Accessed 5 8 2015].

APPENDIX A

CAMERA BASICS
AND
EPIPOLAR GEOMETRY

Camera modeling and calibration are to be reviewed for VO. In perspective projection, pinhole camera models are more favorable. In pinhole cameras, the intersection of light rays produces an image through the projection center [3]. In other words, the principle of pinhole camera is based on one small hole –pinhole– in a light-proof system in which the light enters from the pinhole to reflect a reversed image in the box on the other side. Thus, a pinhole camera has a single aperture, rather than lenses. As illustrated in Figure A.1, running through a small pinhole, light rays of an item produce an image in an inverted way.

To easily visualize and calculate the necessary parameters, image plane is taken between the pinhole plane and the 3D object as virtual image plane. The distance between virtual image plane and pinhole plane is as much as focal length in order to ensure the mirror image representation with same magnitude. This is shown in Figure A.1. Then, using triangle similarities, x and y coordinates of the 3D world point can be calculated by Eqns. (A.1) and (A.2). These are the simplest form of the perspective projection, which will be used throughout this study.

$$\frac{X}{Z} = \frac{x}{f} \rightarrow \; x = f\,\frac{X}{Z} \tag{A.1}$$

$$\frac{Y}{Z} = \frac{y}{f} \rightarrow \; y = f\,\frac{Y}{Z} \tag{A.2}$$

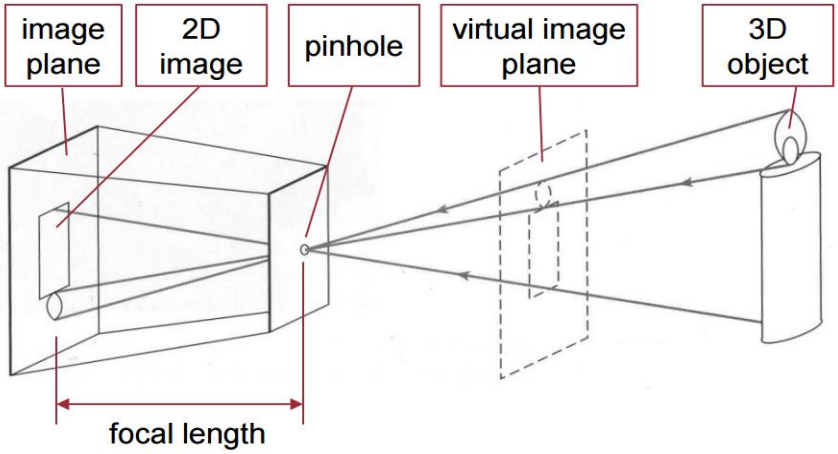

Figure A.1 Pinhole camera working principle [90]

The other vital aspect of the camera is the camera calibration. The aim in this sense is to gauge the intrinsic and extrinsic parameters from the camera systems accurately [3]. First of all, intrinsic parameters consist of the focal length (f_x, f_y), skew coefficient (s), and the principal point (c_x, c_y). In Figure A.2, for instance, the item is a 3-D object whereas the image is a 2-D inverted image. The hole is the focal point whereas the distance between the focal point and the inverted image is the focal length. Accordingly, the camera calibration matrix, **K**, can be shown as

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \tag{A.3}$$

Secondly, extrinsic parameters relate the 3D world point to camera image plane by means of rotation and translation components. For stereo cameras, a multi-camera system, the translation and orientation between two cameras can also be designated by

extrinsic parameters. Extrinsic camera parameters can be retrieved by calibrating the stereo system using a well-defined image pattern like checkerboard as in Figure A.3.

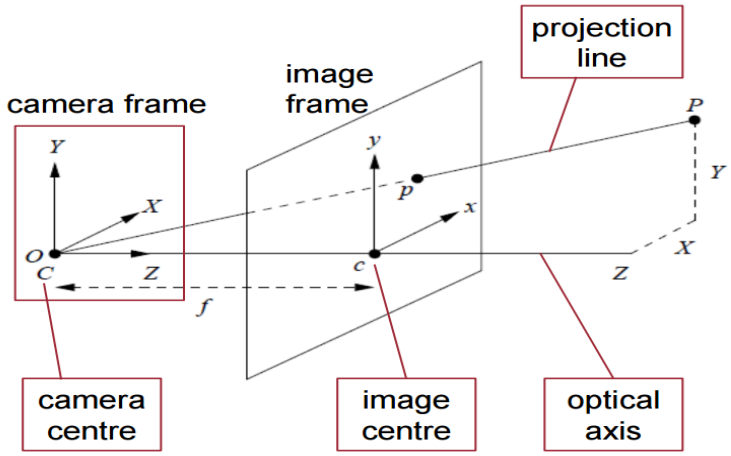

Figure A.2 Pinhole camera model [90]

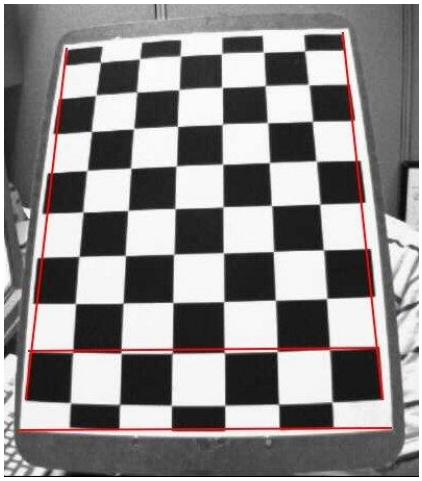

Figure A.3 Radial distortion and correction lines [91]

Cameras do not give a perfect image output due to distortions. Most common distortion types for pin hole camera systems are radial- and tangential distortions. Radial distortion makes straight lines seem curved which is called *barreling* effect. Barreling effect is much more dominant when moved away from the center of the image. Figure

A.3 shows the original grey scale image and how the lines should be in aa undistorted image in red lines.

Radial distortion can be formulated as

$$x_c = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad \text{(A.4a)}$$

$$y_c = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad \text{(A.4b)}$$

Because of the lens mount imperfections, image plane cannot be perfectly aligned with the lens resulting with a non-parallel configuration. This deficiency can be defined as tangential distortion. As a result of tangential distortion, part of the image can be seen closer than it is supposed to. Similarly, tangential distortion can be calculated as

$$x_c = x + (2p_1 xy + p_2(r^2 + 2x^2)) \quad \text{(A.5a)}$$

$$y_c = y + ((p_1(r^2 + 2y^2) + 2p_2 xy)) \quad \text{(A.5b)}$$

Note that there are a total of five parameters to define in order to model radial and tangential distortion:

$$Dist.Coef. = (k_1, k_2, k_3, p_1, p_2) \quad \text{(A.6)}$$

As mentioned before, camera calibration can be done using a checkerboard pattern. Corners of the checkerboard black squares are detected with subpixel accuracy. Since the location of the detected corners are known both in the image and in the real world, the mathematical problem of the camera calibration parameters and the distortion coefficients can be solved accordingly. Detailed explanation of the used functions from Utilized functions from MATLAB computer vision library for the camera calibration and undistortion can be found in the Appendix F.

