ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

ALTERNATIVE NAVIGATION METHODS: FUSION OF OPTICAL FLOW AND VISUAL-INERTIAL POSE ESTIMATION USING EKF

M.Sc. THESIS

Abdel Salam Bawarshi

Department of Aerospace Engineering

Aeronautics and Astronautics MSc Programme

May 2019

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

ALTERNATIVE NAVIGATION METHODS: FUSION OF OPTICAL FLOW AND VISUAL-INERTIAL POSE ESTIMATION USING EKF

M.Sc. THESIS

Abdel Salam Bawarshi
(511151101)

Department of Aerospace Engineering

Aeronautics and Astronautics MSc Programme

Thesis Advisor: Asst. Prof. Emre Koyuncu

May 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ALTERNATİF NAVİGASYON METODLARI: EKF KULLANILARAK, POZ TAHMİNİ İÇİN OPTIK AKIŞI İLE GÖRSEL ATALETLİYİ FÜZYON ETMEKTEDİR

YÜKSEK LİSANS TEZİ

Abdel Salam Bawarshi
(511151101)

Uçak Ve Uzay Bilimleri Bölümü

Uçak ve Uzay Mühendisliği

Tez Danışmanı: Asst. Prof. Emre Koyuncu

Mayıs 2019

Abdel Salam Bawarshi, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 511151101 successfully defended the thesis entitled "ALTERNATIVE NAVIGATION METHODS: FUSION OF OPTICAL FLOW AND VISUAL-INERTIAL POSE ESTIMATION USING EKF", which he prepared after fulfilling the require-ments specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Emre Koyuncu** Istanbul Technical University	.....
Jury Members :	**Asst. Prof. Ramazan Yeniçeri** Istanbul Technical University	.....
	Dr. Mustafa Umut Demirezen Roketsan	.....

Date of Submission : **03 May 2019**
Date of Defense : **June 2019**

To my dearest mom, beloved family and my lovely friends (E4W) ...
Thank you for endless support, encouragement, and sacrifices...

FOREWORD

I would like first to thank my adviser Prof. Emre Koyuncu who without his support, guidance and constant supervision throughout the process of writing this work I would not have been able to accomplish it. I also wish to thank all the colleagues, for their cooperation to conduct this analysis. My parents deserve a note of thanks: your wise counsel and kind words have, as always, served me well.

May 2019 Abdel Salam Bawarshi

TABLE OF CONTENTS

Page

ABBREVIATIONS

EKF : Extended Kalman Filter
SLAM : Simultaneous Localization And Mapping
GPS : Global Positioning System
GNSS : Global Navigation Satellite System
PTAM : Parallel Tracking and Mapping
IMU : Inertial Measurement Unit
VINS : Integrated Visual Inertial System
6DoF : Six Degrees of Freedom
MSCKF : Multi-State Constraint Kalman Filter
MSF : Multi Sensor Fusion Framework
RMSE : Root Mean Squared Error
OVIO : Visual Inertial Odometry
ROS : Robot Operating System
UAV : Unmanned Aerial Vehicle
MAV : Micro Aerial Vehicle
PDF : Probability Density Function

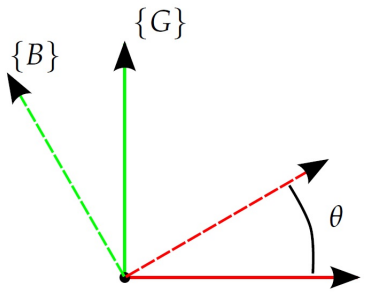
SYMBOLS

The following notation will be used throughout this thesis:

Basic Notation	**Example**
Scalars are lower case	x
Vectors are lower case bold	$\mathbf{x}$
Matrices are upper case bold	$\mathbf{X}$
A dot implies a continous time derivative	$\dot{\mathbf{v}} = \mathbf{a}$
A hat implies an estimated value	$\hat{\mathbf{x}}$
A bar implies a unit vector or unit quaternion	$\bar{\mathbf{q}}$
Coordinate frames are upper case	B or $\{B\}$
The coordinate frame of a point resides in the upper left	${}^{G}\mathbf{P}$
Rotation from frame A to B	${}^{B}_{A}\mathbf{R}$

Error Notation	**Example**
Δ represents the differential error of a vector	$\Delta\mathbf{P} = \mathbf{P} - \hat{\mathbf{P}}$
δ represents the orientation error	$\delta\theta$ or $\delta\mathbf{q}$
A tilde implies error in general	$X = \mathbf{x} - \hat{\mathbf{x}}$ with $\mathbf{x} = (\mathbf{P}, \theta)^T$

General coordinate frames used in this thesis are G, the global or world frame, B, the inertial body frame, and C, the camera frame.



$$ {}^{B}_{G}\mathbf{R} = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{B}_{G}\mathbf{q} = \begin{bmatrix} 0 \\ 0 \\ sin(\theta/2) \\ cos(\theta/2) \end{bmatrix} $$

Figure 1 : A simple example of the rotation and quaternion convention

LIST OF TABLES

Page

LIST OF FIGURES

Page

ALTERNATIVE NAVIGATION METHODS:
FUSION OF OPTICAL FLOW AND VISUAL-INERTIAL
POSE ESTIMATION USING EKF

SUMMARY

Localization and navigation are the two most important tasks for mobile robots. They try to equip the robot with the ability to solve two problems: how to reach a target destination, and how to determine where the robot is located with respect to its environment at a specific moment.

The GPS systems are the most common navigation systems seen in the robotics field nowadays. But unfortunately, not all locations are serviced by GPS signals (e.g., navigation in buildings, underwater, underground or in navigation in presence of GPS jamming) and that evoked the need to find alternative navigation methods, a non-GPS based system which can aid an inertial navigation system.

For autonomous vehicle navigation both cameras and inertial sensors seem to be ideal candidate because they do not project any detectable energy into the environment, estimate six degrees of freedom motion, are not subject to outages or jamming, and are not limited in range.

This thesis outlines a research effort focused on fusing optical and inertial sensors for autonomous navigation. Motivated by the requirement for autonomous navigation in GNSS denied environments, the approach is to integrate a camera with an inertial sensor. One tremendously important advantage is that the sensors are completely passive and no transmission (or reception) of radio signals is required. Which yields, an immunity to disruptions in the spectrum.

The main incentive of this thesis is, by fusing optical and inertial systems we will be able to tackle the issue of the current precision navigation methods limitations. The powerful characteristic of integrating more than one sensor in navigation systems, such as our VINS system, provides the capacity for it to perform at high level which with either sensor alone would be rather difficult to attain. In fact, this concept of combining various information sources to improve the overall outcome is omnipresent in almost every scientific and nonscientific field.

Also recognizing the fact that VINSs are highly nonlinear, we see significant challenges in terms of estimator initialization. This why in this work, the extended Kalman filter (EKF) algorithm for sensor fusion has been used which gives us a further assistance to improve the accuracy of the system with the backup sensors especially in the non-linear systems. The EKF additionally incorporates inertial measurement unit (IMU) data to increase the output framerate, which is important for control tasks of high level.

The proposed method utilizes (EKF) algorithm to estimate recursively the navigation state and the errors associated by tracking the locations of stationary objects in an image-aided INS.

ALTERNATİF NAVİGASYON METODLARI: EKF KULLANILARAK, POZ TAHMİNİ İÇİN OPTIK AKIŞI İLE GÖRSEL ATALETLİYİ FÜZYON ETMEKTEDİR

ÖZET

Yerelleştirme ve navigasyon mobil robotlar için en önemli iki görevdir. Robotu iki problemi çözme yeteneği ile donatmaya çalışıyorlar: Bir amaçlanan hedefe nasıl ulaşılmasını, ve nasıl robotun belirli bir anda çevresine göre nerede bulunduğunu belirleme.Elbette, bu iki soru birbirinden bağımsız değil ama oldukça sıkı bir şekilde bağlıdır.
Amaçlanan bir yolun başlangıcındaki bir robot doğru pozisyonunu bilmiyorsa, sonraki, amaçlanan hedefe ulaşma görevini yerine getirme konusunda sıkıntılarla karşılaşacaktır. Geçmişte, lokalizasyon, navigasyon ve haritalama için çeşitli algoritmik yaklaşımlar önerildi.

Son zamanlarda, bütün problem kompleksini bir kerede çözmeyi amaçlayan (örneğin, SLAM, eşzamanlı lokalizasyon ve haritalama) ve belirsizliği en aza indiren olasılıksal yöntemleri kullanarak farklı teknikler uygulanmaktadır. Yerelleştirme mobil robot biliminin temel konularından biri haline geldi, çünkü konumasl algılama kullanmadan, robotun sadece hareketsiz görevleri gerçekleştirmesi engellenecek ve çevresini dolaşma kapasitesine sahip olmayacak ve çarpışma gibi ciddi durumlardan kaçınamayacaklardır. GPS sistemleri, günümüzde robotik alanında görülen en yaygın navigasyon sistemleridir. Ancak ne yazık ki, tüm konumlara GPS sinyalleri tarafından hizmet verilmemektedir (örneğin, binalarda, su altında, yer altında navıgasyon veya GPS sıkışması varlığında navigasyonda) ve bu alternatif navigasyon yöntemleri bulma ihtiyacını uyandırdı, ataletsel navigasyon sistemine yardımcı olabilecek GPS tabanlı olmayan bir sistem.

Otonom araç navigasyonu için, hem kameralar hem de atalet sensörleri çevreye algılanabilir herhangi bir enerji yansıtmadıkları için ideal bir aday gibi görünmektedir, altı serbestlik dereceli hareket tahmini, kesintilere veya sıkışmalara maruz kalmamakta ve sınırlandırılmamaktadır. Ayrıca, yalnızca bir monoküler kamera kullanan yaklaşımlar, küçük boyutları, düşük maliyetli ve kolay donanım kurulumları nedeniyle bu alanda önemli ilgi görmüştür.
Bununla birlikte, yalnızca monoküler görme kullanan sistemler, metrik ölçeği kurtarma kabiliyetine sahip değildir. Dolayısıyla, gerçek dünyada robotik uygulama kullanımları sınırlıdır. Son zamanlarda, monoküler kameraya yardımcı olmak için görsel sistemle birlikte çeşitli sistemlerde bir atalet ölçüm birimi (IMU) kullanılmış ve metrik ölçek, yuvarlanma ve eğim açılarını gözlemleme özelliğine sahip entegre bir görsel eylemsizlik sistemi (VINS) ile sonuçlanmıştır . Hangi sırayla metrik durum tahminlerinin gerekli olduğu görevleri yerine getirmek için navigasyon sistemini nitelendirir. Ayrıca, hareket izleme performansı, görsel kayıplar arasındaki boşluğu kaplayan bu IMU ölçümleri entegrasyonu ile büyük ölçüde geliştirilebilir.

Bu kavram, insanların ve genel olarak hayvanların hassas navigasyon yeteneklerini gözlemlerinden ilham alır. Araştırma, farklı bilgi kaynaklarının sadece bu kombinasyonunu bilinçli olarak değil, refleks olarak da gerçekleştirdiklerini göstermiştir. Aslında, bilginin bilinçaltı kombinasyonu, çok sayıda sensörün tüm çıktılarını işleyerek sürekli olarak ortaya çıkar. Geleneksel olarak öğretilen duyuların yanı sıra işitme, görme, tat alma, dokunma ve koku almanın- hayvanlar aynı zamanda eklemlerinin birbirlerine göre ve daha pek çok pozisyondaki konumlarında eğim ve dönme hissine sahiptir. Havada, karada ve denizde hassasiyetle gezinmek için görsel ve ataletsel gözlemlerden yararlanır [4]. Navigasyon prensiplerinin bu güçlü doğal gösterimi, bu çalışmayı görsel ve ataletsel ölçümleri kullanarak hassas navigasyonu mümkün kılar. Bu tez, otonom navigasyon için optik ve atalet sensörlerini birleştirmeye odaklanmış bir araştırma çabasını göstermektedir. GNSS'nin engellendiği ortamlarda otonom navigasyon gerekliliği ile motive edilen yaklaşım, bir kamerayı ataletsel bir algılayıcıya entegre etmektir. Muazzam derecedeki önemli bir avantajı, sensörlerin tamamen pasif olması ve radyo sinyallerinin iletilmesi (veya alınması) gerekmemesidir. Hangi verim, spektrumdaki bozulmalara karşı bir bağışıklık.

Bu tezin ana teşviki, optik ve atalet sistemlerinin birleştirilmesiyle mevcut hassas navigasyon yöntemleri sınırlamaları konusunu ele alabileceğiz. VINS sistemimiz gibi navigasyon sistemlerine birden fazla sensörün entegre edilmesinin güçlü özelliği, yalnızca herhangi bir sensörle elde etmenin oldukça zor olacağı yüksek seviyede performans göstermesini sağlar. Aslında, genel olarak sonucun iyileştirilmesi için çeşitli bilgi kaynaklarının birleştirilmesi konsepti neredeyse her bilimsel ve bilimsel olmayan alanda her yerde bulunmaktadır. Ayrıca, VINS'lerin lineer olmadıkları gerçeğinin de farkında olarak, tahminci başlatması açısından önemli zorluklar görüyoruz. Bu nedenle, bu çalışmada, sensör füzyonu için genişletilmiş Kalman filtre (EKF) algoritması kullanılmıştır, bu da bize özellikle sistemdeki yedek sensörlerle doğruluğunu arttırmada yardımcı olur.

Görüntüleme sistemleri için, özellikle robotik uygulamalarında, geniş açılı imge elde edebilme her zaman önemli bir hedef olmuştur. Geleneksel kameraların çok kısıtlı bir görüş alanı vardır. Bu alanı arttırmak için kameranın döndürülmesi bir çözüm olabilir ancak bu sistem hareketli parçalara ve hassas konumlandırmaya ihtiyaç duyar. En önemli kusuru geniş açı elde etmek için gereken zamandır. Bunun nedenle gerçek zamanlı uygulamalar için uygun değildir. Balıkgözü mercekli kameralar ise çok kısa odak mesafesi olan dolayısıyla bir yarım küre içindeki objeleri görmemize olanak sağlayan kameralardır. Ancak bu kameraların zorluğu tüm gelen ana ışınların tek bir noktada kesişmesini sağlayacak merceğin tasarımıdır. Sahnenin bozunumsuz olarak perspektif şekilde görüntülenmesi gerekir. Bir diğer çözüm ise aynaların (kataoptrik elemanlar) merceklerle (dioptrik elemanlar) birlikte kullanıldığı katadioptrik sistemlerdir. Bu sistemlerde çok geniş görüş açısı tek seferde elde edilebilmektedir. Fiyatlarının da ucuzlamasıyla bu sistemler gün geçtikçe yaygınlaşmaktadır.

Tüm yönlü kameraların İHA'larda kullanımı son birkaç yılda görülmektedir. Katadioptrik kamerayı otonom hava aracına ilk uygulayanlar Hrabar ve Sukhatme'dir. İlk çalışmalarında helikopterin üç adet 'H' harfinin merkezine yönlenmesi sağlanmıştır. Ancak yaptıkları çalışmada katadioptrik imgelerin bozulma gibi karakteristik özellikleri dikkate alınmamıştır. Uygulamada yapay görsel hedeflere ihtiyaç duyulmakla

birlikte aracın sadece iki boyutlu kontrolü yapılmıştır. Diğer çalışmalarında ise helikopter üzerine yerleştirilmiş katadioptrik kamera ile balıkgözü mercekli iki kameranın kıyaslaması yapılmıştır. Bir diğer grup araştırmacı ise İHA'larının üç boyutlu pozunun tahmini üzerine çalışmaktadır. çalışmaları yunuslama ve yalpalama açılarının katadioptrik imgede ufkun Rasgele Markov Alanı (Random Markov Field) veya KYM (RGB) tabanlı Mahalanabis mesafesinin büyütülerek bulunmasına dayanır. Ancak bu çalışmalarda ufkun açıkça görünebildiği farz edilmiştir. Ayrıca bu ufka dayalı yöntemler sapma açısının belirlenmesine olanak vermediğinden tam bir durum saptaması yapılamamaktadır.

birlikte kullanan hibrit bir yaklaşım öne sürmüşlerdir. Ancak üç dönme açısı hesaplanırken düz araziye ihtiyaç duyulmaktadır ve hata birikmesini sıfırlamak için büyük oranda ufkun çıkarılmasına dayanmaktadır. Daha sonra ufka bağımlılıktan kurtulmak için kentsel çevrelerde kullanılmak üzere çizgiye dayalı bir metot geliştirilmiştir. Dikey yönü yeryüzü düzleminin normali gibi düşünerek yöntemlerle yine yalpalama ve yunuslama açıları tahmin edilebilmektedir. Ancak hala önemli kısıtlamalar vardır. Sapma açısı hesaplanamadığı gibi dikey yönün belirlenebilmesi için gökyüzünü temsil eden bölgelerin belirlenmesi gerekmektedir. Dolayısıyla bu yöntem yoğun kentsel bölgelerde ve kapalı alanlarda uygulanamamaktadır. Aynı zamanda gökyüzü ve yer piksellerinin ayrılması kolay bir işlem değildir. En son olarak yayınında bu eksiklikleri tamamlayacak bir yöntem sunulmuştur. Kentsel bölgelerde kullanılabilen ve hatanın birikmediği bu yayında katadioptrik imgelerdeki kaçış noktaları ve bu imgeler arasında sonsuz homografi kullanılmıştır. Bu yöntemle tüm dönme açıları belirlenebilmektedir. Gerçek görüntü dizileriyle yaptıkları deneylerin sonuçları gerçek değerlere çok yakındır.

1. INTRODUCTION

localization and navigation are the two most important tasks for mobile robots [2]. They try to equip the robot with the ability to solve two problems: how to reach a target destination, and how to determine where the robot is located with respect to its environment at a specific moment. Certainly, these two questions are not isolated from each other, but rather tightly coupled. If a robot at the beginning of an intended path does not know its accurate position, the latter will encounter troubles in accomplishing the task of reaching the aimed destination. In the past, various algorithmic approaches were proposed for localization, navigation, and mapping. Recently, different technique is being implemented which aims to solve the whole problem complex at once (e.g. SLAM, simultaneous localization and mapping) and that by employing the probabilistic methods which minimize uncertainty.

Localization has become one of the fundamental topics in mobile robotics, because without the use of its spatial detection, the robot would be restricted to merely perform immobile missions and will not have the capacity to navigate its surrounding and will not be able to avoid parlous cases such as collisions. The GPS systems are the most common navigation systems seen in the robotics field nowadays. But unfortunately, not all locations are serviced by GPS signals (e.g., navigation in buildings, underwater, underground or in navigation in presence of GPS jamming) and that evoked the need to find alternative navigation methods, a non-GPS based system which can aid an inertial navigation system. For autonomous vehicle navigation both cameras and inertial sensors seem to be ideal candidate because they do not project any detectable energy into the environment, estimate six degrees of freedom motion, are not subject to outages or jamming, and are not limited in range. Moreover, approaches that use only a monocular camera have gained significant interests in the field due to their small size, low-cost, and easy hardware setup [3]. However, systems which use monocular vision-only do not have the capability to recover the metric scale. Thus, their usage is limited in real-world robotic applications. Lately, an inertial measurement unit (IMU)

has been used in various systems along with the visual system to aid the monocular camera and results in an integrated visual inertial system (VINS) equipped with the ability to observe the metric scale, roll and pitch angles. Which in turn qualifies the navigation system to perform missions where the metric state estimations are required. Furthermore, the motion-tracking performance can be tremendously enhanced by this IMU measurements integration which bridging the gap between losses of visual. This concept is inspired by observing the precision navigation capabilities of humans, and animals in general. Research has indicated that they not only perform this combination of different information sources consciously but also subconsciously. In fact, the subconscious combination of information occurs continuously by processing all outputs of the vast number of sensors. Apart from the traditionally taught senses — hearing, seeing, taste, touch, and smell — animals also have a sense of inclination and rotation, of the position of their joints with respect to each other and much more. it utilizes visual and inertial observations to navigate with precision in the air, land, and sea [4]. This powerful natural demonstration of navigation principles guides this work to make precision navigation using visual and inertial measurements possible. This thesis outlines a research effort focused on fusing optical and inertial sensors for autonomous navigation. Motivated by the requirement for autonomous navigation in GNSS denied environments, the approach is to integrate a camera with an inertial sensor [5]. One tremendously important advantage is that the sensors are completely passive and no transmission (or reception) of radio signals is required. Which yields, an immunity to disruptions in the spectrum. The main incentive of this thesis is, by fusing optical and inertial systems we will be able to tackle the issue of the current precision navigation methods limitations. The powerful characteristic of integrating more than one sensor in navigation systems, such as our VINS system, provides the capacity for it to perform at high level which with either sensor alone would be rather difficult to attain. In fact, this concept of combining various information sources to improve the overall outcome is omnipresent in almost every scientific and nonscientific field. Also recognizing the fact that VINSs are highly nonlinear, we see significant challenges in terms of estimator initialization. This why in this work, the extended Kalman filter (EKF) algorithm for sensor fusion has been used which gives us a further assistance to improve the accuracy of the system with the backup sensors especially in the non-linear systems. The EKF additionally incorporates inertial measurement unit

(IMU) data to increase the output framerate, which is important for control tasks of high level. The proposed method utilizes (EKF) algorithm to estimate recursively the navigation state and the errors associated by tracking the locations of stationary objects in an image-aided INS.

1.1 Purpose of Thesis

The main purpose of this thesis is developing a combination of accuracy and low latency state estimation algorithms required for flying robots in order to perform steady and robust mission and that by using cameras and inertial measurement units (IMUs) to form visual-inertial odometry (VIO) algorithms due to their ability to operate without external localization from motion capture or global positioning systems such as (GPS, GNSS, etc.).

This work evaluates an array of publicly-available VIO pipelines i.e. EuRoC datasets, which contain six degree of freedom (6DoF) trajectories typical of flying robots. And that by using EKF to fuse optical and inertial systems we will be able to tackle the issue of the current precision navigation methods limitations.

provide these qualities under the computational constraints of embedded hardware. by fusing optical flow and inertial systems to solve the problem of the current precision navigation methods limitations.

1.2 Literature Review

As robots are becoming more popular, they are now more obtainable and the access for them is way easier and they also became simpler to develop. Especially with the hardware becoming less expensive, and the process reach a point where they can perform rapidly. Modern advances in research have allowed many technological applications such as Mars driverless car Rovers which they are considered high-quality semi autonomous vehicles. However, these models are outliers. The larger part of robots still possess shortage in the most essential forms of autonomy. There are three main questions which each autonomous mobile robot has to respond to: Where am I? How did I reach here? and How do I reach there? The response to this riddle lies in three concepts odometry, localization, and control.

- **Odometry** is the method of gradually estimating the pose (position and orientation) of the robot utilizing mobile sensors. Abstract odometry is sometimes independent from localization and mapping, trying to compare the pose in respect to, for instance, the position at which it begins.

- **Localization** provides the robot with information about where its location in a map and is sometimes instantly combined with the mapping procedure. To localize, a map is needed. And to generate a map, localization is needed.

- **Control** is the technique for reaching one when is at another. Control theory makes up a tremendous share of robotics, it is full of research on topics such as grasping, moving and even flight.

The main focus for this thesis is the odometry. Particularly, by developing a method for non-map self-contained odometry. Some times referred to as pose estimation as well, is extensively researched and extremely significant in robotics and computer-science. Almost each type of robot, whether it was flying droneso or automated Roomba (robot vacuum cleaner), must acquire the knowledge to accurately estimate where they are in order to work. In the cases when the map is not provided or when generating one is not possible, there is no way to figure out the precise location of the robot, so an acceptable guess of pose estimation is able to be conserved along with a degree of uncertainty. There are plenty of famous examples of advanced technology that utilize odometry. Any vehicle with a GPS navigation-system is exploiting a basic form of odometry. Some more advanced vehicles such as Google's self-driving car [1] utilize far more sophisticated techniques for more accurate measurements. These vehicles utilize high-price hardware such as 3-D scanners and various cameras. A more cost-efficient odometry is that performed by the Roomba . It just employs infrared sensors and bumper sensor in order to navigate a room, along with otary encoder that can convey information about the wheels movement and acoustic sensors to search for the unclean spots in the environment.

Figure 1.1 : Google's driverless car [1] utilize odometry with expensive high-quality sensors

1.3 Problem Statement

There are plenty of techniques for odometry and many are specific to a certain type of hardware or are limited to a particular environment. Some of these techniques utilize exceedingly expensive hardware, where sophisticated calibration is needed, or do not perform in real time without being equipped with powerful parallel processors. Thus, it is important for the sensors used to be low-cost and possess easy hardware setup and require modest calibration and that to make to render odometry easily attainable. In addition, the implemented algorithm must be able to work in real time and on a CPU with a single core. The interest of this thesis is for the pose of the body in motion to be estimated with small size and minimal low-cost sensors. Specifically, our goal is to develop an odometry method which satisfies the following objectives:

- Consistency: The uncertainty of the pose estimate must be correctly reported by the method.
- Quality: The method must perform an estimation which is close to the actual value.
- Accessibility: low-cost hardware must be found without overly complicated calibration techniques.
- Scalability: The method must be able to work over long distances and for relatively long time periods

- Robustness: And that is for the method to be sufficiently robust to operate in various environments (i.e. outdoors and indoors) and in the existence of common environmental disturbances (e.g. cars and pedestrians).
- Efficiency: The method must be to run in real time, ideally on the equipment itself.

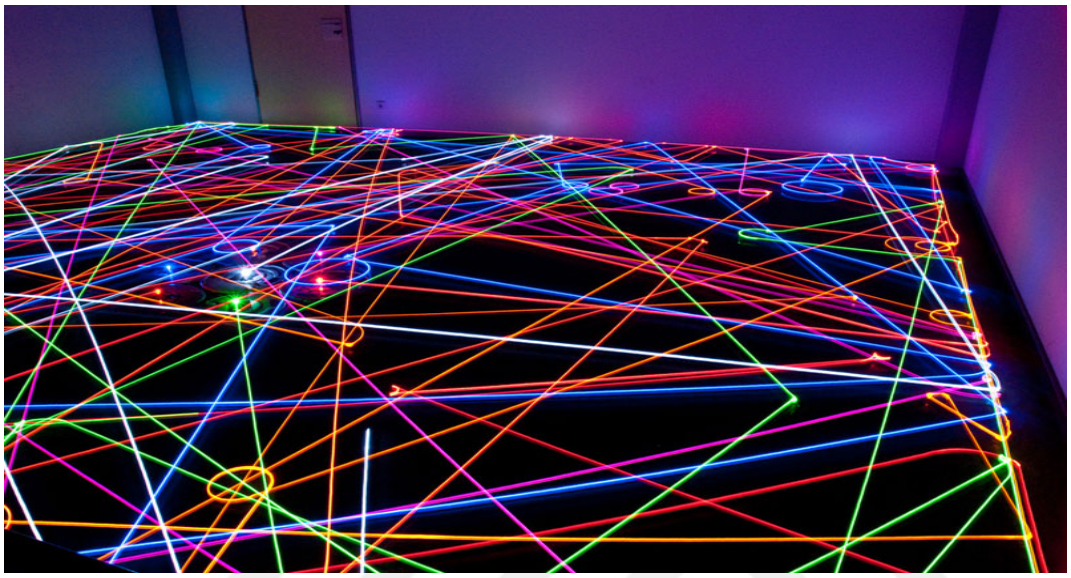


Figure 1.2 : A time-lapse image of a Roomba as it moves around a room. the Odometry helps in preventing traversing the same positions repeatedly.

1.4 Proposed Solution

In the recent years, the camera has come to the scene as being a low-cost, widely-used small sensor, it can be found in several devices such as phones, quadcopter and numerous vehicles. A video camera can solely offer a massive quantity of data. The notion of putting that huge amount of information into usable structural data which describes the surroundings is a complicated process and a sufficiently mature field to engage in a procedure of education nowadays. Many of the vision-based navigation techniques focus on finding features in the image, usually edges and corners, along with characteristics which make these images when seen again easily recognizable. Hence, by matching properties between images, along with using either various cameras or a single mobile camera, several 3D data can be obtained via triangulation [6]. The points in the triangulation show structural data about the camera's environment which in turn provide information about the position of the camera itself.

The systems that operate with Stereo camera are relatively larger, more costly, and calibrating them is rather complex compared to the single camera system. On the other hand, they are capable of offering depth information accurately and swiftly [7]. The extent to which a stereo system can accurately estimate depth relies heavily on the baseline, the range between these cameras, and the standard of calibration. The fundamental preference over monocular systems is that no mobility requirement is needed to estimate the depth. Mobile cameras is also able to offer depth data, but they possess an inherent shortage of scale information. For instance when the camera is used as the only sensor , the system will suffer from the problem of determining scale. This issue stems as a outcome of the method a camera projects the 3D-world into a 2D-image. This famous issue has been approached by adding additional sensors or an initialization procedure [8].

Several sensors can be found to be low-powered, inexpensive and small-sized and they provide scale data. Ultrasound sensors for example are capable of offering metric information, the distance of the object can be estimated by measuring the time between sending a signal and receiving an echo. These type of sensors are adequate with quadrotors for detecting their height above ground. However, due to the lack of orientation and panoramic view, ultrasound sensors are not practicable. Another height detectors which can be used are barometers, they can determine air pressure levels accurately which in turn can be converted over time into scaled height differences, the problem is they have a high sensitivity to the changes in weather and they do not perform good in indoors settings. (IMU) The Inertial Measurement Unit can be found in many modern applications such as quadrotors and smartphones and it gives accurate high-frequency readings of acceleration and rotational velocity. Although these sensors measurements are usually associated by noise, and there is an integration process needed in order get position from acceleration and that by integrating the signal twice, this inertial information is useful to successfully provide short term scale data.

Combining the camera with the IMU for odometry is called Visual Inertial Odometry, or VIO. In this thesis we will explore VIO and its implementation on a mobile device (drone) with respect to all the criteria listed above.

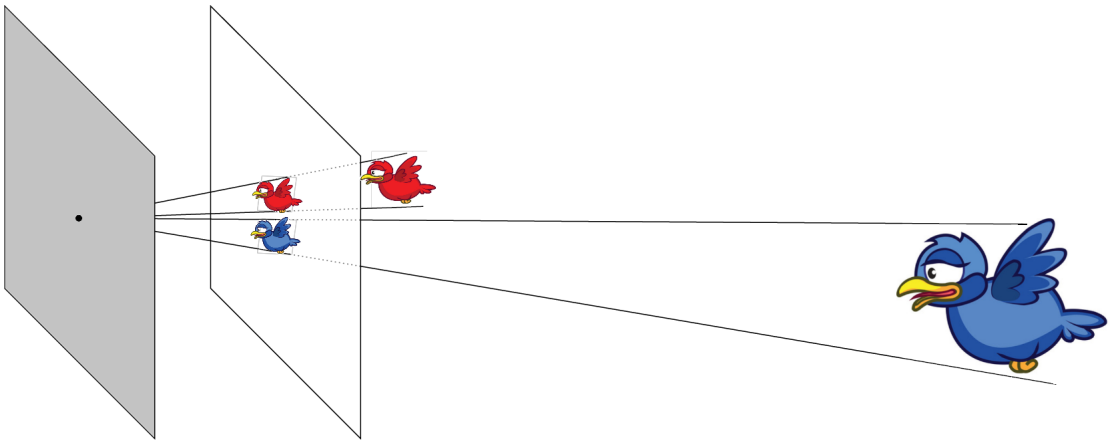


Figure 1.3 : The projective model of the camera brings some scale ambiguity. It is extremely difficult to tell the scale of an object with only one image. In figure above both birds appear to have the same size in the image plane because the bigger bird is simply farther away.

2. VISUAL INERTIAL ODOMETRY

Since its introduction in 1986, (SLAM) Simultaneous Localization and Mapping has been a common and a significant progress toward autonomy in robotics. In the SLAM model [9], the landmarks are being detected and compared with a map to localize the robot and in the meantime newly detected landmarks are appended to the map. Landmark observation process is doable with various sensors including laser-scanners and cameras. Early SLAM systems experienced a number of flaws.

- Having a map is required for localization, on the other hand, one must localize to create a map [10]. This egg and chicken dilemma causes rather complicated initialization procedures, depending on the hardware engaged.
- Systems with monocular (single)camera possess a built-in inability to assess scale. Initialization patterns of defined size or movements of known length were one of the early popular solutions [11]. Combining several sensors has lately been a more common choice [8].
- Running in real time was initially far-fetched. Over the years much research has been devoted to this issue. New solutions are able to be run on mobile smart phones [12].
- Conventional SLAM closely couples mapping, localization and odometry making it hard to scale to large settings [13]. Recent systems are getting closer toward decoupled systems for scalability, efficiency, and encapsulation.

For the rest of the chapter we will talk about important historical solutions to the mentioned issues and how they have lead to VIO. In addition, we represent the solution we have picked and how it is related to the state of the art.

2.1 Filter Based SLAM Methods

Early SLAM methods described the robot's state, as well as the 3D-landmarks, probabilistically by employing the weighted average of noisy readings in a process called filtering. The Extended Kalman Filter is a widespread filtering approach and was the first common option for SLAM. In EKF-SLAM, the uncertainty correlations between the landmarks and the pose of the robot are kept in a big covariance matrix. As the number of landmarks increases, the computational complexity of EKF-SLAM experiences a quadratic growth, which is for real-time applications quite unfeasible.

Real-time monocular SLAM (MonoSLAM) was first introduced by Davison [13] where he used an Extended Kalman Filter in 2003. Before that, state of the art SLAM systems could just perform an off-line batch processing. However, the real-time necessity had to bring a slight reduction in precision in exchange for constant computation time. In MonoSLAM, most of the attention is focused on merely attaching landmarks to the map that remove features that are no longer useful and contribute important location information. While MonoSLAM can run in real time, it is nevertheless restricted to a small area due to the complexity that remains quadratic regarding the number of landmarks. Furthermore, MonoSLAM parameterizes observed landmarks with bearing vectors and an initially unexplored depth estimate. Few errors in measurements can produce huge non-linear errors in the depth, which generates inconsistency in the filter. To solve this issue, Montiel et al. [14] established the unified inverse depth representation to ameliorate consistency. Inverse depth notation of the landmarks permits a more exact uncertainty representation due to the high degree of linearity which renders it feasible for the features to be represented at great distances.