In order to match corresponding points in a stereo image pair, epipolar geometry is utilized. Epipolar geometry, as shown in Figure A.4, uses the intersection of the stereo image planes of which stereo baseline constitutes the pencil of planes. Formulation in

this section is mostly gathered up from the book *Multiple View Geometry in Computer Vision* [13].

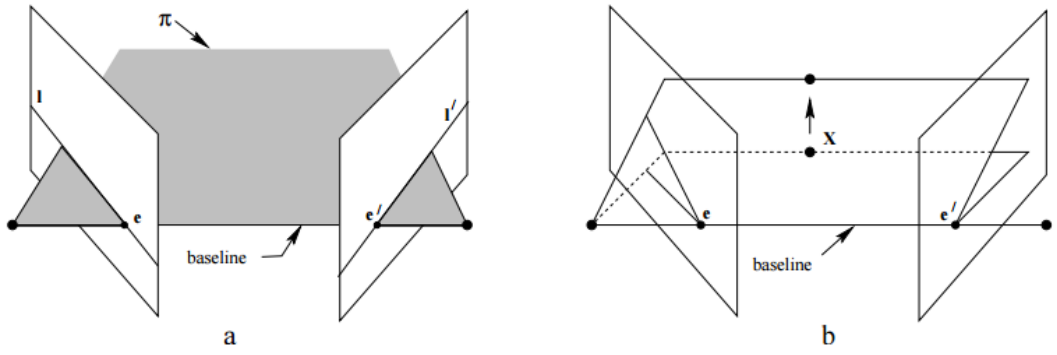

Figure A.4 Epipolar geometry for stereo camera setup [13]

In Figure A.4;

- Baseline is defined as the line joining the camera centers.
- *e* and *e'* represents the epipoles, the intersection of the baseline with the image plane.
- Epipolar plane, π, is the plane consisting the baseline.
- Epipolar lines, *l* and *l'*, are the intersection of the epipolar plane and the image plane.

3×3 fundamental matrix relates left image feature points, **x**, with the corresponding right image features, **x'**, using epipolar geometry. **Fx** defines a line, *l'*, in which the corresponding feature point, **x',** must lie. Therefore, all feature matches can be described using fundamental matrix as

$$\mathbf{x'^T F x = 0} \tag{A.7}$$

Fundamental matrix is of rank 2 and has 7-DOF, i.e. can be determined using seven point correspondences. A specialized version of **F** is called essential matrix which also relates stereo image pairs in case of calibrated cameras where the intrinsic parameters

are known. Thus, the 3×3 essential matrix, **E**, is only dependent to the extrinsic parameters which are rotation and translation components:

$$\hat{\mathbf{x}}'^{\mathbf{T}}\,\mathbf{E}\,\hat{\mathbf{x}} = \hat{\mathbf{x}}'^{\mathbf{T}}\,\mathbf{R}\,\mathbf{t_{ss}}\,\hat{\mathbf{x}} = \mathbf{0} \tag{A.8}$$

where $\hat{\mathbf{x}}$ is the normalized feature coordinate and $\mathbf{t_{ss}}$ is the skew-symmetric form of the translation vector. Essential matrix is also of rank 2 and has both a left- and right null space. **K** is being the intrinsic camera parameters, essential and fundamental matrices can be related as

$$\mathbf{E} = \mathbf{K}'^{\mathbf{T}}\,\mathbf{F}\,\mathbf{K} \tag{A.9}$$

Essential matrix can be decomposed to obtain rotation matrix and translation vector using *Singular Value Decomposition (SVD)* method. SVD of **E** can be written as

$$\mathbf{E} = \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^{\mathbf{T}} \tag{A.10}$$

where **U** and **V** are 3×3 orthogonal matrices:

$$\boldsymbol{\Sigma} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.11a}$$

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.11b}$$

Here, the diagonal entries of $\boldsymbol{\Sigma}$ are the singular values of **E**, which and must consist of two identical and one zero value. Finally, translation and rotation matrices can be given as

$$\mathbf{t_{ss}} = \mathbf{U}\,\mathbf{W}\,\boldsymbol{\Sigma}\,\mathbf{U}^{\mathbf{T}} \tag{A.12a}$$

$$\mathbf{R} = \mathbf{U}\,\mathbf{W}^{-\mathbf{1}}\,\mathbf{V}^{\mathbf{T}} \tag{A.12b}$$

APPENDIX B

KANADE-LUCAS-TOMASI TRACKER

Kanade-Lucas-Tomasi (KLT) feature tracker [70, 71] works on optical flow principle which is the movement pattern of the objects between two successive images. Optical flow assumes that the intensity of feature point in an image does not change in the consecutive image. It also assumes that the neighboring points make a similar motion. Considering a feature point $I(x, y, t)$, which is moved by (dx, dy) after a small time period dt:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{B.1}$$

Thus, Taylor series expansion of Eqn. (A.10) gives the optical flow equation:

$$I_x u + I_y v + I_t = \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + I_t = 0 \tag{B.2}$$

KLT tracker method uses a 3×3 patch around the feature point and assuming that all the 9 points would have the same motion. Then, (I_x, I_y, I_t) can be revealed with over-determined nine equations using a least-squares minimization. Complete iterative pseudocode is given in Table B.1.

Table B.1 Pseudocode of KLT feature tracker.