Several solutions have been suggested to address the real-time limitations of SLAM in large areas. SLAM shortcut approaches such as [15] try to make a deferment of particular information evaluation until definitely necessary, this method it found to accomplish the best outcomes when it remains in a small repeated area. Another method for example FastSLAM [16], utilizes a Particle Filter in the place of the Extended Kalman Filter for estimating the landmarks locations. This decreases the

complexity to O(k log n), where k represent the number of particles on the order of 1000 and n represents the landmarks number. While this shows an enhancement over EKF-SLAM's quadratic complexity, it nevertheless does not scale indefinitely.

2.2 Key-Frame Based Methods

In order for the monocular SLAM system to accomplish a real-time performance, Klein and Murray [11] created Parallel Tracking and Mapping (PTAM) in 2007. In which they used keyframe based algorithms instead of using only filters. The key-frame based methods keeps a sparse set of significant images while exploring a map, along with the landmarks observed and the position of the camera. The main advantage of keyframes is the retention of recorded information for explicit use instead of marginalizing out camera poses and previous landmarks, as in EKF-SLAM. PTAM increased the number of usable landmarks remarkably in comparison to EKF-SLAM, despite the fact the original PTAM was still restricted to a comparatively small work area. PTAM also detached mapping from tracking and made it able to run on separate threads in parallel. This uncoupling was a leading step which unfolded the research field in applying graph optimization to the task of mapping. Every one of these keyframes could be described as a node in a graph with local euclidean limitations. There have been various suggested methods for optimizing this graph for real-time large scale SLAM [17] [18].

2.3 Dense Methods

The aforementioned approaches are camera based methods and they almost entirely depend on the detection and triangulation of landmarks. Detecting prominent repeatable patches in an image produce a sparse representation of the surrounding environment. Dense methods, which also known as direct methods, utilize the entire image for frame to frame tracking. Kinect Fusion [7] and Dense Tracking and Mapping (DTAM) [19] in 2011 were the first essential

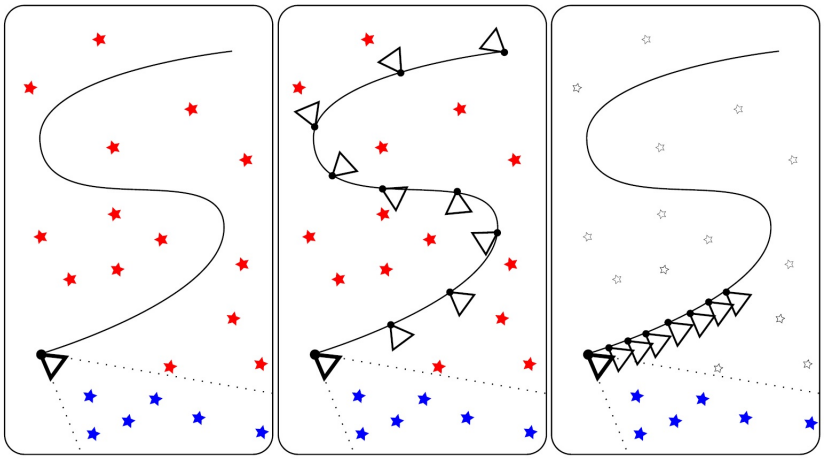


Figure 2.1 : A simple illustration of various methods. On the left is EKF-SLAM, with old (red) points kept in the map and only the latest camera pose. In the middle is a keyframe example with sparse poses and their landmark associations. On the right is the MSCKF with a sliding window of camera poses to track shared landmarks. Previous poses and landmarks are discarded.

real-time advances in dense tracking and mapping. There is another example, REMODE [20], which makes use of probabilistic methods for 3D monocular reconstruction. Nevertheless, high levels of parallel processing is required for these early methods, which were done on a GPU. In 2013 semi-dense visual odometry for a monocular camera has been introduced introduced Engel et al. [21]. This system was able to run in real-time on several cores and was later on successfully ported to a modern smartphone.

2.4 Inertial Aided Methods

As mentioned above, the lack of scale is a popular issue amid monocular odometry and SLAM systems. Moreover, scale drift could happen when the scale is being estimated during initialization solely [22]. The projective model of a single camera hinders its ability to determine scale in an image. Numerous SLAM algorithms, such as PTAM, have an initialization sequence to gain scale or they utilize other sensors. Sadly, a drifting over time for the scale estimate as error accumulates will occur due to initializing the scale at the beginning of an odometry sequence. As stated before, the ideal sensor configuration for a generalized system with unknown motor capabilities is a camera with an (IMU) the Inertial Measurement Unit integrated together. Inertial aided EKF-SLAM [23] has succeeded in reducing the scale issue without a complicated initialization procedure. But the inertial aided EKF-SLAM

algorithm still suffers from all the normal EKF-SLAM problems. Lately, Weiss et. al. [24] have put into use an IMU and monocular camera for odometry by exploiting the camera as a black box 6-DOF sensor, which is then loosely coupled with a Kalman Filter using IMU readings for state estimation. A main disadvantage of this system is that important probabilistic information conserved in the filter is not utilized by the visual odometry module for outlier detection. Outliers happen when the visual odometry module miscalculate while tracking objects or it tracks mobile objects that it believes are stationary.

2.5 MSCKF

The MSCKF (Multi-State Constraint Kalman Filter) has been represented by Mourikis and Roumeliotis [25] [26] in 2006, to approach several issues with inertial aided EKF-SLAM. The MSCKF is a pure odometry approach. Because it does not build a map, its intricacy is linear in the number of features and it does not experience assumptions about the landmarks nature. Instead of estimating the poses of landmarks in the filter, it keeps a sliding window of camera poses from which landmarks are precisely triangulated utilizing the entire data available. The landmarks are then employed as limitations on the window of camera poses. Subsequently, the filter accomplishes rather improved results than EKF-SLAM. Numerous suggested enhancements have been published since the establishment of the MSCKF, M. Li and Mourikis augmented the accuracy, made it applicable on a resource restricted system like mobile-phone for example [27], and designed a innovative approach for rolling shutter camera compensation [12]. Online calibration is achievable also for various parameters of the system. The transformation between the IMU and camera is not always accurately determined, but able to be estimated online [15]. For this thesis the MSCKF has been chosen as our solution and that for several reasons all listed earlier and because it also have incorporated the latest techniques for online calibration.

2.6 Outline

The rest of the thesis is devoted to clarify the details, the implementation, and what the outcomes indicate. As represented in the algorithm, the state and the uncertainty have to be propagated exploiting IMU measurements and then updated through camera

measurements. The Extended Kalman Filter is composed of this two-step process and will be described in Chapter 3. The propagation stage principally utilizes the IMU measurements to estimate the way the state varies at a high frequency. The nature of these readings, the IMU noise characteristics, and the IMU calibration parameters are explained in Chapter 4. Images are observed less repeatedly than IMU measurements are available. These images involve significant information which is distinguished and compared with old images. The details of computer vision applicable to this thesis are explained in depth in Chapter 5. The algorithm explained here is on a sophisticated degree and leaves out a considerable load of important details. These details are presented in the essential chapter of this thesis, Chapter 6. We have implemented the MSCKF to work with EuRoC MAV Dataset. Details of the implementations and issues we encountered are explained in Chapter 7. Eventually, the results of our experiments, both in simulation and with actual data, are represented in Chapter 8.

3. PROBABILISTIC STATE ESTIMATION

The probability theory is main player in filter based approaches for state estimation. When tracking the location of a robot, we aspire to examine its location in a probabilistic manner, instead of the explicit method, this way the uncertainty in our estimate can be tracked. In this chapter we describe the basics of probability theory inasmuch as they apply to state estimation. Especially, we describe what it means to estimate the state and keep a measure of the uncertainty. Then linear filtering with the Kalman filter will be introduced, and eventually nonlinear filtering which uses the Extended Kalman Filter.

3.1 Probability Theory Basics

We represent the true unidentified state as a random variable, x, with a probability distribution. This distribution possesses a probability density function (pdf): which is continuous function, p, that expresses the relative likelihood of x. The pdf is not the exact concept as the probability. Unlike the probability, a pdf can acquire values higher than 1. On the other hand, it is crucial, that the entire area under the curve is equal to 1. The probability that x is within a specific range is equal to the integral of p in that range.

$$Pr(a \leq x \leq b) = \int_a^b p(x)dx \tag{3.1}$$

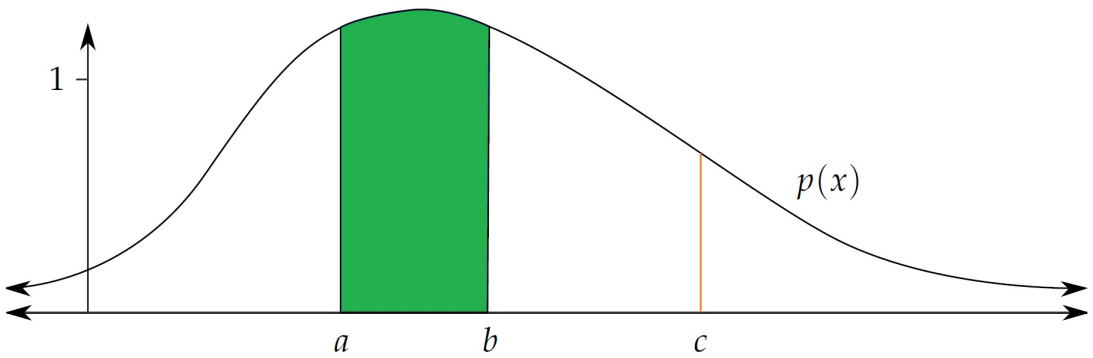


Figure 3.1 : An example probability density function. The probability $Pr(a \leq x \leq b)$ is the blue area while $Pr(x = c) = 0$. The total area beneath the curve must be equivalent

The expected value of x, or E[x] is the weighted average of the whole potential values of x. The expected value is calculated by multiplying each possible value of x by p(x), which serves as a weight for that value, and adding the results. In some cases, particularly the **normal distribution**, the expected value is the equivalent to the maximum value of p(x). The variance of x, Var[x], is the average squared difference between the expected value and every value of x and. High values for the variance imply that x is distributed far away from the mean. If x is a sensor reading, it could mean the sensor is heavily corrupted by noise.

$$E[x] = \int x \;\; p(x) \;\; dx \tag{3.2}$$

$$Var[x] = E[(x - E[x])^2] = \delta^2 \tag{3.3}$$

We describe the standard deviation of x as δ, and that is the square root of the variance.

3.2 The Normal Distribution

The most useful and common class of probability density functions is the normal, or Gaussian, distribution. The normal distribution has a number of advantages.

- The sensor noise is generally distributed normally, which nominates it to be a natural preference for modeling sensor data. See figure 3.2. This is also known as Gaussian white noise.
- The expected value, E[x], is also the value that maximizes the pdf and is the mean of the Gaussian.
- The mean and variance are the Gaussian distribution parameters, introducing it as intuitive and simple to analysis.
- A linear transformation of a Gaussian random variable is also Gaussian.
- Joint probability or, the intersection of two Gaussian random variables also stays Gaussian.

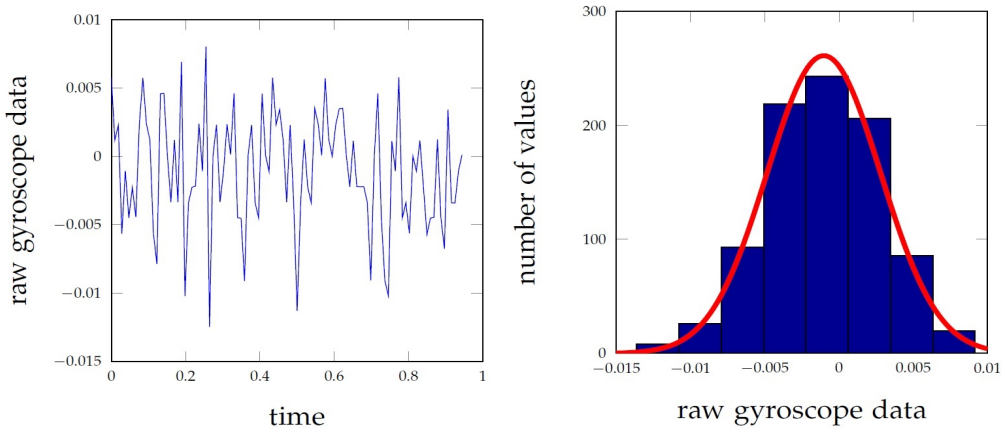


Figure 3.2 : Gyroscope data taken from EuRoC MAV Dataset. Although the drone is in a static state, the sensor data is corrupted by noise. This noise follows the normal distribution.

3.2.1 The one-dimensional case

In one dimension, we have the following formula for the (pdf) which describes the normal distribution:

$$p(x;\mu,\delta) = \underbrace{\frac{1}{\sqrt{2\pi\delta^2}}}_{\text{normalization term}} \exp\left(-\frac{(x-\mu)^2}{2\delta^2}\right) \tag{3.4}$$

The mean, μ, and variance, δ^2, are the parameters which decide the shape of the Gaussian. The overall probability (the area beneath the curve) is assured to be equivalent to 1 through the normalization term. Figure 3.3 shows the influence of various values for the mean and variance in the one-dimensional state. Small values of the variance form a narrow and high peak around the mean, which produces a high probability for x to be near μ. And for this reason the variance in many cases is referred to as the uncertainty.

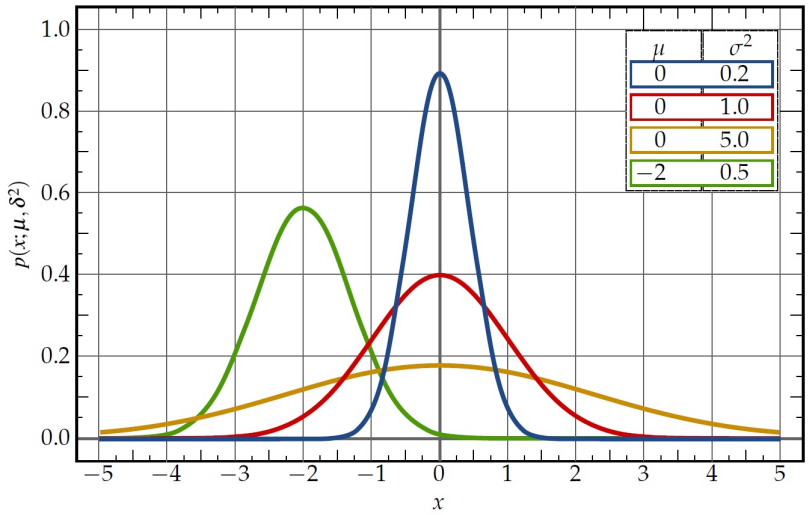


Figure 3.3 : The normal distribution forms a bell-shaped curve centered at the mean, μ. Large values of the standard deviation δ, make the curve spread wider while small values produce in a narrow shape.

3.2.2 The multidimensional case

In this case the true unknown state is introduced as a random multidimensional variable with a multivariate Gaussian distribution, x, for pose estimation, which denoted as $x \sim N(\mu, \Sigma)$. Where x and μ form $n \times 1$ vectors and Σ is an $n \times n$ matrix. for the multidimensional context, in the place of a single value for the uncertainty (δ^2) we have a full **covariance matrix** Σ. The latter's diagonal entries are just the variances of each element of x: $(\delta_{x1}^2, \delta_{x1}^2, \ldots \delta_{xn}^2)$. The non-diagonal entries are $\rho(x_a, x_b)\delta_{x_a}\delta_{x_b}$ where ρ represents the correlation.

$$Cov = E[(x - E[x])(y - E(y))] \tag{3.5}$$

$$Cov\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\right) = \begin{bmatrix} \delta_{x_1}^2 & \rho(x_1,x_2)\delta_{x_1}\delta_{x_2} & \ldots & \rho(x_1,x_n)\delta_{x_1}\delta_{x_n} \\ \rho(x_2,x_1)\delta_{x_2}\delta_{x_1} & \delta_{x_2}^2 & \ldots & \rho(x_2,x_n)\delta_{x_2}\delta_{x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \rho(x_n,x_1)\delta_{x_n}\delta_{x_1} & \rho(x_n,x_2)\delta_{x_n}\delta_{x_2} & \ldots & \delta_{x_n}^2 \end{bmatrix} \tag{3.6}$$

Afterwards, we will be concerned with the n-dimensional multivariable function of Gaussian probability density. With attention that in conduct, the Gaussian pdf is not assessed in an explicit way. That is due to our interested which is limited to the mean and covariance of x, and as we are going to observe, the Gaussian distribution nature give us the opportunity to conserve and update the values in in the course of the filtering operation.

$$p_n(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^2 |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) \tag{3.7}$$

where $|\Sigma|$ is the Σ determinant.

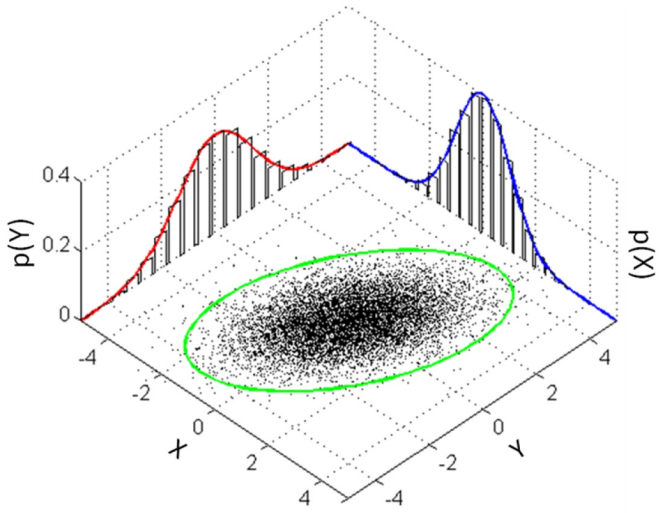


Figure 3.4 : The normal distribution a two dimensional case

3.2.3 Operations on a Gaussian

In this part we will talk about essential tools which help us while working with Gaussian random variables. The distinguished characteristics of the normal distribution permit particular applications on Gaussians to be evaluated in closed manner (i.e. without estimation) and with solely knowing the mean and covariance. These operations are marginalization, conditioning and intersection [1]. Let us examine a two-dimensional Gaussian random variable $(x,y)^T \sim N(\mu,\Sigma)$. By partitioning the mean and covariance matrix, we can rephrase this equation and thus we obtain the below joint probability distribution.

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim N(\mu,\Sigma) = N\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}\right) \tag{3.8}$$

We denote the partitions of Σ in uppercase letters even though it is 1×1 matrix and that to illustrate that the succeeding operations are scalable to multiple dimensions for x and y. In the two-dimensional case, $\Sigma_{xx} = \delta_x^2$. And the pdf shall acquire the below form (from equation 3.7).

$$p\left(\begin{bmatrix} x \\ y \end{bmatrix}; \mu, \Sigma\right) = \eta \exp\left(-\frac{1}{2}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}\right)^T \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}^{-1} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}\right)\right) \tag{3.9}$$

$$\text{Where } \eta = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} \text{ and } \Sigma_{xy} = \Sigma_{yx}^T$$

[1] Not any intersection, but intersection of a random variable with a linear combination of this variable

3.2.3.1 Marginalization

It is a spacial case in which we have the joint pdf of x and y but we are just concerned with the one of x. If we want to detect the pdf of x only, a process of marginalizing out y will be needed. Here we can examine the advantage of the normal distribution as p(x) is easy to find with knowledge of μ and Σ. We discarded the proof for briefness.

$$\begin{aligned} p(x) &= \int p(x,y)dy \\ &= \int p(x|y)p(y)dy \\ &= N(\mu_x, \Sigma_{xx}) \end{aligned} \tag{3.10}$$

3.2.3.2 Conditioning

Some other popular case is when have the joint pdf of x and y, and are defined with a certain value $y = y_0$. In such situation, the pdf of x could be obtain using conditioning. The fundamental idea is that both marginalization and conditioning are able to be constructed in closed fashion with the mean and covariance of the multivariate Gaussian.

$$x|_{y=y_0} \sim N(\mu_x + \underbrace{\Sigma_{xy}\Sigma_{yy}^{-1}(y_0 - \mu_y)}_{\text{mean offset}}, \Sigma_{xx} - \underbrace{\Sigma_{xy}\,\Sigma_{yy}^{-1}\,\Sigma_{xy}}_{\text{covariance offset}}) \tag{3.11}$$

We need to consider that the recent mean for x has been slightly moved in accordance with the correlation between x and y and how near y_0 was in respect to the mean μ_y. Alternatively stated, if $y_0 = \mu_y$ then the mean μ_x stays unchanged. In a similar way, when there is no correlation between, the off-diagonal entries of Σ, Σ_{xy} will be 0, which render the recent updated covariance plainly Σ_{xx}. And when there is correlation between them, having the value of y known shall automatically reduce the uncertainty of x.

3.2.3.3 Intersection

When the $x \sim N(\mu_x, \Sigma_{xx})$ is acknowledged and y is a linearly dependent on x, the joint probability p(x, y), also called the intersection, can be simply assessed, . Let $y = Ax + b$ where A is a constant and $b \sim N(0, Q)$.

$$\begin{aligned} p(\begin{bmatrix} x \\ y \end{bmatrix}) &= N(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}) \\ &= N(\begin{bmatrix} \mu_x \\ A\mu_x + b \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xx}A^T \\ A\Sigma_{xx} & A\Sigma_{xx}A^T + Q \end{bmatrix}) \end{aligned} \tag{3.12}$$

3.3 The Kalman Filter

In this section the Kalman filter will be discussed along with its derivations utilizing the formerly described operations on Gaussians. Kalman Filters are exceedingly practical for estimating the pose of a linear system. For this intention, the pose of the drone will be presumed to change linearly in respect to certain mobile model. For instance, a car on a level-plane with no recognized forces applied on it will ultimately decelerate and halt. if we know certain information about the dynamics of the car and using certain classical physics laws, we will be able to compute the probability density of the following state with knowledge of the present state. We describe the linear relationship between the following state and the present state as $x_{t+1} = A_t x_t$. This data could be utilized to propagate the PDF. If $x_t \sim N(\mu_t, \Sigma_t)$ and $x_{t+1} = A_t x_t$ then using intersection, $x_{t+1} \sim N(A_t \mu_t, A_t \Sigma_t A_t^T)$.

The more data regarding how x varies, like an activity or a measurement that is not correlated with the present state, the more our capability to utilize this information to propagate the probability increases. Assuming that the parameter u describes the velocity of a car, it is both independent of the present state (if the state considered to be the position) and it influences the following state. In several applications of the Kalman filter, u is known as the control input and defines activities like the movement commands forwarded to a robot. It is very likely, nevertheless, that u denotes a sensor reading, as far as it is both not dependent of x and possesses a linear relationship with it. In any case, an inherent quantity of noise there will present in the model $\varepsilon \sim (0, Q)$, a diminutive unidentified value which will rise the uncertainty in estimating the pose.

$$x \sim N(\mu_t, \Sigma_t) \tag{3.13}$$

$$x_{t+1} = A_t x_t + B_t u_t + \varepsilon_t \tag{3.14}$$

For the sake of determining $p(x_{t+1})$, initially we employ intersection to calculate $p(x_{t+1}, x_t)$. Afterwards, we marginalize out x_t.

$$
\begin{aligned}
p(x_{t+1}, x_t) &= N\left(\begin{bmatrix}\mu_t \\ A_t\mu_t + B_t u_t\end{bmatrix}, \begin{bmatrix}\Sigma_t & \Sigma_t A_t^T \\ A_t\Sigma_t & A_t\Sigma_t A_t^T + Q_t\end{bmatrix}\right) \\
x_{t+1} &\sim N(A_t\mu_t + B_t u_t, A_t\Sigma_t A_t^T + Q_t) \\
x_{t+1} &\sim N(\mu_{t+1}, \Sigma_{t+1})
\end{aligned} \tag{3.15}
$$

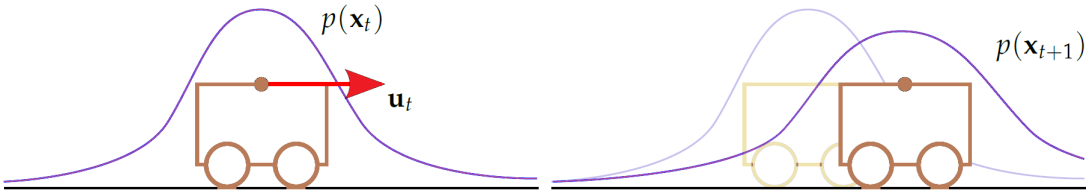


Figure 3.5 : The pdf for the car position varies in respect to a ordinary motion model. The pdf contracts vertically and extends horizontally due to the fact that uncertainty in the motion model is plainly interpreted into uncertainty in the state.

As time passes, the motion model is going to rise the uncertainty of the state estimate indeterminately. To decrease the uncertainty, explicit measurements of the state are needed. Such a measurement, z, could be utilized to update the pdf of x, presuming that the linear relationship of the true values $z = Cx + \delta$ is in hand, where $\delta \sim N(0, R)$ is the measurement noise. Hence by determining the intersection $p(x_t, z_t)$ and afterwards applying conditioning (equation 3.11) to figure out the updated covariance and mean of x_t.

$$
\begin{aligned}
p(x_t, z_t) &= N\left(\begin{bmatrix}\mu_t \\ C_t\mu_t\end{bmatrix}, \begin{bmatrix}\Sigma_t & \Sigma_t C_t^T \\ C_t\Sigma_t & C_t\Sigma_t C_t^T + R_t\end{bmatrix}\right) \\
x_t|_{z_t=z_0} &\sim N(\mu_t + K_t(z_0 - C_t\mu_t), \Sigma_t - K_t C_t\Sigma_t) \\
K_t &= \Sigma_t C_t^T (C_t\Sigma_t C_t^T + R_t)^{-1}
\end{aligned} \tag{3.16}
$$

A significant presumption in pose estimation is that every state is evaluated solely based on the prior state and the control input, u. In the same manner, observations dependent merely on the present state. Presuming independence from former states is perceived as the **Markov Assumption** which is a major factor in filtering algorithms.

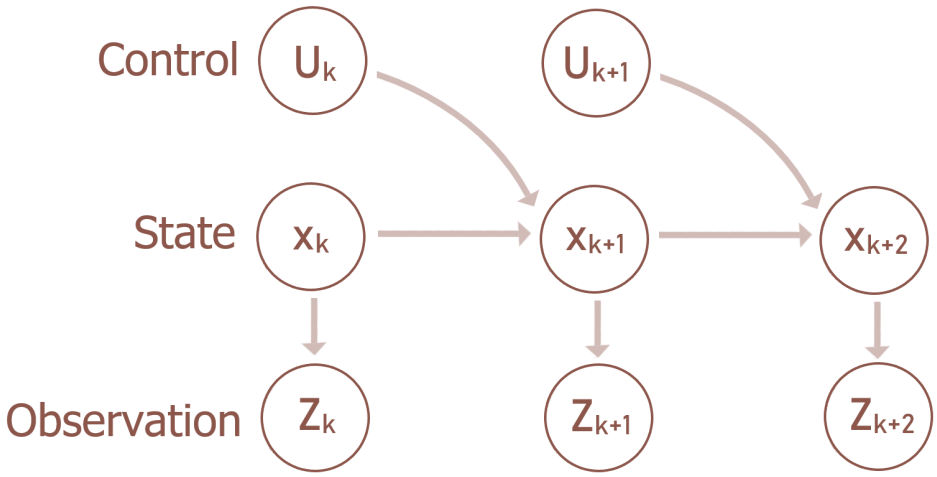


Figure 3.6 : The Markov assumption simplifies recursive computations by affirming independence from prior states and measurements.

3.3.1 Kalman filter summary

Here a little bit different denotation will be employed to stay persistent with the rest of the thesis. Particularly, in place of μ the mean of the state pdf, we utilize the estimate $\hat{x}$. Moreover, we will also employ the following notation $\hat{x}_{t|t-1}$, which means "the state estimate at time t knowing measurements from time 0 through $t-1$".

Initial Estimate

$$x_0 \sim N(\hat{x}_{0|0}, \Sigma_{0|0}) \tag{3.17}$$

Prediction Step

$$\begin{aligned} &\text{Given: } x_{t+1} = A_t x_t + B_t u_t + \varepsilon_t \text{ where } \varepsilon_t \sim N(0, Q_t) \\ &\hat{x}_{t+1|t} = A_t \hat{x}_{t|t} + B_t u_t \\ &\Sigma_{t+1|t} = A_t \Sigma_{t|t} A_t^T + Q_t \end{aligned} \tag{3.18}$$

Update Step

$$\begin{aligned} &\text{Given: } z_t = C_t x_t + \delta_t \text{ where } \delta_t \sim N(0, R_t) \\ &\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - C_t \hat{x}_{t|t-1}) \\ &\Sigma_{t|t} = \Sigma_{t|t-1} - K_t C_t \Sigma_{t|t-1} \\ &K_t = \Sigma_{t|t-1} C_t^T (C_t \Sigma_{t|t-1} C_t^T + R_t)^{-1} \end{aligned} \tag{3.19}$$

3.4 Example

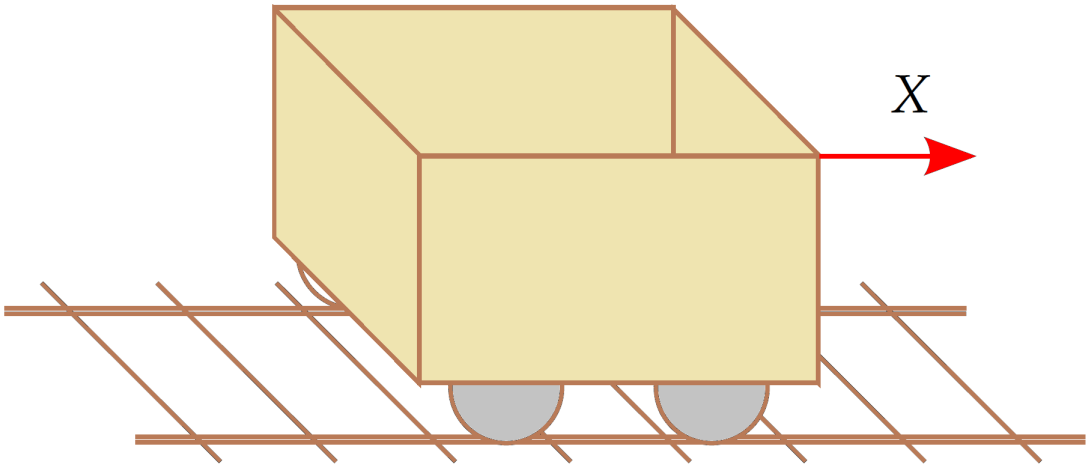


Figure 3.7 : A simple object which is solely able to perform a translation movement on one axis.

A simple car is merely able to perform a one-dimensional back and forth movement (Figure 3.7). Then we would like to do an estimation of this object's position and velocity utilizing certain simple sensors. We primary pick the state representation. $x = (p,v)^T$, where p and v are the object's position and velocity respectively. The primary state is $x_0 = 0_{2\times 1}$. There is a single-axis accelerometer mounted on the car that operates at 100Hz, that assesses a, the linear acceleration of the device. This reading will be exploited in the prediction process of the Kalman Filter. In this example, the accelerometer readings are 1-D, bias free, and corrupted by noise $n_a \sim N(0, \delta_a^2)$. Primary the system dynamics must be determined.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ a \end{bmatrix} \\ p_{t+1} &= p_t + \int_t^{t+1} v_\tau d_\tau \simeq p_t + v_t \Delta_t \\ v_{t+1} &= v_t + \int_t^{t+1} a_\tau d_\tau \simeq v_t + a_t \Delta_t \end{aligned} \tag{3.20}$$

Where $\dot{x}$ is the continuous time derivative of the state. It can be easily perceived that this is a linear system. Thus, the prediction stage could be evaluated as below:

$$\begin{aligned}
\hat{\boldsymbol{x}}_{t+1|t} &= \boldsymbol{A}_t \hat{\boldsymbol{x}}_{t|t} + \boldsymbol{B}_t \boldsymbol{u}_t \\
\begin{bmatrix} \hat{p}_{t+1|t} \\ \hat{v}_{t+1|t} \end{bmatrix} &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{p}_{t|t} \\ \hat{v}_{t|t} \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} a_{m_t} \\
\boldsymbol{\Sigma}_{t+t|t} &= \boldsymbol{A}_t \boldsymbol{\Sigma}_{t|t} \boldsymbol{A}_t^T + \boldsymbol{Q}_t \\
\boldsymbol{Q}_t &= \begin{bmatrix} 0 & 0 \\ 0 & \delta_a^2 \end{bmatrix}
\end{aligned} \tag{3.21}$$

Sometimes we can get some readings of the velocity with a speed-meter. And this is due to the fact that the velocity is included in the state vector, this reading should be utilized to update the step of the Kalman filter. If z_t is the velocity evaluated at time t along with some noise corruption, and has the variance δ^2. The matrix $\boldsymbol{C}$ identifies the linear relationship between x and z, which is quite straightforward since the velocity is included in the state vector.

$$\begin{aligned}
\boldsymbol{z}_t &= \boldsymbol{C}_t \boldsymbol{x} + \delta \\
&\text{where } \delta \sim N(\boldsymbol{0}, \boldsymbol{R}) \text{ and } \boldsymbol{R}_t = \delta_v^2 \\
\boldsymbol{z}_t &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + \delta_t
\end{aligned} \tag{3.22}$$

The update process of Kalman filter see the difference between the assessed value $\boldsymbol{z}_t$ and the value it would predict based on the state estimate, $\boldsymbol{C}_t \hat{\boldsymbol{x}}_t$. Afterwards, The Kalman filter update the process as below:

$$\begin{aligned}
\hat{\boldsymbol{x}}_{t|t} &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t (\boldsymbol{z}_t - \boldsymbol{C}_t \hat{\boldsymbol{x}}_{t|t-1}) \\
\boldsymbol{\Sigma}_{t|t} &= \boldsymbol{\Sigma}_{t|t-1} - \boldsymbol{K}_t \boldsymbol{C}_t \boldsymbol{\Sigma}_{t|t-1} \\
\text{with } \boldsymbol{K}_t &= \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{C}_t^T (\boldsymbol{C}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{C}_t^T + \boldsymbol{R}_t)^{-1}
\end{aligned} \tag{3.23}$$

3.4.1 Example simulation

Here a basic simulation of the car example can be perceived. The car can move forward and backward along a ordinary sinusoidal wave pattern at the same time its pose is being estimated with KF.