***Goal*:** To find location of feature point **v** in image **J** corresponding feature **u** in image

Image pyramid construction: $\{\mathbf{I^L}\}$ and $\{\mathbf{J^L}\}$ from L_0 to L_m

Initialization: $\mathbf{g^{L_m}} = \left[g_x^{Lm}\ g_y^{Lm}\right]^T = [0\ 0\ ]^T$

For *each pyramid in the sequence from L to L_m*:

Location of point: $\mathbf{u:}\ \boldsymbol{u^L} = [p_x\ p_y]^T = \frac{\mathbf{u}}{2^L}$

Find I_x: $I_x(x,y) = [I^L(x+1,y) - I^L(x-1\ ,y)]\ /\ 2$

Find I_y: $I_y(x,y) = [I^L(x,y+1) - I^L(x,-+1)]\ /\ 2$

Spatial Gradient Matrix:

$$\mathbf{G} = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix}$$

Initialize v: $\bar{\boldsymbol{v}}^{\mathbf{0}} = [\ 0\ \ 0\ ]^T$

For *each point k in the feature set:*

Difference:

$$\delta I_k(x,y) = I^L(x,y) - J^L\left(x + g_x^L + v_x^{k-1}, y + g_y^L + v_y^{k-1}\right)$$

Mismatch vector:

$$\bar{\mathbf{b}}^{\mathbf{k}} = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix}$$

Optical Flow: $\bar{\boldsymbol{\mu}}^{\mathbf{k}} = \mathbf{G^{-1}}\bar{\mathbf{b}}^{\mathbf{k}}$

Update v: $\bar{\mathbf{v}}^{\mathbf{k}} = \bar{\mathbf{v}}^{\mathbf{k-1}} + \bar{\boldsymbol{\mu}}^{\mathbf{k}}$

Final optical flow: $\mathbf{d} = \mathbf{g^0} + \mathbf{d^0}$

Find tracked feature point: $\mathbf{v} = \mathbf{u} + \mathbf{d}$

APPENDIX C

STEREO IMAGE RECTIFICATION

Stereo rectification is used in stereo motion estimation to project the images onto a common plane so that the corresponding image points are gathered into the same row, i.e. epipolar line. Since calibration results of the image sensors are not reliable, stereo rectification is done using uncalibrated image rectification techniques. Then, calibration results are used throughout the study. Figure C.1 shows the left and right images blended onto each other to show the difference in the angle of the view. Figure C.2 shows anaglyph of left and right images. Anaglyph is used for viewing stereoscopic 3D effects. It is implemented using red and cyan color filters which are chromatically opposite colors. Figure C.3 and C.4 show detected key points in left and right images, respectively and Figure C.5 presents the matched features between left and right images on the same plane. Outlier feature matches are removed in Figure C.6 and rectified stereo images are shown in Figure C.7. Figure C.8 shows the initial image anaglyph to easily compare with the resulting rectified stereo image Figure C.7.

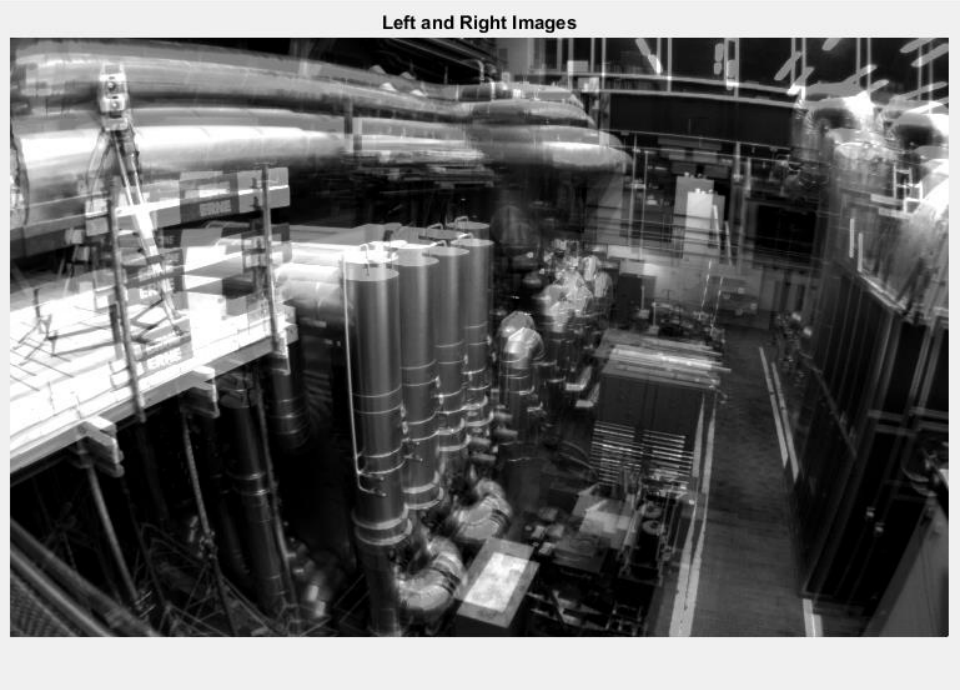

Figure C.1 Blended image of left and right images

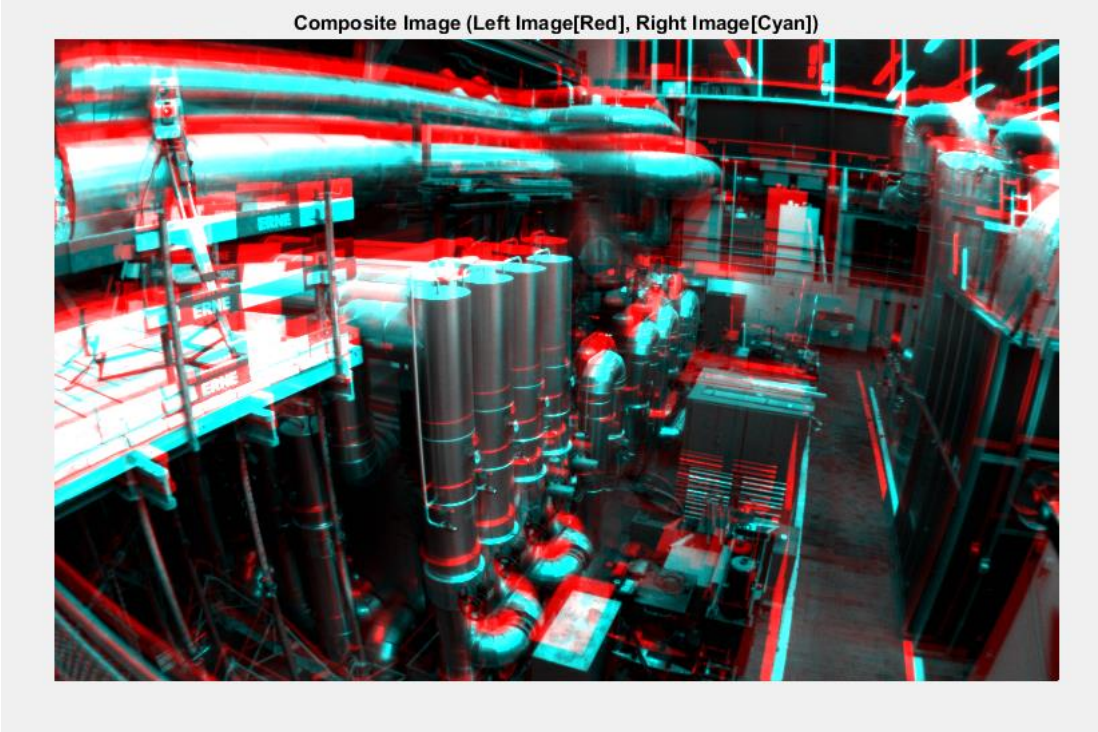

Figure C.2 Anaglyph of left and right images

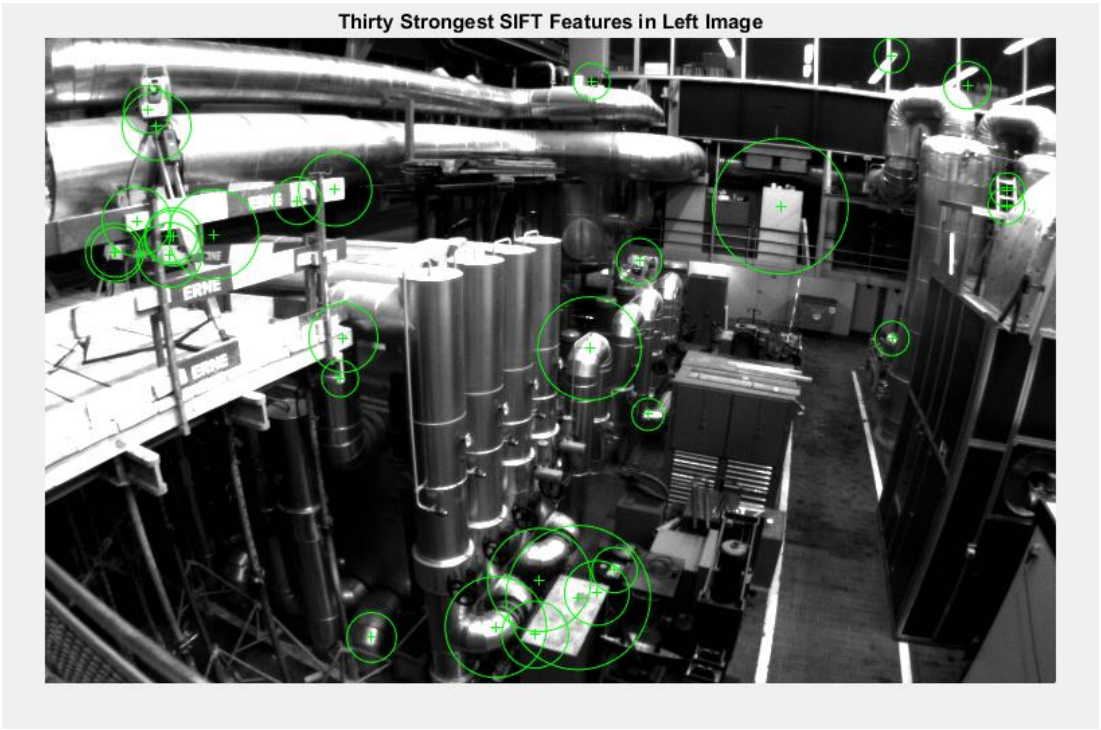

Figure C.3 SIFT features found in the left image

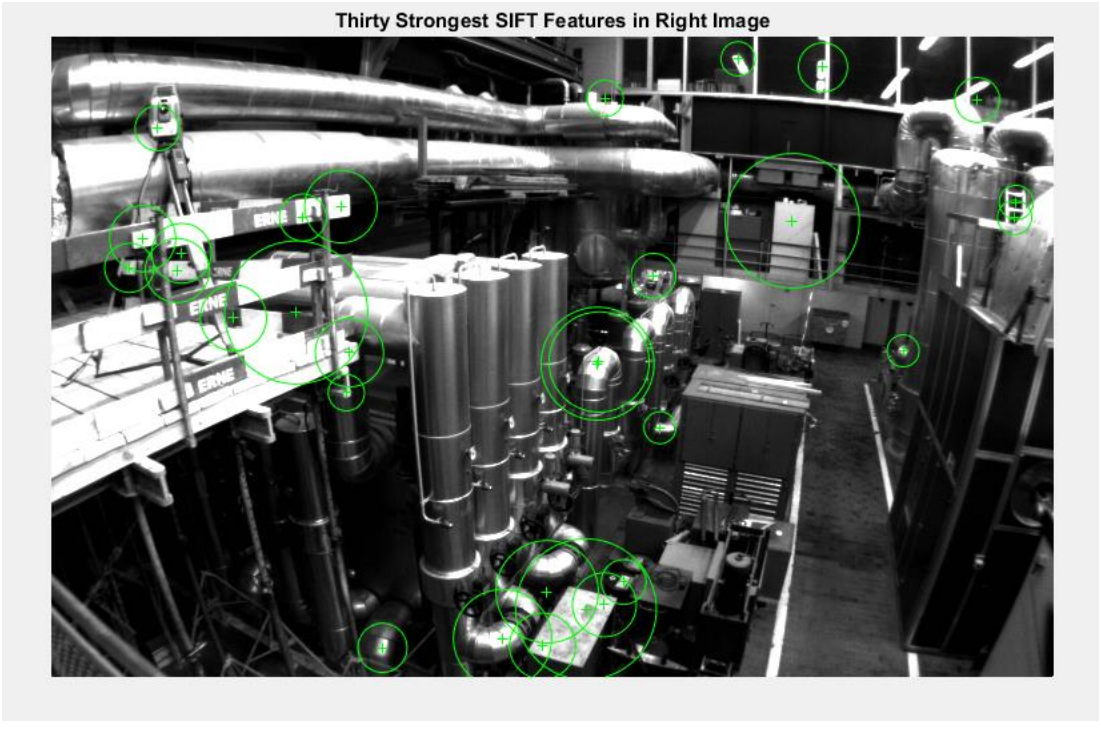

Figure C.4 SIFT features found in the right image

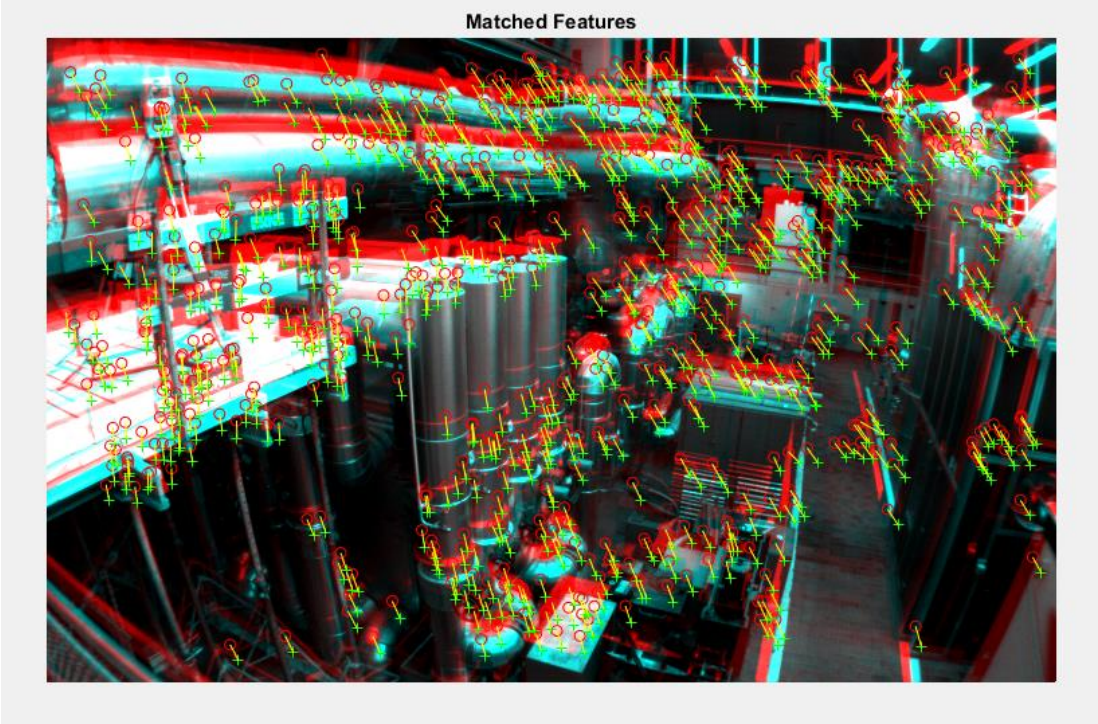

Figure C.5 Putative match of the key points from left to right image

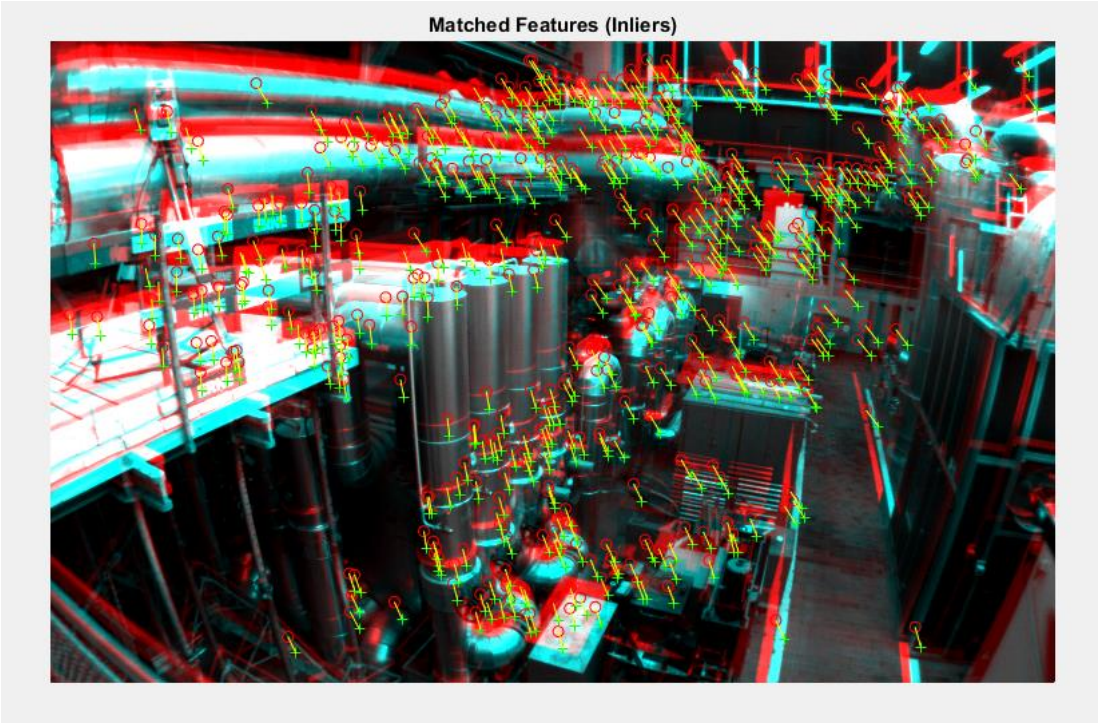

Figure C.6 Image after outlier rejection is applied

Figure C.7 Anaglyph of rectified images

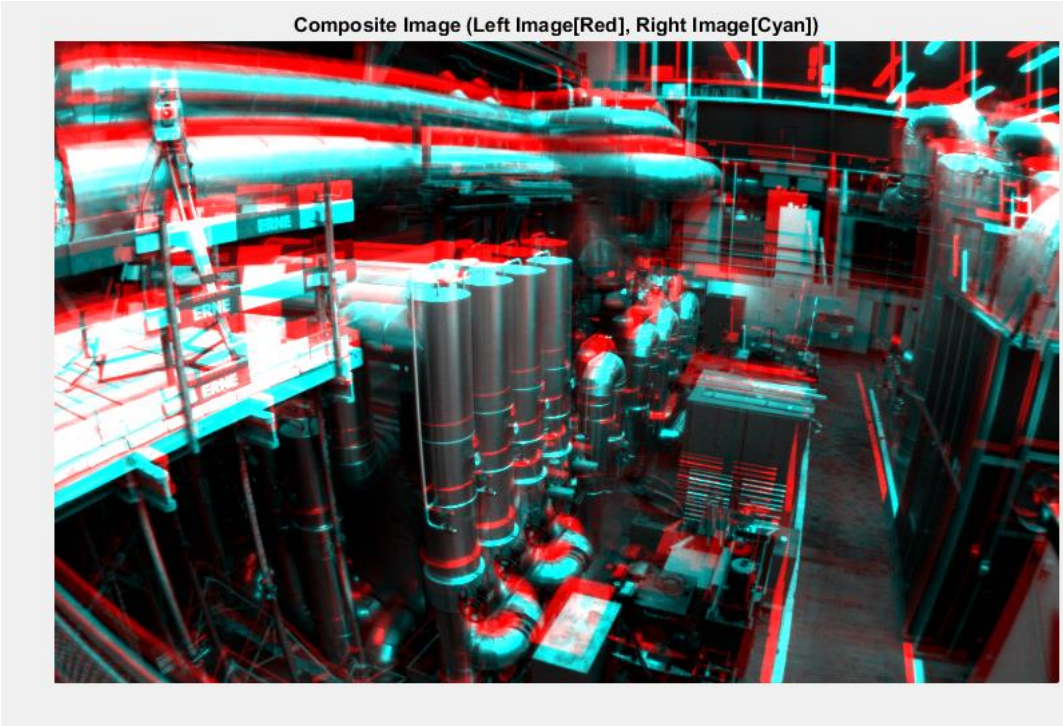

Figure C.8 Anaglyph of left and right images before rectification process

APPENDIX D

BUNDLE ADJUSTMENT

For the calculation of the BA, the objective of the sparse Levenberg-Marquardt algorithm [13] is to obtain an optimization when the parameter vector is defined as $\boldsymbol{P} = [\mathbf{a}^{\mathbf{T}}\ \mathbf{b}_{\mathbf{1}}^{\mathbf{T}}\ \ldots\ \mathbf{b}_{\boldsymbol{n}}^{\mathbf{T}}]^{\mathbf{T}}$, incremental vector is $\boldsymbol{\delta} = [\boldsymbol{\delta}_{\mathrm{a}}\ \boldsymbol{\delta}_{\mathrm{b}}]^{\mathbf{T}}$, and the measurement vector is $\mathbf{X} = [\mathbf{X}_1^{\mathrm{T}}\ \ldots\ \mathbf{X}_n^{\mathrm{T}}]^{\mathrm{T}}$, such that $\partial\widehat{\mathbf{X}}_i\ /\ \partial\boldsymbol{b}_j = 0$ for $i \neq j$. Nomenclature for the formulation is given as:

$$\mathbf{A} = \left[\mathbf{A}_1^{\mathbf{T}}\ \mathbf{A}_2^{\mathbf{T}}\ \ldots\ \mathbf{A}_{\mathbf{n}}^{\mathbf{T}}\right]^{\mathbf{T}} \tag{D.1a}$$

$$\mathbf{B} = diag(\mathbf{B_1}, \mathbf{B_2}, \ldots, \mathbf{B_n}) \tag{D.1b}$$

$$\sum\nolimits_{\boldsymbol{x}} = diag(\sum\nolimits_{\boldsymbol{x_1}}, \ldots, \sum\nolimits_{\boldsymbol{x_n}}) \tag{D.1c}$$

$$\boldsymbol{\delta}_{\mathrm{b}} = [\boldsymbol{\delta}_{\mathrm{b}_1}^{\mathrm{T}}\ \boldsymbol{\delta}_{\mathrm{b}_2}^{\mathrm{T}}\ \ldots\ \boldsymbol{\delta}_{\mathrm{b}_n}^{\mathrm{T}}]^{\mathrm{T}} \tag{D.1d}$$

$$\boldsymbol{\epsilon} = [\boldsymbol{\epsilon_1}^{\mathbf{T}}\ \boldsymbol{\epsilon_2}^{\mathbf{T}}\ \ldots\ \boldsymbol{\epsilon_n}^{\mathbf{T}}]^{\mathbf{T}} \tag{D.1e}$$

Here $\mathbf{A}$ and $\mathbf{B}$ are the Jacobian matrices, $\boldsymbol{\Sigma_x}$ is the covariance matrix, $\boldsymbol{\epsilon}$ is the error matrix with $\mathbf{J}\boldsymbol{\delta} = \boldsymbol{\epsilon} = \mathbf{X} - \widehat{\mathbf{X}}$

Algorithm:

1. $\lambda = 0.001$ is used as the starting point.
2. The derivate matrices as well as the error vectors are computed:

$$\mathbf{A_i} = \left[\partial \widehat{\mathbf{X}}_i / \partial \mathbf{a}\right], \qquad \mathbf{B_i} = \left[\partial \widehat{\mathbf{X}}_i / \partial \mathbf{b}_i\right], \qquad \in_i = \mathbf{X}_i - \widehat{\mathbf{X}}_i \tag{D.2}$$

3. Intermediate values are calculated as

$$\mathbf{U} = \sum_i \mathbf{A_i^T} \sum_{\mathbf{X}_i}^{-1} \mathbf{A_i} \tag{D.3a}$$