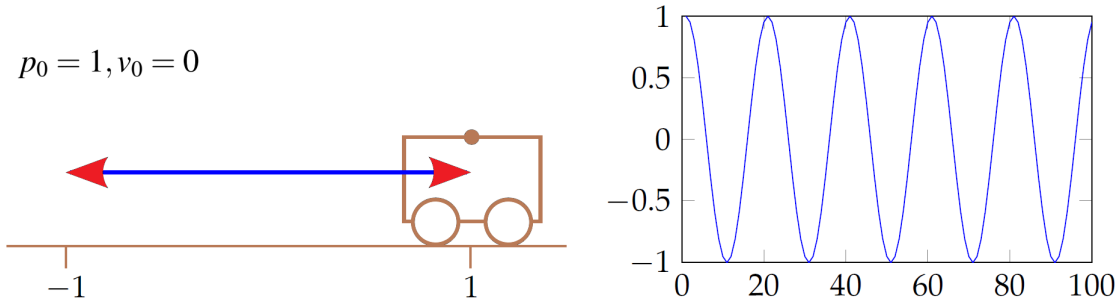


Figure 3.8 : The position of the car in time.

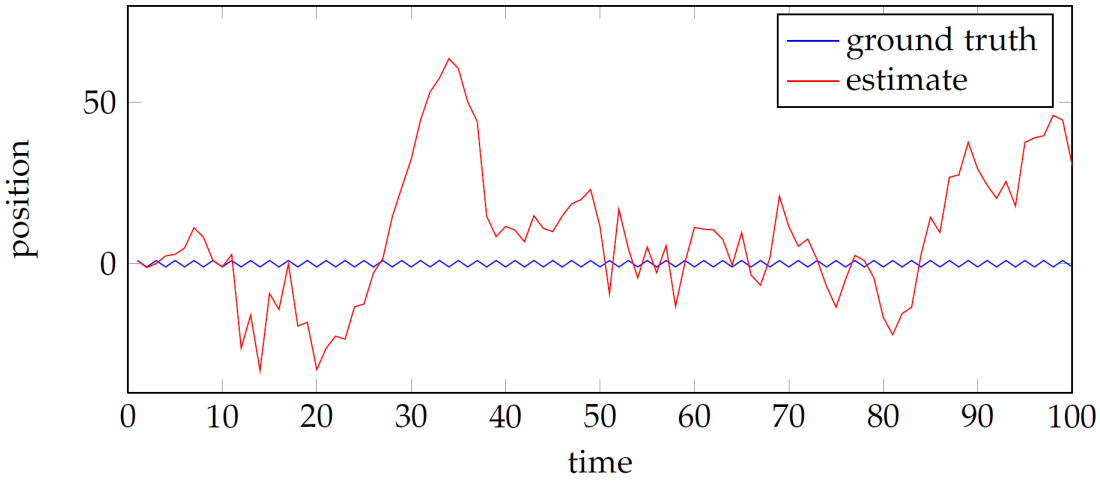


Figure 3.9 : The position error is enormous and increase even more in time. This is due to unobservability in the position.

The estimate pose produced after a prolonged period seems to look quite incorrect. Nevertheless, if we perceive the error in comparison with the conserved uncertainty (Figure 3.9), it is easy to observe that it stays well-bounded in 3δ, with δ being the square root of the corresponding diagonal index in Σ. Due to the fact that the velocity is instantly observed, the uncertainty is conserved at a coherently low-level. The uncertainty of the position, on the other hand, is unbounded. Owing to unobservability in the position in this example.

3.5 Observability

The system is said to be observable if we can identify the behavior of the whole system with just outputs provided. With respect to KF, the measurement vector z is perceived as the output of the system. The system can be observable if these outputs are sufficient

to conclude the true state. Sometimes it possible that the system is solely partially observable.

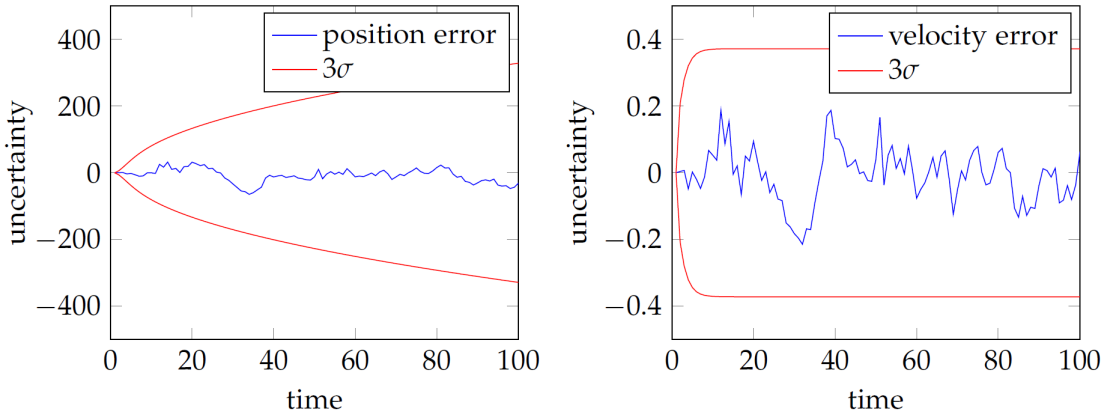


Figure 3.10 : The error in the position and velocity of the car stay in the bounds of 3 standard deviations. The velocity is observable and the uncertainty stays bounded, however, the position is unobservable, inducing the uncertainty growth with time.

In other words, the system is said to be observable at time t_k if the state vector x_k is able to be concluded with measurements $z = \{z_0, z_1, ..., z_k\}$, with $t_0 < t - k < \infty$. To figure out whether a system is observable or not, we form and investigate the observability matrix, O_k. Afterwards, if $O_k^T O_k$ possesses full rank, the system is observable.

$$O_k = \begin{bmatrix} C_0 \\ C_1 A_0 \\ C_2 A_1 A_2 \\ \vdots \\ C_k A_{k-1} \dots A_0 \end{bmatrix} \tag{3.24}$$

$$O_k^T O_k = C_0^T C_0 + \sum_{i=1}^{k} A_{i-1,0}^T C_i^T C_i A_{i-1,0} \tag{3.25}$$

It is simple to illustrate that the car example mentioned is not an observable system, owing to its rank of $O_k^T O_k$ for any k is 1. It can be perceived that the velocity is observable as well, whereas the position is not. To prove that, we can easily inspect the columns of the observability matrix seperately. For a specific row i of O,

$$O^{(i)} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0$$

Since this is valid for each value of i, it can be perceived that the position is unobservable because it belongs to the null space of O. In addition, the same is not not

valid for the velocity.

$$O^{(i)} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1$$

In Figure 3.10 we can perceive the evident difference between observable and unobservable variables in a system. Even though the position is able to be estimated, the uncertainty is unbounded and will increase infinitely. On the contrary the velocity is observable, which is anticipated knowing direct velocity measurements, however direct measurements are not needed for observability. Furthermore, observability is required but not enough to guarantee a bounded uncertainty. For instance, large values for the noise matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ might be the reason behind the uncertainty growth to infinity.

3.6 Extended Kalman Filter

Mainly we are concerned in estimating the pose of a system that possesses a full potential range of mobility. In other words 3 degrees of freedom in terms of position, and 3 in terms of the orientation. Pose estimation with KF is unachievable since the dynamics are nonlinear. To address this issue, we will present the nonlinear Extended Kalman Filter (EKF). The major differences are about the dynamical motion model and the sensor model assumptions.

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, w_t) & w_t &\sim N(0, Q_t) \\ z_t &= h(x_t, v_t) & v_t &\sim N(0, R_t) \end{aligned} \tag{3.26}$$

Both f and h are nonlinear functions. We are not able to easily find the intersection of x_{t+1} and x_t anymore as in KF because of nonlinearity. The EKF solve this issue by performing linearization in the vicinity of the present estimate $\hat{x}_t$. To achieve this, we utilize the **Taylor Series Expansion**: any differentiable function $f(x)$ is able to be estimated around x with a series as follows:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)(x-a)^2}{2!} + \frac{f'''(a)(x-a)^3}{3!} + \dots \tag{3.27}$$

The terms in the series diminish in value and importance at higher orders and for ease we can neglect them. We will utilize solely the terms up to the first order, which makes the Extended Kalman Filter a 1st-order estimator.

3.6.1 Prediction step

Let us examine the function $f(x,u,w)$ at time t. The Taylor series expansion yields:

$$\begin{aligned} f(x_t,u_t,w_t) &\approx f(\hat{x}_t,u_t) + f'(\hat{x}_t,u_t)(xt-\hat{x}_t) + \dots \\ &\approx f(\hat{x}_t,u_t) + F(\hat{x}_t,u_t)\tilde{x}_t \end{aligned} \tag{3.28}$$

Where $\tilde{x}$ is the state error, which the variance between the true state and the state estimate, while F is the Jacobian matrix for f. If $f(x) = (f_1(x), f_2(x), \vdots, f_n(x))^T$ and $x = (x_1, x_2, \vdots, x_n)^T$

$$F = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots \frac{\partial f_n}{\partial x_n} & \end{bmatrix} \tag{3.29}$$

Then we will be able to propagate the state estimate for the EKF prediction step. By equating equation 3.26 with 3.28, we will perceive that the following state is possible to be estimated.

$$\hat{x}_{t+1} = f(\hat{x}_t, u_t) \tag{3.30}$$

The propagated error may now be introduced as a linear combination of the former error and the noise. We write $F(\hat{x}_t,u_t)$ as F_t for readability.

$$\tilde{x}_{t+1} = F_t\tilde{x}_t + G_t w_t \text{with } \tilde{x}_{t+1} = x_{t+1} - \hat{x}_{t+1} \tag{3.31}$$

The matrix G is the Jacobian of f W.R.T the noise. It also helps to transmute the noise vector into the same dimensionality as the state error. Utilizing the formerly stated definition of the covariance, $\text{Cov}(x,y) = E[(x-E[x])(y-E(y))]$, then it is possible to determine the propagated covariance matrix at time $t+1$.

$$\begin{aligned} \Sigma_{t+1} &= E[(x_{t+1}-\hat{x}_{t+1})(x_{t+1}-\hat{x}_{t+1})^T] \\ &= E[\tilde{x}_{t+1}\tilde{x}_{t+1}^T] \\ &= E[F_t\tilde{x}_t\tilde{x}_t^T F_t^T] + E[G_t w_t w_t^T G_t^T] \\ &= F_t E[\tilde{x}_t\tilde{x}_t^T]F_t^T + G_t E[w_t w_t^T]G_t^T \\ &= F_t\Sigma_t F_t^T + G_t Q_t G_t^T \end{aligned} \tag{3.32}$$

3.6.2 Discrete vs. continuous filtering

Up to this point only the continuous time filtering has been addressed. However, in real systems we encounter discrete nonlinear time sensitive measurements. The method for discretization changes based on system characteristics. Presumptions considered, if existing, throughout discretization of the propagated state estimate will have no

impact on the uncertainty. Thus, whenever feasible, it would be better to identify an analytically closed-form solution. In case that was not achievable, numerical integration might be a practicable alternative. Either way, we would like to determine the matrix Φ that complies with

$$\tilde{x}_t + 1 = \Phi_t \tilde{x}_t + w_{d_t} \tag{3.33}$$

with discrete noise $w_{d_t} \sim N(0, Q_d)$. This might be achieved by resolving the differential equation $\dot{\Phi}(t, t_i) = F(t)\Phi(t, t_i)$ from $t \in [t, t+1]$ along with initial condition $\Phi_t = I$. It is necessary for the noise covariance to be discretized as shown below, where Q_c is the covariance matrix of the continuous noise vector.

$$Q_d = \int_t^{t+1} \Phi(t+1, \tau) G(\tau) Q_c G(\tau)^T \Phi(t+1, \tau)^T d\tau \tag{3.34}$$

3.6.3 Update step

Knowing a reading $z_t = h(x_t) + v_t$, it is possible to make benefit again of the Taylor expansion to determine $h(x_t)$ with the present estimate.

$$\begin{aligned} h(x_t) &= h(\hat{x}_t) + H(\hat{x}_t)(x_t - \hat{x}_t) + \dots \\ h(x_t) &\approx h(\hat{x}_t) + H(\tilde{x}_t)\tilde{x}_t \end{aligned} \tag{3.35}$$

Here H is the Jacobian of h W.R.T x (equation 3.29). The Jacobian H is possible then to be utilized in the EKF update equations. It can be yielded from equation 3.35 that

$$\begin{aligned} h(x_t) - h(\hat{x}_t) &\approx H_t \tilde{x} \\ z_t - h(\hat{x}_t) &\approx H_t \tilde{x} - v_t \\ r_t &= H_t \tilde{x}_t + n_t \end{aligned} \tag{3.36}$$

Here r is the **residual** vector, the difference between the measurement and hypothesis, and n refers to the noise. This is a significant equation which illustrates that the residual should be a linear combination of the state error and noise. The update now advances in a alike fashion to the one of the normal KF.

$$\begin{aligned} \tilde{x}_{t|t} &= \tilde{x}_{t|t-1} + K_t(z_t - h(t|\tilde{t-1})) \\ \Sigma_{t|t} &= \Sigma_{t|t-1} - K_t H_t \Sigma_{t|t-1} \\ K_t &= \Sigma_{t|t-1} H_t^T (H_t \Sigma_{t|t-1} H_t^T + R_t)^{-1} \end{aligned} \tag{3.37}$$

The derivation of the EKF update step is resembling in concept the prediction step. But since it is complicated and not directly related to this thesis, we omit it entirely. For more information should refer to Thrun's Probabilistic Robotics [28]

3.6.4 EKF summary

The last form of EKF equations appears to be quite relevant to the normal KF. Actually, if the functions f and h are linear, the Extended Kalman Filter turns back to be a regular Kalman Filter. **Initial Estimate**

$$x_0 \sim N(\hat{x}_{0|0}, \Sigma_{0|0}) \tag{3.38}$$

Prediction Step

$$\begin{aligned} \text{Given: } x_{t+1} &= f(x_t + u_t) + w_t \text{ where } w_t \sim N(0, Q_t) \\ \hat{x}_{t+1|t} &= f(\hat{x}_{t|t} + u_t) \\ F_t &= \frac{\partial f}{\partial x}(x_t, u_t) \\ \Sigma_{t+1|t} &= F_t \Sigma_{t|t} F_t^T + Q_t \end{aligned} \tag{3.39}$$

Update Step

$$\begin{aligned} \text{Given: } z_t &= h(x_t) + v_t \text{ where } \delta_t \sim N(0, R_t) \\ \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t(z_t - h(\hat{x}_{t|t-1})) \\ H_t &= \frac{\partial h}{\partial x}(x_t, u_t) \\ \Sigma_{t|t} &= \Sigma_{t|t-1} - K_t H_t \Sigma_{t|t-1} \\ K_t &= \Sigma_{t|t-1} H_t^T (H_t \Sigma_{t|t-1} H_t^T + R_t)^{-1} \end{aligned} \tag{3.40}$$

4. IMU (INERTIAL MEASUREMENT UNIT)

The IMU (Inertial Measurement Unit) is a widely utilized electronique device generally composed mainly of gyroscopes, accelerometers, and in some cases magnetometers. Recent IMU's are inexpensive to manufacture and are common in, especially, AV, and smart-phones. Our main concern are the gyroscope and accelerometer and this is due to their noise properties. The magnetometer (Triple-axis compass) is highly affected by magnetic-field around it which makes it difficult to be modelled.

Figure 4.1 : Various examples of devices that utilize low-cost MEMS IMU's such as smart-phone, Oculus Rift virtual reality, and Hubsan Drone.

4.1 Accelerometer

The accelerometer measures the change in gravitational acceleration in the devices in which it is equipped. It is mostly utilized to read acceleration, tilt and vibration in various devices.

At the rest condition, the accelerometer measures 1-G: which is the gravitational pull of the earth, and that roughly equals to 9.81 $\frac{m}{s^2}$ or 32.185 feet $\frac{ft}{s^2}$. Some of the accelerometers employ the piezoelectric effect and measure voltage variation. Others measure capacitance between two components.

To measure motion in multi-directions, an accelerometer should be provided with multi-axis sensors or multiple linear-axis sensors. Three linear accelerometers are adequate to measure motion in 3-D.

These three axis devices can be seen in many of the latest devices, e.g. smart-phones, quadrotors, and virtual-reality headsets. A standard accelerometer can provide

measurement as below.

$$^{B}\mathbf{a}_{m} = {}^{B}_{G}\mathbf{R}\,(^{G}\mathbf{a} - {}^{G}\mathbf{g}) \tag{4.1}$$

Where $^{G}\mathbf{a}$ is the true acceleration of the IMU in the global frame and B denotes the inertial body (IMU) frame. The global reference frame places the z-axis upwards far from the earth in a way that $\boldsymbol{g} = (0,0,-1)^{T}$. The subscript m of $\mathbf{a}_{m}$ means that it is the measurement. The unit of measurement of the accelerometer is g. Low-cost accelerometers, such as:

tho ones in smart-phones, are not very reliable and the measurements they provide are usually associated by noise and bias.

$$^{B}\mathbf{a}_{m} = {}^{B}_{G}\mathbf{R}\,(^{G}\mathbf{a} - {}^{G}\mathbf{g}) + \boldsymbol{n}_{a} + \boldsymbol{b}_{a} \tag{4.2}$$

The noise $\boldsymbol{n}_{a}$ represents a zero-mean normally distributed random variable, $\boldsymbol{n}_{a} \sim N(0,\boldsymbol{N}_{a})$. The bias, $\boldsymbol{b}_{a}$, varies with time and is modelled as a random walk process driven by its own noise vector $\boldsymbol{n}_{wa} \sim N(0,\boldsymbol{N}_{wa})$. Accelerometers sometimes also endure some misalignment and scale errors.

$$^{B}\mathbf{a}_{m} = \boldsymbol{T}_{a}\,{}^{B}_{G}\mathbf{R}\,(^{G}\mathbf{a} - {}^{G}\mathbf{g}) + \boldsymbol{n}_{a} + \boldsymbol{b}_{a} \tag{4.3}$$

Where $\boldsymbol{T}_{a}$ is the shape matrix which induces both misalignment and scale errors in the accelerometer measurements.

4.2 Gyroscope

The gyroscope measures ω, the angular velocity of the IMU. Similar to the accelerometer, it experiences noise, bias, misalignment errors and scale errors.

$$^{B}\boldsymbol{\omega}_{m} = \boldsymbol{T}_{g}\;{}^{B}\boldsymbol{\omega}_{m} + \boldsymbol{n}_{g} + \boldsymbol{b}_{g} \tag{4.4}$$

The noise $\boldsymbol{n}_{g}$ is Gaussian white noise like $\boldsymbol{n}_{g} \sim N(0,\boldsymbol{N}_{g})$. The bias, $\boldsymbol{b}_{g}$, is able to be modelled as a random walk process guided by the noise vector $\boldsymbol{n}_{wg} \sim N(0,\boldsymbol{N}_{wg})$. Gyroscopes usually get affected by acceleration in something known as g-sensitivity. The magnitude of this effect is deemed negligible if it falls under the range of the cumulative white noise. However, in certain hardware, specifically the low-cost MEMS gyroscopes, it is further substantial and able to be modelled.

$$^{B}\boldsymbol{\omega}_{m} = \boldsymbol{T}_{g}\,{}^{B}\boldsymbol{\omega}_{m} + \boldsymbol{T}_{s}\,{}^{B}\mathbf{a}_{m} + \boldsymbol{n}_{g} + \boldsymbol{b}_{g} \tag{4.5}$$

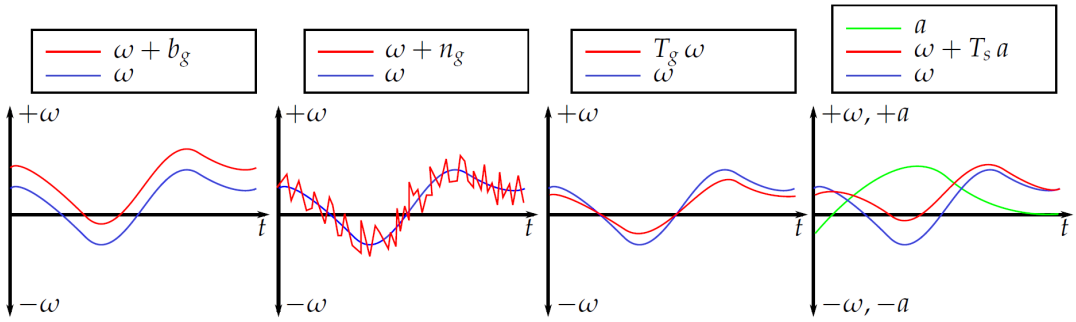


Figure 4.2 : Representative examples of several varieties of noise which can be seen on an MEMS gyroscope. From left to right: bias, white noise, scale factor, and g-sensitivity. Noting that the units for ω and a are different.

4.3 Noise and Bias Properties

In order to utilize the measurements we get from the accelerometer and gyroscope, we need first to compute the noise differences $(\mathbf{N}_a, \mathbf{N}_g)$ and random walk differences $(\mathbf{N}_{wa}, \mathbf{N}_{wg})$. The subsequent part studies a sole axis of the accelerometer. However, it can be generalized to all accelerometer axes and to the gyroscope. The accelerometer noise variance is able to be inferred by computing the standard deviation of a consecutive group of measurements when the sensor is at rest position for a short period of time, t. If the time segment is rather long, the varying bias could corrupt the readings and falsely enlarge the outcome. The measured standard deviation is discrete, δ_{a_d}, and is not the intended value for the EKF prediction step. To get the continuous values, we follow [29] and propagate by the sample noise.

$$\delta_{a_c} = \delta_{a_d}\sqrt{\Delta t} \tag{4.6}$$

In the same manner the random walk bias standard deviation should be transformed to a continuous value as well.

$$\delta_{\omega a_c} = \frac{\delta_{\omega a_d}}{\sqrt{\Delta t}} \tag{4.7}$$

The obtained continuous values are able now to be utilized in the noise matrix for the EKF prediction step.

5. OPTICAL FLOW/COMPUTER VISION BASICS

Motion analysis is one of the main tasks of computer vision [30]. From an application point of view, the data obtained by the dynamical behavior of observed objects or by the motion of the camera itself is a crucial element for the interpretation of observed phenomena. The motion characterizations can be extremely variable among the large number of application domains. Of course, it is interesting to track objects, quantify deformations, retrieve dominant motion, detect abnormal behaviors, and so on. The most low level characterization is the estimation of a dense motion field, corresponding to the displacement of each pixel, which is known as optical flow. Most high level movement analysis tasks utilize optical flow as a essential basis upon which more semantic interpretation is built.

On the other hand, computer vision is a fast growing field on the forefront of robotics and computer science. While the quality of the cameras is increasing, their dimensions and cost are declining. In addition, the number of open source tools available, introduced the area of computer vision as accessible and feasible for a broad range of applications. What made employing cameras as an odometry sensor such a fertile soil for so many researches especially in comparison with rather accurate and high-cost sensors. For this chapter, we explain how to utilize the camera images to obtain reliable structural data about the environment.

5.1 Triangulation

For this section, we will utilize 2-D pixel measurements to obtain 3-D positions. Let us suppose we have two distinct cameras at recognized places and they both take images of an identical 3-D object. In a perfect scenario, obtaining the 3-D position of the object is an easy problem of resolving a system of equations. But, in actuality, error is introduced in a several means resulting in a system that possesses no solution. The main aim is to determine the best guess for the location of the object. The best way to do that is to reformulate the problem as a minimization problem which is then able to

be solved with the Gauss-Newton algorithm, a technique utilized to resolve non-linear least-square problems [31]. To do this, we start by constructing the error function f.

$$f_i(\theta) = z_i - h(\theta) \tag{5.1}$$

Where here, θ is the variable, z_i is the measurement, and the function h projects the variable to the measurement space. A feature point, ${}^{G}\mathbf{p}_f$, is traced by a camera from n places (Figure 5.2). For a known camera frame, $\{C_i\}$, the feature position in that frame is ${}^{C_i}\mathbf{p}_f = ({}^{C_i}\mathrm{X}, {}^{C_i}\mathrm{Y}, {}^{C_i}\mathrm{Z})^T$. However, the camera can solely measure the pixel values $(u_i, v_i)^T$. The measurement is identified as $z_i = \mathbf{h}({}^{C_i}\mathrm{X}, {}^{C_i}\mathrm{Y}, {}^{C_i}\mathrm{Z})$, here $\mathbf{h}$ represents the projection function from the former section and z_i is in pixel coordinates. The frame $\{C_0\}$ is the camera frame from which the object was detected, and the position of the object in the ith camera frame is as below.

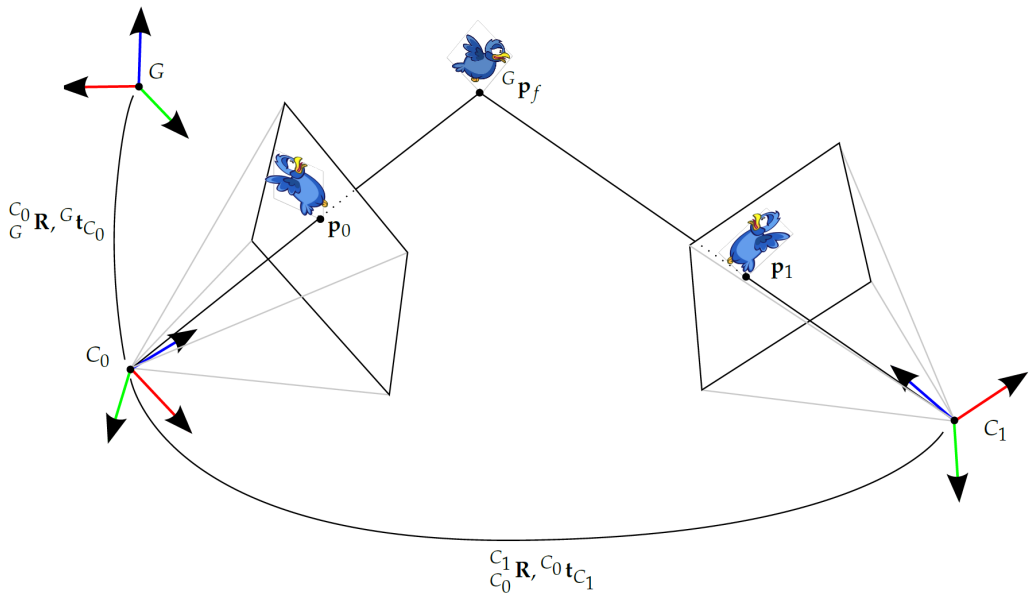


Figure 5.1 : an ideal case of triangulation with two objects in two images. In actuality, it is exceedingly implausible for the lines to intersect.

$$\begin{aligned} {}^{C_i}\mathbf{p}_f &= {}^{C_i}_{C_0}\mathbf{R}({}^{C_0}\mathbf{p}_f - {}^{C_0}\mathbf{p}_{C_i}) \\ {}^{C_i}\mathbf{p}_f &= {}^{C_i}_{C_0}\mathbf{R}\,{}^{C_0}\mathbf{p}_f - {}^{C_i}\mathbf{p}_{C_0} \end{aligned} \tag{5.2}$$

We are able to reformulate with the inverse depth characterization [14] so that we enhance the numerical stability which will assist to evade local minima.

$$
\begin{aligned}
{}^{C_i}\mathbf{p}_f &= {}^{C_i}_{C_0}\mathbf{R}\begin{bmatrix} {}^{C_n}\mathrm{X} \\ {}^{C_n}\mathrm{Y} \\ {}^{C_n}\mathrm{Z} \end{bmatrix} + {}^{C_i}\mathbf{p}_{C_0} \\
&= {}^{C_0}\mathrm{Z}\left({}^{C_i}_{C_0}\mathbf{R}\begin{bmatrix} {}^{C_0}\mathrm{X}/\,{}^{C_0}\mathrm{Z} \\ {}^{C_0}\mathrm{Y}/\,{}^{C_0}\mathrm{Z} \\ 1 \end{bmatrix} + \frac{1}{({}^{C_0}\mathrm{Z})}{}^{C_i}\mathbf{p}_{C_0} \right) \\
&= {}^{C_0}\mathrm{Z}\left({}^{C_i}_{C_0}\mathbf{R}\begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} \rho\, {}^{C_i}\mathbf{p}_{C_0} \right) \\
&= {}^{C_0}\mathrm{Z}_{g_i}\begin{pmatrix} \alpha \\ \beta \\ \rho \end{pmatrix}
\end{aligned}
\tag{5.3}
$$

The function g_i is a 3D function of α, β, and ρ.

$$
\alpha = {}^{C_0}\mathrm{X} \,/\, {}^{C_0}\mathrm{Z} \qquad \beta = {}^{C_0}\mathrm{Y} \,/\, {}^{C_0}\mathrm{Z} \qquad \rho = 1 \,/\, {}^{C_0}\mathrm{Z} \tag{5.4}
$$

Afterwards, we will be able to reformulate the general error function (equation 5.1) for according to our particular example. The point of this new equation is straightforward: the 3-D object, characterized by θ in frame C_0, is converted into frame C_i by g_i and projected into pixel coordinates by the function h where it is matched with the pixel measurement in that frame. The aim of the Gauss-Newton minimization algorithm is to minimize the error for all measurements by iteratively updating the parameters.

$$
\mathbf{f}_i(\boldsymbol{\theta}) = \mathbf{z}_i - \mathbf{h}(\mathbf{g}_i(\boldsymbol{\theta})) \; with \; \boldsymbol{\theta} = \begin{pmatrix} \alpha \\ \beta \\ \rho \end{pmatrix} \tag{5.5}
$$

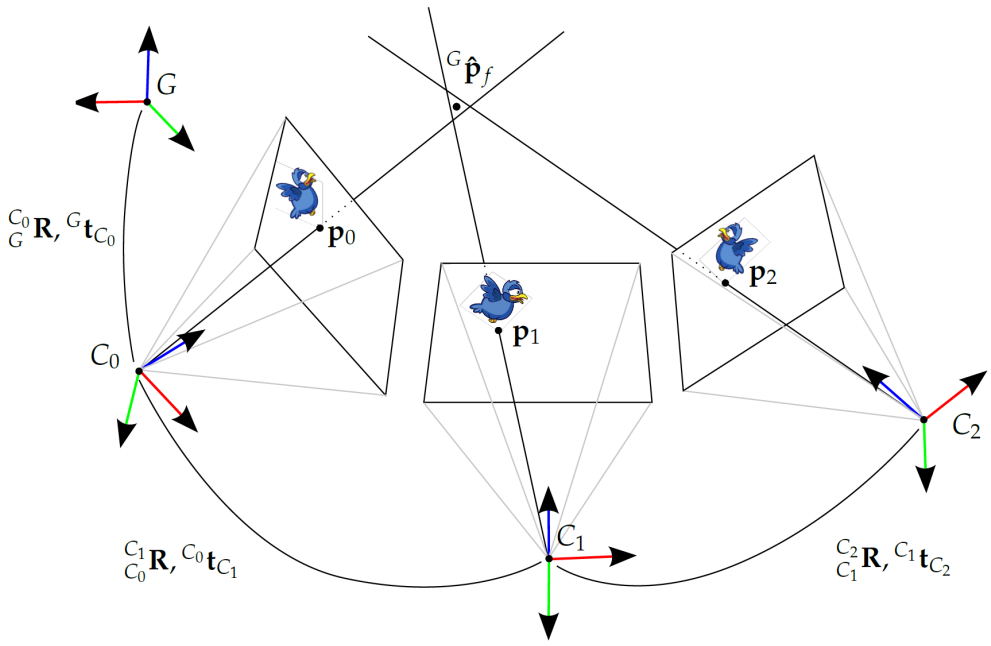


Figure 5.2 : Actual case triangulation acquire non-intersecting lines. Determining the best estimate reduces to a minimization problem.

5.2 Rolling Shutter

The majority of small-scale, low-cost cameras, particularly tho ones integrated with smart-phones, they possess rolling shutters property. Rather than capturing the whole image concurrently, every column (or row, according to the sensor) is captured consecutively in a specific time interval known as the read time. Lengthy read times could sometimes result in considerable image distortion and that may cause issues with triangulation. Various odometry approaches function under the presumption of a global shutter [12] since they are usually developed in laboratories where such cameras are used.

Figure 5.3 : The rolling shutter may result in visible distortion in the case of fast camera rotation. This figure is an example of an image captured by camera while moving in opposite directions.

When a rolling shutter camera captures an image, two temporal values have to be defines in order to utilize the image precisely. The 1st value is the offset t_d separating the time-stamp of the image and the time in which the central column has been captured. This value is often negative, which indicates that the time-stamp is late. However, in instances where video information and IMU information are entirely non-synchronized and solely the frame-rate is defined, t_d can technically take a positive value. The second significant calibration value is the read time of the camera, t_r. When this value is negative, it suggests that the rolling shutter in a motion from right to left. The approach that has been used in this thesis compute these two temporal calibration factors by estimating them in the EKF, as described in chapter 6.