$$\mathbf{V} = diag\,(\boldsymbol{V_1}, \ldots, \boldsymbol{V_n}) \text{ in } \boldsymbol{V_i} = \mathbf{B_i^T} \sum_{\mathbf{X}_i}^{-1} \mathbf{B_i} \tag{D.3b}$$

$$\mathbf{W} = [\mathbf{W_1}, \mathbf{W_2}, \ldots, \mathbf{W_n}] \text{ in } \mathbf{W_i} = \mathbf{A_i^T} \sum_{\mathbf{X}_i}^{-1} \mathbf{B_i} \tag{D.3c}$$

$$\in_{\mathbf{A}} = \sum_i \mathbf{A_i^T} \sum_{\mathrm{X}_i}^{-1} \in_i \tag{D.3d}$$

$$\in_{\mathbf{B}} = \left[\in_{\mathbf{B_1}}^{\mathbf{T}} \in_{\mathbf{B_2}}^{\mathbf{T}} \ldots \in_{\mathbf{B}_n}^{\mathbf{T}}\right]^{\mathbf{T}} \text{ in } \in_{\mathrm{B}_i} = \mathbf{B_i^T} \sum_{\mathbf{X}_i}^{-1} \in_i \tag{D.3e}$$

$$\mathbf{Y_i} = \mathbf{W_i V_i^{*-1}} \tag{D.3f}$$

4. $\boldsymbol{\delta}_{\mathrm{a}}$ in the following equation is computed as

$$\left(\mathbf{U}^* - \sum_i \mathbf{Y_i W_i^T}\right)\boldsymbol{\delta}_{\mathbf{a}} = \in_{\mathbf{A}} - \sum_i \mathbf{Y_i} \in_{\mathbf{B_i}} \tag{D.4}$$

5. Each $\boldsymbol{\delta}_{\mathrm{b}_i}$ is computed.

$$\boldsymbol{\delta}_{\mathrm{b}_i} = \mathbf{V_i^{*-1}} \left(\in_{\mathbf{B_i}} - \mathbf{W_i^T} \boldsymbol{\delta}_{\mathbf{a}}\right) \tag{D.5}$$

6. Parameter vector is updated and the new error vector is computed.
7. If $\in_{new} < \in_{old}$, then $\lambda = \lambda/10$ and start again from step 2. If $\lambda < 0.001$ then terminate.
8. If $\in_{new} > \in_{old}$, then $\lambda = \lambda * 10$. and start again from step 4.