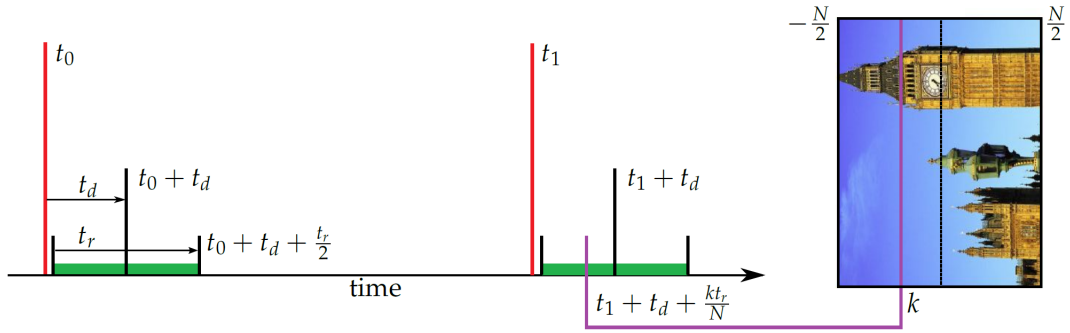


Figure 5.4 : When an image is taken, a time-stamp is submitted $(t_0, t_1, \ldots)$. The central column of the image has been taken at specific time $t + t_d$. The whole image of N columns possesses a read time of t_r, which indictes that indictes that column k has been captured at $t + t_d + \frac{kt_r}{N}$ where $k \in \left[-\frac{N}{2}, \frac{N}{2}\right]$.

5.3 Feature Points

In the former part we addressed the issue of triangulating an item captured in images at various recognized camera positions. Now we will discuss how the points are detected, what they depict, and how they are matched across images. Our main concern is to detect objects in an image that are affiliated with recurrently trackable 3-D entities. Considerable numbers researches has been conducted in the feature detection field, the majority of them concentrates on ”corners” or lines. Algorithms that detect repeatable features in an image are called feature detectors and there are plenty of them, every one them possesses its own pros and advantages and disadvantages. However, the comprehensive details of the way the feature detector operates is beyond the scope of

this thesis. Here, we will explain and dicuss solely the picked algorithm. For more information, we recommend the reader to check [32].

5.3.1 The fast feature detector

The FAST feature detector [33] is commonly employed in odometry since it is fast and acquire high feature output. Every value of the gray scale pixel is contrasted with the surrounding circle (Figure 5.5). When a specific combination of pixels is larger or smaller in value to the center, the center pixel is decided to be a corner point. The precise combinations that direct to corner points have been detected through an offline machine learning utilizing a huge dataset provided with ground truth information. The outcome is a number of open-source implementations which operate on several devices. Determining all the FAST features in an image utilizing a single-core processor takes almost 5ms.

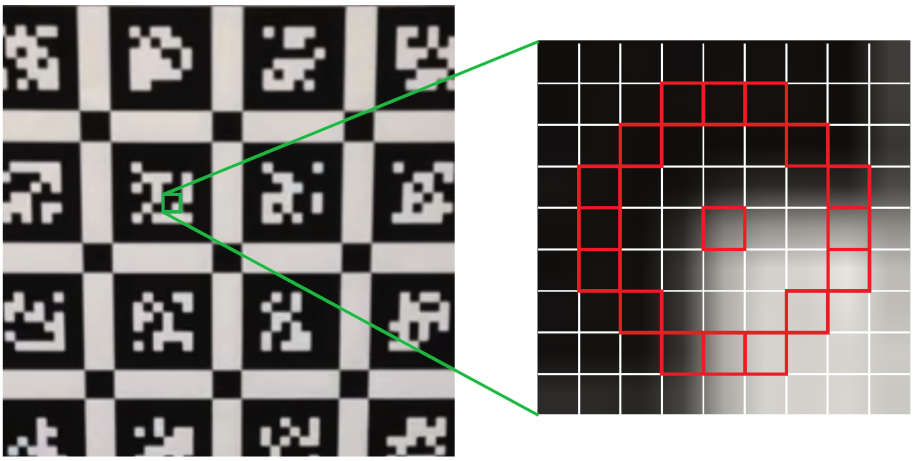


Figure 5.5 : The FAST corner detector compares the intensities of the image in a circle around a center pixel to find whether it is a corner or not.

The quantity of features returned relies heavily on a threshold value, the variance in intensity between the center and outer pixels needed to activate the detector. Utilizing a static value for this threshold could result in very big or very small number of features in an image, depending largely on motion blur and the type of scene. Dynamically changing the threshold offers better results, compelling the output to offer a specific number of features. To guarantee an even distribution of features across the image, we utilized Grid-FAST from the OpenCV library [34]. This algorithm splits the image

into a grid, every one of the grids possesses a distinct dynamic threshold to ensure that the entire number of features is inside a specific range.

The FAST feature detector possesses its part of drawbacks. Such as any feature detector that locates corner points, it depends on textured surfaces or cluttered scenes. Blank walls and floors usually do not acquire visible features, rendering visual odometry hard. Motion blur, induce the FAST detector to determine small number of features. This is by comparison with further computationally costly feature detectors which calculate image gradients at multiple scales. Nonetheless, the enhanced speed from the FAST detector compensate for this deficiency.

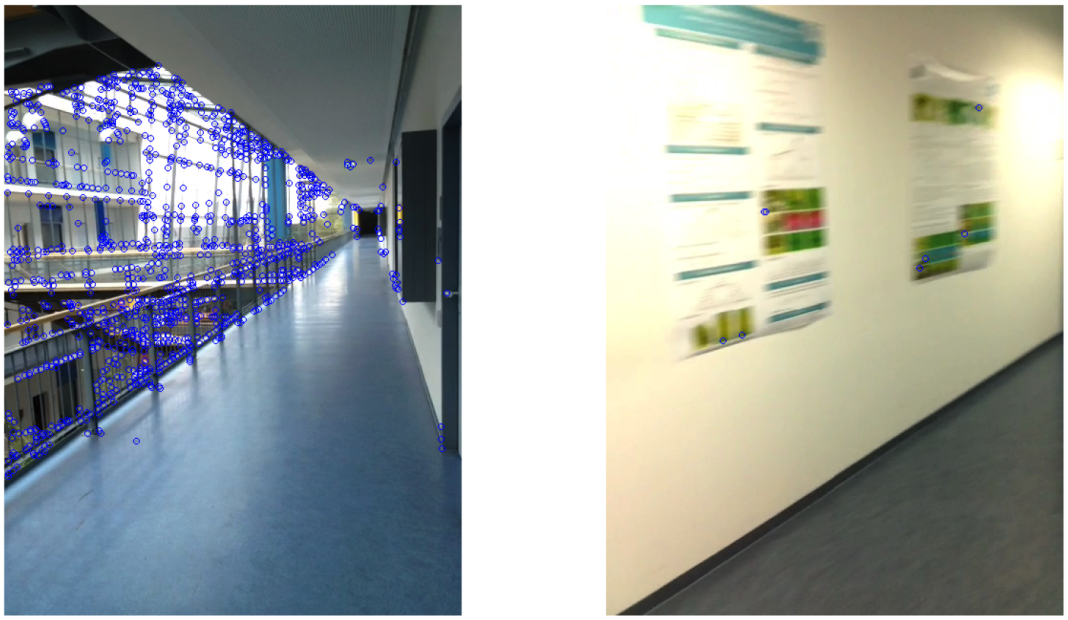


Figure 5.6 : The FAST feature detector is, for sure, rapid. However, it comes with drawbacks. Areas with no features such as the floor and ceiling in the left image, and motion blur like in the right image may result in considerable problems with odometry.

5.4 Feature Matching

When features are detected in consecutive images, they have to be matched with together in a fast and accurate manner. One method to find out whether two regions in two different images depict the identical physical location, is to match abstract characterizations of these regions. Several algorithms are utilized for extracting details of image patches to generate robust and replicable matches. SIFT [35] and SURF [36] are two common examples, with in-variance to slight rotations, lighting changes,

etc. However, both of them are too computationally costly for our use. They need computation of the image gradient, which is to a great extent able to be a parallel process, but it is on the other hand slow on motion devices. The enormous dimension of the descriptors (64 floating point values) hinder matching descriptors as well. Lately, binary descriptors like BRIEF [37] were developed. Those descriptors operate in a alike manner to the FAST corner detector, by matching image intensity values that surround the central feature point. The outcome of the comparison is a single binary digit denoting a bigger or smaller intensity value. All of the comparisons are gathered into a vector. The consequent binary vector is comparatively small and able to be rapidly compared with the Hamming distance [1]. The ORB [38] descriptor bring BRIEF to the next phase by appending orientation in-variance, which BRIEF miss by design. In experiments, ORB operates fine for real time use on a single-core CPU. However, a more rapid resolution do exist, rendering direct utilization of the supplied odometry.

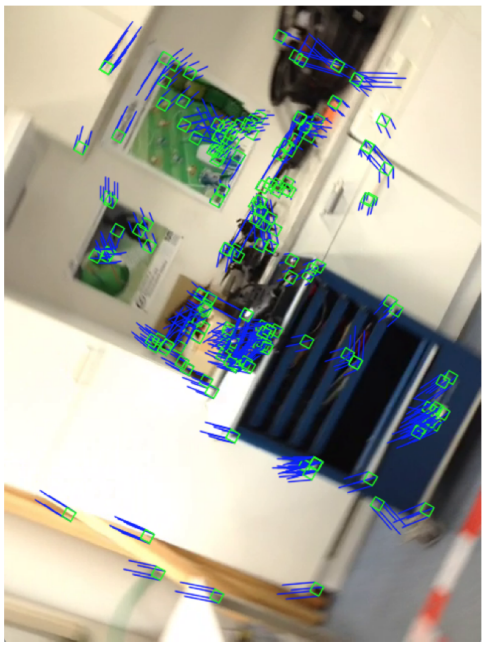


Figure 5.7 : An illustration of fast rotation while tracking patches. The green squares stand for the patches that are rotated to a canonical orientation (upright in line with gravity). The blue lines portray how the patches shifted since the former image. While number of outliers are present, most of tracked patches accurately transmit the type of rotation in progress.

The majority of feature descriptors are intended for general matching with no earlier evidence of the method in which the images have been taken. When performing

[1]The Hamming distance of two binary numbers is equal to the XOR operator pursued by summing up the number of 1's (the Hamming weight) in the outcome.

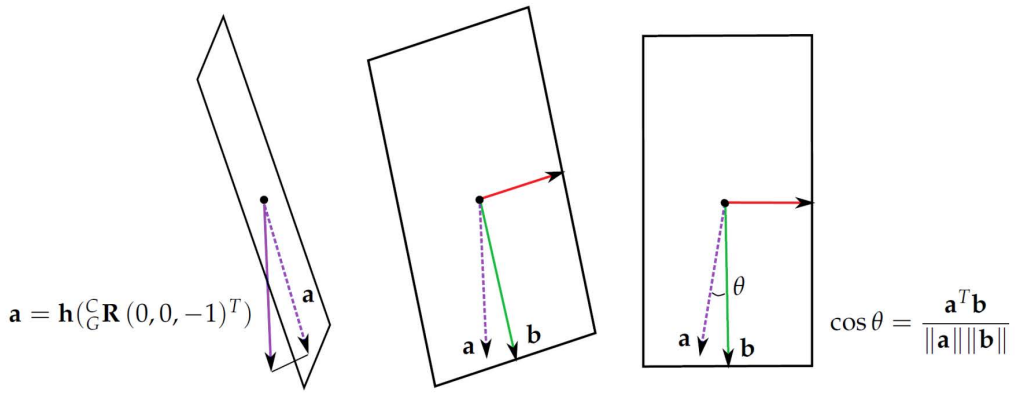


Figure 5.8 : The angle θ required to rotate patches is able to be indicated with the estimated camera rotation ${}^{B}_{G}\mathbf{R} = {}^{C}_{B}\mathbf{R}\,{}^{B}_{G}\mathbf{R}$. The global vector $(0,0,-1)^T$ is rotated into the frame of the camera and projected into the image plane. Then it is normalized and instantly matched with the vertical y-axis of the image through a dot product

odometry, we have some information about the orientation from which the image has been taken. We extract a square image patch surrounding every feature point. However, prior to extraction, A rotation to the square area to a canonical orientation is applied. while comparing with the next image patches, they can be rotated to the exact global orientation prior to comparison as well, enhancing match quality and rendering the system invariant to rotation surrounding the principal axis of the camera. The extraction of the patches is done through bi-linear interpolation. Afterwards, the mean value of the patch is deducted from the overall values and normalize the whole patch for a complete value of 1. This process of normalization significantly enhances the performance when matching. For patches comparison, we merely use the sum of squared differences (SSD) of the patch values and normalize it. This process is usually known as normalized cross correlation, or NCC. When we want to find out whether two patches match, we do comparison between the NCC result and the pixel distance to certain thresholds. If 3-D data is available, we can estimate a possible location for every correspondence by back-projecting the point into the image and searching in that surrounding area. As far as we are concerned, and since we will not estimate the depth of features, it is not feasible. On the other hand, we possess access to the system dynamics , particularly the translational and the rotational velocities. Theoretically they can be used to help narrow the search for point correspondences to decrease possibly fault matches. We perceive this as future work.

6. MULTI-STATE CONSTRAINT KALMAN FILTER.

6.1 Overview

The MSCKF is, similar to any inertial-aided EKF-SLAM, an Extended Kalman Filter based estimator which trails the pose of the object $\{B\}$ over certain period utilizing an IMU and a camera and. The two algorithms employ the IMU to propagate the state and co-variance matrix within the prediction step of the EKF, and the two algorithms trails feature points in images. The feature point is appended to the state vector in EKF-SLAM and afterwards, to co-variance matrix. Utilizing the inverse-depth representation, the feature then is appended after getting detected in that solely in 1 camera frame. In following frames in which it has been tracked, it is utilized to restrict the camera pose when updating its own inverse-depth estimate as well. A substantial drawback of this method is that the inverse-depth is provided by an initial estimate and variance, which is undetermined. Consequently, when features do not equal the picked previous distribution, the filter eventually turns into incoherent case. More serious still, outliers that have been detected have formerly had non-reversible effects on the filter, since they are employed in updates at each step. The MSCKF makes no attempt to append feature points to the state vector by no means. Alternatively, after every camera image, the present pose of the object is added to the state vector. After adding m, the first pose is omitted from the state vector in a First In First Out (FIFO) fashion. Simply put, the state vector keeps a sliding window of poses. utilizing this sliding window, a feature point is able to be triangulated after it disappear of sight or it appears m times, resulting in a highly accurate estimate of the 3-D feature pose. All of those features are afterwards employed to impose restrictions on the sliding window of poses in the update state of EKF. As mentioned in former chapters, several parameters are required to be predefined so that we can precisely utilize the camera and IMU data.

- The g-sensitivity matrix T_s, in addition to the IMU shape matrix T_g and T_a.
- The distortion and calibration values of the camera.

- The Camera to IMU time delay, t_d.
- The parameters of the IMU random walk bias, b_a and b_g.
- The offset position of the IMU-camera , ${}^{C}\mathbf{p}_{B}$
- The read time of the camera's rolling shutter , t_r.

6.2 State Representation

6.2.1 Rotation representation

To represent a rotation with minimum requirements, the 3-DoF is required. Euler angles, commonly referred to as roll, pitch, and yaw, are a popular method to describe orientation, however their drawback is that they experience singularities. Which happens if the angles are not uniquely defined, causing what is known as gimbal-lock. To hinder this type of ambiguity, the rotations of both body state and body camera are described using quaternions. A quaternion is a 4-D orientation representation that experiences no singularities. A generalized approach to perceive a quaternion is through the axis-angle representation, in which three parameters describe a 3-D axis whilst the fourth parameter represents a rotation around that axis, as shown in the equation below:

$$\bar{\boldsymbol{q}} = \begin{bmatrix} \vec{\boldsymbol{q}} \\ q_4 \end{bmatrix} = \begin{bmatrix} k_x \sin(\frac{\theta}{2}) \\ k_y \sin(\frac{\theta}{2}) \\ k_z \sin(\frac{\theta}{2}) \\ \cos(\frac{\theta}{2}) \end{bmatrix} \tag{6.1}$$

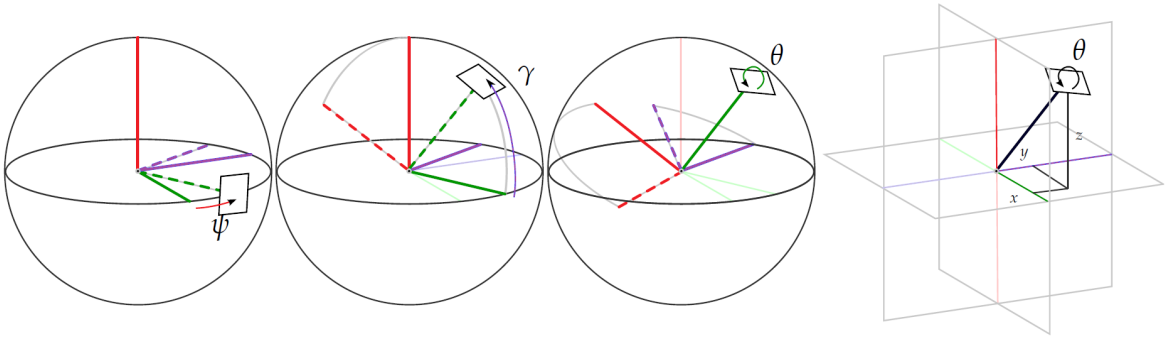


Figure 6.1 : An illustration of how Euler Angles and a quaternion are able to be employed to describe a specific rotation. It is possible to utilize Euler angles in different arranges in order to accomplish similar outcome.