Covariance of the parameters a and b estimated using the above algorithm:

1. ***U, V and W is computed as in step 3, and also*** $\mathbf{Y_i} = \mathbf{W_i V_i^{-1}}$ is redefined.
2. $\sum_{\mathbf{a}} = \left(\mathbf{U} - \sum_i \mathbf{Y_i W_i^T}\right)^+$
3. $\sum_{\boldsymbol{b_i b_j}} = \mathbf{Y_i^T} \sum_{\mathbf{a}} \mathbf{Y_j} + \boldsymbol{\delta_{ij}} \mathbf{V_i^{-1}}$
4. The cross covariance $\sum_{\boldsymbol{ab_i}} = -\sum_{\mathbf{a}} \mathbf{Y_i}$

APPENDIX E

ROTATION QUATERNIONS

Quaternion notation [13] is used in this study which can be defined as:

$$\tilde{q} = q_s + q_a i + q_b j + q_c k \tag{E.1}$$

where q_s, q_a, q_b, and q_c are all real numbers. By definition, the complex numbers in (E.1) are

$$i^2 = -1, \qquad ij = -ji = k$$
$$j^2 = -1, \qquad jk = -jk = i$$
$$k^2 = -1, \qquad ki = -ik = j$$

A convenient form for notation is

$$\tilde{q} \Leftrightarrow \boldsymbol{q} = \begin{bmatrix} q_s \\ q_a \\ q_b \end{bmatrix} \tag{E.2}$$

Eqn. (E.3) denotes the addition and subtraction properties of two quaternions (Namely, $\tilde{q}_1 = q_{1,s} + q_{1,a} i + q_{1,b} j + q_{1,c} k$ and $\tilde{q}_2 = q_{2,s} + q_{2,a} i + q_{2,b} j + q_{2,c} k$):

$$\tilde{q}_1 + \tilde{q}_2 = (q_{1,s} + q_{2,s}) + (q_{1,a} + q_{2,a}) i + (q_{1,b} + q_{2,b}) j + (q_{1,c} + q_{2,c}) k \tag{E.3a}$$

$$\tilde{q}_1 - \tilde{q}_2 = (q_{1,s} - q_{2,s}) + (q_{1,a} - q_{2,a})i + (q_{1,b} - q_{2,b})j + (q_{1,c} - q_{2,c})k \quad \text{(E.3b)}$$

Their multiplication constitutes a Hamiltonian product. For three imaginary parameters, one has

$$\tilde{q}_3 = \tilde{q}_1\tilde{q}_2 \Longrightarrow \begin{cases} s_3 = q_{1,s}q_{2,s} - q_{1,a}q_{2,a} - q_{1,b}q_{2,a} - q_{1,c}q_{2,a} \\ a_3 = q_{1,s}q_{2,a} + q_{1,a}q_{2s} - q_{1,b}q_{2,c} - q_{1,c}q_{2,b} \\ b_3 = q_{1,s}q_{2,b} - q_{1,a}q_{2,c} - q_{1,b}q_{2,s} - q_{1,c}q_{2,a} \\ c_3 = q_{1,s}q_{2,c} - q_{1,a}q_{2,b} - q_{1,b}q_{2,a} - q_{1,c}q_{2,s} \end{cases} \quad \text{(E.4)}$$

Similarly, the norm of a quaternion is defined as

$$\|\tilde{q}\| = \sqrt{q_s^2 + q_a^2 + q_b^2 + q_c^2} \quad \text{(E.5)}$$

Consequently, the inverse of a quaternion is

$$\tilde{q} * \tilde{q}^{-1} = 1 \quad \Longrightarrow \quad \tilde{q}^{-1} = \frac{q_s - q_a i - q_b j - q_c k}{\|q\|} \quad \text{(E.6)}$$

Unit quaternion with a unit form and pure quaternion with zero real part are two types of quaternions. The inverse of unit quaternions equals to the complex conjugate of it ($\tilde{q}^{-1} = \tilde{q}^*$). For rotations, unit quaternions can be utilized. For example, in a 3D point **p** = (p_x, p_y, p_z), the axis of rotation can be formed as $\tilde{u} = x_u i + y_u j + z_u k$ and magnitude of rotation as θ. That is, the quaternion for this rotation becomes

$$\tilde{q} = \cos(\theta/2) + \sin(\theta/2)\, u \quad \text{(E.7)}$$

As a result, the Hamilton-Cayley formula is $\tilde{P}' = \tilde{q}\tilde{P}\tilde{q}^{-1}$ where $\tilde{P}$ is the pure quaternion representing the point **p** in $\tilde{P} = p_x i + p_y j + p_z k$ while $\tilde{P}'$shows the rotated point **p′**.

B.1. Transformation from Rotation Quaternions to Rotation Matrices

An orthogonal 3x3 matrix R with unit determinant shows a three-dimensional rotation. The transformation from a rotation quaternion to a rotation matrix can be made through Hamilton-Cayley formula with a matrix, resulting in the Euler-Rodrigues formula:

$$\begin{aligned} \tilde{P}' &= \tilde{q}\tilde{P}\tilde{q}^{-1} \\ \text{or} \quad \mathbf{p}' &= \mathbf{R}\mathbf{p} \end{aligned} \tag{E.8a}$$

$$\Rightarrow \mathbf{R} = \begin{bmatrix} q_s^2 + q_a^2 - q_b^2 - q_c^2 & -2q_sq_c + 2q_aq_b & 2q_sq_b + 2q_aq_c \\ 2q_sq_c + 2q_aq_b & q_s^2 - q_a^2 + q_b^2 - q_c^2 & -2q_sq_a + 2q_bq_c \\ -2q_sq_b + 2q_aq_c & 2q_sq_a + 2q_bq_c & q_s^2 - q_a^2 - q_b^2 + q_c^2 \end{bmatrix} \tag{E.8b}$$

The followings are the inverse transformations:

$$q_s^2 = (1 + r_{11} + r_{22} + r_{33})/4 \tag{E.9a}$$

$$q_a^2 = (1 + r_{11} - r_{22} - r_{33})/4 \tag{E.9b}$$

$$q_b^2 = (1 - r_{11} + r_{22} - r_{33})/4 \tag{E.9c}$$

$$q_c^2 = (1 - r_{11} - r_{22} + r_{33})/4 \tag{E.9d}$$

while

$$q_sq_a = (r_{32} - r_{23})/4 \tag{E.10a}$$

$$q_sq_b = (r_{13} - r_{31})/4 \tag{E.10b}$$

$$q_sq_c = (r_{21} - r_{12})/4 \tag{E.10c}$$

$$q_aq_c = (r_{13} + r_{31})/4 \tag{E.10d}$$

$$q_aq_b = (r_{21} + r_{12})/4 \tag{E.10e}$$

$$q_bq_c = (r_{32} + r_{23})/4 \tag{E.10f}$$

Since the signs of parameters are ambiguous, a sign for one parameter can be set using Eqn. (E.9) and others using Eqn. (E.10).

B.2 Orientation Represented as Quaternions

Orientation change can be shown as

$$\boldsymbol{\Theta} = [\theta_x \quad \theta_y \quad \theta_z]^T \tag{E.11}$$

where θ_x, θ_x and θ_x represent the change in attitude in radians. Small change in two consecutive orientation measurements is

$$\theta = \|\boldsymbol{\Theta}\| \tag{E.12}$$

Then, the orientation quaternion can be expressed as

$$\boldsymbol{q} = \begin{bmatrix} q_s \\ q_a \\ q_b \\ q_c \end{bmatrix} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \sin\left(\frac{\theta}{2}\right)\frac{\theta_x}{\theta} \\ \sin\left(\frac{\theta}{2}\right)\frac{\theta_y}{\theta} \\ \sin\left(\frac{\theta}{2}\right)\frac{\theta_z}{\theta} \end{bmatrix} \tag{E.13}$$

B.3 Advantages of Quaternions for Rotations

Two fundamental advantages of quaternions make them favorable to use in this study rather than other means of representing rotations such as Euler angles which have a pitch (rotation about the *x*-axis), yaw (rotation about the *y*-axis), and roll (rotation about the *z*-axis) and provide a convenient visualization of rotation.

One advantage is that quaternions are both global and non-singular in contrast to some three dimensional representations of rotation such as Euler angles that is non-singular but cannot represent some rotations. In other words, Euler angle is disadvantageous due to the phenomenon called *gimble lock*. The other advantage is that numerical accuracy of quaternions is much better under restricted rotations about an arbitrary axis.

APPENDIX F

MATLAB FUNCTIONS / OBJECTS DEVELOPED

This section consists of selected optimization, image processing and machine vision functions, objects and libraries used in MATLAB R2016a.

Function: `blur_filter`
Explanation: Custom function that takes gyroscope data, calculates the CWT and extract image index corresponding the blurry images
Input: Gyroscope measurements
Output: Index of blurred images

Function: `complementary_filter`
Explanation: Custom function to estimate velocity and acceleration from IMU.
Input: Sampling rate, time, accelerometer, gyroscope
Output: IMU-only elocity and position estimates.

Function: `csv2plot`
Explanation: Custom function that takes estimates written on a .csv file and plots the results.
Input: IMU position estimates, VO position estimates, GT position estimates, filtered position estimates
Output: plot for comparison

Function: `cwtft`

Explanation: built-in function that returns the continuous wavelet transform of the 1D input signal using an FFT algorithm.

Input: Gyroscope

Output: Transformed gyroscope signal

Function: `detectHarrisFeatures`

Explanation: Built-in function to find corner points using Harris-Stephens algorithm

Input: An image

Output: Corner points represented in 2D

Function: `disparity`

Explanation: Built-in function to form a disparity map using stereo images.

Input: Left and right images

Output: Disparity color map

Function: `errorcalc`

Explanation: Custom function that shows errors between ground truth and an estimate

Input: 3-axis estimate, 3-axis ground truth in the same frame

Output: RMS, max and percentage errors in x, y, z and total.

Function: `EKF_demo`

Explanation: Custom function that takes imu and visual estimations and applies extended Kalman Filter.

Input: IMU position estimates, VO position estimates, GT position estimates

Output: Filtered position estimates, filtered position estimates' errors

Function: `featureBucketing`

Explanation: Custom function to uniformly distribute the corner features in an image

Input: Input image, number of corners, window size

Output: List of bucketed feature vectors

Function: `filtfilt`

Explanation: Built-in function that performs zero-phase digital filtering

Input: a signal, designed filter

Output: filtered signal

Function: `findAdditionalNodes`

Explanation: Custom function to add additional tracking points if available

Input: Tracked clique, graph

Output: New set of nodes

Function: `MadgwickFilter`

Explanation: Custom function for inertial orientation estimates

Input: Sampling rate, time, gyroscope, accelerometer

Output: Orientation estimates of inertial sensors

Function: `minimize`

Explanation: Custom function that uses optimoptions and Levenberg-Marquardt algorithm to minimize the reprojection error

Input: 2D projections, corresponding 3D points, projection matrices,

Output: Updated reprojection matrices for both left and right images

Function: `numberOfTrackedFeatures`

Explanation: Custom function that takes the number of tracked features in a cell array and plots it to compare with the average times

Input: number of tracked features per frame

Output: plot of the number of tracked features per frame in comparison with the average times.

Function: `optimoptions`

Explanation: Built-in optimization function from MATLAB's optimization toolbox. This is used to minimize the image reprojection error in motion estimation stage.

Input: Algorithm such as Levenberg-Marquardt minimization, Maximum number of iterations
Output: Minimized reprojection error

Function: `PSD`
Explanation: Custom function for plotting PSD
Input: Sampling frequency and signal (Gyroscope or accelerometer data)
Output: PSD plot of the signal

Library: `quaternion_library`
Explanation: Custom library for quaternion manipulation such as converting quaternion to euler angles, rotation matrix or vice versa.

Function: `rectifyStereoImages`
Explanation: Built-in function to make stereo image pairs have epipolar lines on the same row.
Input: Stereo image pairs and stereo parameters consisting of intrinsic, extrinsic and distortion elements.
Output: Rectified left and right images

Function: `rms_error`
Explanation: Custom function that takes two arrays in the same length and calculates the root mean square error
Input: 2 arrays of same length
Output: RMS error

Function: `stereoParameters`
Explanation: Built-in function to store stereo parameters.
Input: Left and right camera parameters consisting of intrinsic, extrinsic and distortion elements, rotation and translation of right camera with respect to left camera.
Output: Stereo parameters object.

Function: `undistortImage`

Explanation: Built-in function to undistort radial and tangential distortions using camera parameters.

Input: An image with camera parameters consisting of intrinsic, extrinsic and distortion elements.

Output: Undistorted image and new image origin after undistortion.

Function: `timesAveraging`

Explanation: Custom function that takes VO computation times and presents the mean values in a proper manner.

Input: times cell array with the length of number of processed frames

Output: Total and average computation times for disparity map, KLT, outlier rejection and optimization, separately

Object: `vision.PointTracker`

Explanation: Built-in object to store a set of point tracks using Kanade-Lucas-Tomasi tracker. First using initialize function, point features are initialized in the first image. Then, the features are tracked using step function. Step outputs the coordinate of the point in the successive image and score point between 0 and 1 showing the trust ratio to point track.

Function: `visualOdometry`

Explanation: Custom function to calculate translation, rotation and scale from successive frames.

Input: Left and Right images at time t, left and right images at time t+1, left and right camera projection matrices.

Output: 3×3 rotation matrix, 3×1 translation vector, optimization iteration size