For this thesis, quaternions will be perceived as unit quaternions except if the contrary is stated. The rotation matrix like ${}^{C}_{B}\mathbf{R}$ is in fact calculated from associated quaternions, written as ${}^{C}_{B}\mathbf{q}$. For more information about quaternions kindly check Appendix A.

6.2.2 Full state representation

Full state representation might be partitioned into 4 parts. The 1st is the evolving body state, the 2nd is the calibration matrices of the IMU, the 3rd is the rest of calibration values, and the 4th is the body poses sliding window.

$$\boldsymbol{x}_k = \left[\boldsymbol{x}_B^T \mid \boldsymbol{x}_{imu}^T \mid \boldsymbol{x}_c^T \mid \boldsymbol{\pi}_{B_{N-m}}^T \dots \boldsymbol{\pi}_{B_{N-1}}^T \right]^T$$

The varying state estimate of the body frame composes of the orientation, position, velocity, and the biases of the IMU . These characteristics grow in time and should be propagated within the EKF prediction step.

$$\boldsymbol{x}_k = \left[{}^{G}\mathbf{q}_B^T, {}^{G}\mathbf{p}_B^T, {}^{G}\mathbf{v}_B^T, \boldsymbol{b}_g^T, \boldsymbol{b}_a^T \right]^T$$

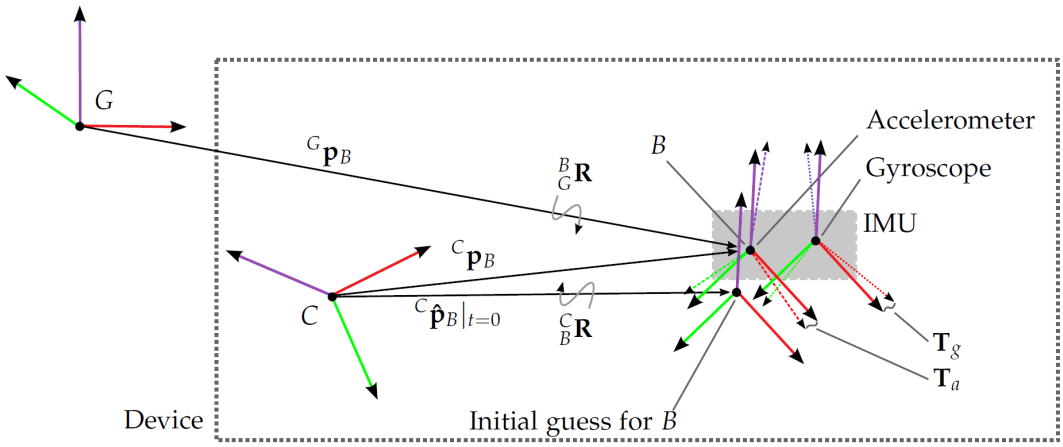


Figure 6.2 : This figure illustrates the three main frames and their association to the sensors. We specify the body frame B as the accelerometer location with predefined rotation according to the camera frame C. Minor errors in this rotation and scale is made up for through $\hat{T}_a$, while the remaining errors in the gyroscope rotation and scale is made up for by $\hat{T}_g$.

The IMU calibration constituent of the state vector is 27×1, composed of 3 vector matrices. As discussed in chapter 4, the matrices T_g and T_a proper scale and misalignment of the gyroscope and accelerometer, in each instance, and the matrix T_s fixes for g-sensitivity.

$$\boldsymbol{x}_{imu} = \left[\vec{\boldsymbol{T}}_g^T, \vec{\boldsymbol{T}}_s^T, \vec{\boldsymbol{T}}_a^T \right]^T$$

The T matrix is factorized as below:

$$\boldsymbol{T}_g = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \qquad \vec{\boldsymbol{T}}_g = [\, a,\, b,\, c,\, d,\, e,\, f,\, g,\, h,\, i \,]^T$$

The calibration variables are constants have been evaluated in the filter and possess certain initial uncertainty.

$$\boldsymbol{x}_c = \left[\, {}^C\mathbf{p}_B^T,\, \boldsymbol{x}_{cam}^T,\, t_d,\, t_r \right]^T$$

$$\boldsymbol{x}_{cam} = [\, f_x,\, f_y,\, o_x,\, o_y,\, k_1,\, k_2,\, k_3,\, t_1,\, t_2 \,]^T$$

Eventually, every pose within the sliding window includes the quaternion, position and velocity. The velocity is utilized to estimate the temporal calibration parameters. With a notice that there is no requirement to append the angular velocity to the sliding window since we acquire access to the IMU measurements precisely.

$$\boldsymbol{pi}_{B_i} = \left[\, {}^G\mathbf{q}_{B_i}^T,\, {}^G\mathbf{p}_{B_i}^T,\, {}^G\mathbf{v}_{B_i}^T \right]^T$$

The full size of the state vector is $(16+27+14+10m)\times 1 = (57+10m)\times 1$, where m is the poses number within the sliding window.

6.3 Propagation

State and co-variance propagation happens when recent IMU measurements are available. The IMU offers the recent rotational velocity and linear acceleration. In the subsequent parts information on how the movement is explained with the IMU data in both continuous and discrete manner will be discussed, in addition to how to propagate the discrete error. In the sense of this chapter, we suppose that the gyroscope and accelerometer data are available simultaneously within a fixed time period, Δt. In actuality they could possess distinct periods and thus interpolation is needed.

6.3.1 Continuous dynamics

Solely the parameters in the evolving body partition of the state vary in time and should be modelled dynamically. The remaining are maintained persistent throughout the

prediction step of the EKF.

$$ {}^{B}_{G}\dot{\mathbf{q}}(t) = \frac{1}{2}\Omega({}^{B}\boldsymbol{\omega}(t)){}^{B}_{G}\mathbf{q}(t) \tag{6.2} $$

$$ \text{with } \Omega(\omega) = \begin{bmatrix} -\omega\times & \omega \\ -\omega^{T} & 0 \end{bmatrix} \tag{6.3} $$

$$ {}^{G}\dot{\mathbf{p}}(t) = {}^{G}\mathbf{v}(t) \tag{6.4} $$

$$ {}^{G}\dot{\mathbf{v}}(t) = {}^{G}\mathbf{a}(t) \tag{6.5} $$

$$ \dot{\boldsymbol{b}}_{g}(t) = \mathbf{n}_{w_g}(t) \tag{6.6} $$

$$ \dot{\boldsymbol{b}}_{a}(t) = \mathbf{n}_{w_a}(t) \tag{6.7} $$

6.3.2 Gyroscope measurements

As addressed in Chapter 4, the IMU data are corrupted by several of causes. While propagating, we utilize the present estimates of the calibration variables to redress these readings errors. The gyroscope offers the rotational velocity in the body frame after compensating for scale factors, misalignment,bias, and g-sensitivity.

$$ {}^{B}\hat{\boldsymbol{\omega}}(t) = \hat{\mathbf{T}}_{g}^{-1}\left({}^{B}\boldsymbol{\omega}_{m}(t) - \hat{\mathbf{T}}_{s}{}^{B}\hat{\mathbf{a}}(t) - \hat{\mathbf{b}}_{g}(t)\right) \tag{6.8} $$

The measured rotational velocity is represented by ${}^{B}\boldsymbol{\omega}_{m}$, and the estimate by ${}^{B}\hat{\boldsymbol{\omega}}$. The rotational velocity estimated then utilized to propagate the present orientation estimate, however we are unable to merge it plainly since the orientation is represented as a quaternion parameters. To calculate the subsequent orientation, we need the minor quaternion ${}^{B_{l+1}}_{B_l}\hat{\mathbf{q}}$ that denotes the rotation from B_l to B_{l+1}.

$$ {}^{B_{l+1}}_{G}\hat{\mathbf{q}} = {}^{B_{l+1}}_{B_l}\hat{\mathbf{q}} \otimes {}^{B_{l+1}}_{G}\hat{\mathbf{q}} \tag{6.9} $$

To calculate this small quaternion, an integrate of the differential equation should be carried out

$$ {}^{B_t}_{B_l}\dot{\hat{\mathbf{q}}} = \frac{1}{2}\Omega\left({}^{B}\hat{\boldsymbol{\omega}}(t)\right){}^{B_t}_{B_l}\hat{\mathbf{q}} \qquad\qquad t \in [t_l, t_{l+1}] \tag{6.10} $$

With discrete readings and time variance Δ, we integrate this utilizing the 4th order Runge-Kutta approach. We utilize the abbreviation $\hat{\boldsymbol{\omega}}_l = {}^{B}\hat{\boldsymbol{\omega}}(t_l)$

$$ \begin{aligned} {}^{B_{l+1}}_{B_l}\hat{\mathbf{q}} &= text\bar{b}fq_0 + \frac{\Delta}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ \bar{\mathbf{q}}_0 &= [0,0,0,1]^{T} \end{aligned} \tag{6.11} $$

The terms from K_1 to K_4 are specified as below, with $\Omega(\hat{\boldsymbol{\omega}})$ from equation (6.3).

$$\begin{aligned} K_1 &= \frac{1}{2}\Omega(\hat{\boldsymbol{\omega}}_l)\bar{\mathbf{q}}_0 \\ K_2 &= \frac{1}{2}\Omega(\frac{\hat{\boldsymbol{\omega}}_l + \hat{\boldsymbol{\omega}}_{l+1}}{2})(\bar{\mathbf{q}}_0 + \frac{\Delta}{2}K_1) \\ K_3 &= \frac{1}{2}\Omega(\frac{\hat{\boldsymbol{\omega}}_l + \hat{\boldsymbol{\omega}}_{l+1}}{2})(\bar{\mathbf{q}}_0 + \frac{\Delta}{2}K_2) \\ K_4 &= \frac{1}{2}\Omega(\hat{\boldsymbol{\omega}}_{l+1})(\bar{\mathbf{q}}_0 + \Delta t K_3) \end{aligned} \tag{6.12}$$

The drawback of this numerical integration is that the derived quaternion could not be a unit quaternion. To solve that, we should normalize it to unit-sphere.

$$^{B_{l+1}}_{B_l}\hat{\bar{\mathbf{q}}} \leftarrow {}^{B_{l+1}}_{B_l}\hat{\mathbf{q}} \;/\; ||{}^{B_{l+1}}_{B_l}\hat{\mathbf{q}}|| \tag{6.13}$$

6.3.3 Accelerometer measurements

Similar to the gyroscope case, the accelerometer readings need to be redressed with the present IMU calibration value estimates to obtain the regional suitable acceleration. This is able to afterwards be rotated to the global frame in which gravity may be omitted. Note that gravity is $(0,0,-1)^T$ and the accelerometer in rest position measures an acceleration of one in the positive global z direction. For this reason we should append gravity to omit it from the global acceleration.

$$\begin{aligned} {}^{B}\hat{\mathbf{a}}(t) &= \hat{\mathbf{T}}_a^{-1}\left(\mathbf{a}_m(t) - \hat{\mathbf{b}}_a\right) \\ {}^{G}\hat{\mathbf{a}}(t) &= {}^{G}_{B}\hat{\mathbf{R}}(t)\,{}^{B}\hat{\mathbf{a}}(t) + {}^{G}\mathbf{g} \end{aligned} \tag{6.14}$$

Integrating the acceleration provides the propagated velocity in the global frame.

$$\begin{aligned} {}^{G}\hat{\mathbf{v}}_{l+1} &= {}^{G}\hat{\mathbf{v}}_l + \int_{t_l}^{t_{l+1}} {}^{G}\hat{\mathbf{a}}(\tau)d\tau \\ &= {}^{G}\hat{\mathbf{v}}_l + \int_{t_l}^{t_{l+1}} \left({}^{G}_{B_\tau}\hat{\mathbf{R}}\,{}^{B}\hat{\mathbf{a}}(\tau) + {}^{G}\hat{\mathbf{g}}\right)d\tau \\ &= {}^{G}\hat{\mathbf{v}}_l + \int_{t_l}^{t_{l+1}} {}^{G}_{B_\tau}\hat{\mathbf{R}}\,{}^{B}\hat{\mathbf{a}}(\tau)d\tau + {}^{G}\mathbf{g}\Delta t \\ &= {}^{G}\hat{\mathbf{v}}_l + {}^{G}_{B_l}\hat{\mathbf{R}}\int_{t_l}^{t_{l+1}} {}^{B_l}_{B_\tau}\hat{\mathbf{R}}\,{}^{B}\hat{\mathbf{a}}(\tau)d\tau + {}^{G}\mathbf{g}\Delta t \\ &= {}^{G}\hat{\mathbf{v}}_l + {}^{G}_{B_l}\hat{\mathbf{R}}\hat{\mathbf{s}}_l + {}^{G}\mathbf{g}\Delta t \end{aligned} \tag{6.15}$$

Integrating the velocity offers the propagated position

$$\begin{aligned} {}^{G}\hat{\mathbf{p}}_{l+1} &= {}^{G}\hat{\mathbf{p}}_l + \int_{t_l}^{t_{l+1}} {}^{G}\hat{\mathbf{v}}(\tau)d\tau \\ &= {}^{G}\hat{\mathbf{p}}_l + {}^{G}\hat{\mathbf{v}}_l + {}^{G}_{B_l}\hat{\mathbf{R}}\hat{\mathbf{y}}_l + \frac{1}{2}{}^{G}\mathbf{g}\Delta t^2 \end{aligned} \tag{6.16}$$

where $\Delta t = t_{l+1} - t_l$, ${}^{G}_{B_l}\hat{\mathbf{R}} = {}^{G}_{B}\hat{\mathbf{R}}(t_l)$, and the components are as below:

$$\begin{aligned}\hat{\mathbf{s}} &= \int_{t_l}^{t_{l+1}} {}^{B_l}_{B_\tau}\hat{\mathbf{R}}\,{}^{B}\hat{\mathbf{a}}(\tau)d\tau \\ \hat{\mathbf{y}} &= \int_{t_l}^{t_{l+1}} \int_{t_l}^{s} {}^{B_l}_{B_\tau}\hat{\mathbf{R}}\,{}^{B}\hat{\mathbf{a}}(\tau)d\tau ds\end{aligned} \tag{6.17}$$

6.4 Augmentation

If an image is taken, the present body quaternion, position and velocity are appended to the state vector and the co-variance is then augmented consequently.

$$\begin{aligned}\mathbf{\Sigma}_{k|k-1} \leftarrow \begin{bmatrix} \Sigma_{k|k-1} & \mathbf{\Sigma}_{k|k-1}\boldsymbol{J}_\pi^T & \\ \boldsymbol{J}_\pi\mathbf{\Sigma}_{k|k-1} & \boldsymbol{J}_\pi\mathbf{\Sigma}_{k|k-1} & \boldsymbol{J}_\pi^T \end{bmatrix}^T \\ \text{with } \boldsymbol{J}_\pi = \begin{bmatrix} I_3 & 0_3 & 0_3 & 0_3 & \dots \\ 0_3 & I_3 & 0_3 & 0_3 & \dots \\ 0_3 & 0_3 & I_3 & 0_3 & \dots \end{bmatrix}\end{aligned} \tag{6.18}$$

Where $\boldsymbol{J}_p i$ is the derivative of the camera pose W.R.T the state vector.

6.5 Update Step

An EKF update needs a measurement, z, that relies on and able to be estimated from the present state. If the comparison is done, the variance between the measurement and the estimate result in the residual vector.

$$\begin{aligned} r &= r - \hat{z} \\ &\simeq H\tilde{x} + n \end{aligned} \tag{6.19}$$

Where H represents the Jacobian of the measurement W.R.T the state vector, $\tilde{x}$ represents the state error vector and n is the noise of the measurement.

6.5.1 Calculating the residuals

For each image taken, its timestamp, t will associated. The IMU measurements are the utilized to propagate the state vector up until time $t + \hat{t}_d$ and the body pose is added to the state vector as mentioned earlier. The features in the image are identified and compared with features in the former frame. If the ith feature is gone, then it is triangulated with the approached mentioned in section 5.2 to calculate ${}^{G}\mathbf{p}_{f_i}$. This triangulated point gives the pixel measurement z_i at time-step j in which it appeared.

$$\mathbf{z}_{i,j} = \mathbf{h}({}^{C_i}\mathbf{p}_{f_i}) + \mathbf{n}_{i,j} \tag{6.20}$$

$$ {}^{C_i}\mathbf{p}_{f_i} = {}^{C}_{B}\mathbf{R}\,{}^{B_i}_{G}\mathbf{R}(t_n)\left({}^{C_i}\mathbf{p}_{f_i} - {}^{G}\mathbf{p}_{B_j}(t_n)\right) + {}^{C}\mathbf{p}_{B} \tag{6.21} $$

The function h is the projection function of the camera as it has been explained earlier, and $n_{i,j}$ is the measurement noise vector of camera. The rotation and position of the body relies on the time t_n, that represents the moment that the column [1] which includes the pixel reading has been taken. The central column of the image is taken at $t + t_d$ whereas row k is taken at $t + t_d + \frac{kt_r}{N}$, $k \in [-\frac{N}{2}, \frac{N}{2}]$. To calculate the estimate $\hat{\mathbf{z}}_{i,j}$, the buffered IMU readings are utilized to carry on a forwards integration (or backwards for $kt_r < 0$) employing the equations from the EKF prediction step in the former part.

$$ \mathbf{z}_{i,j} = \mathbf{h}\left({}^{C}_{B}\mathbf{R} = {}^{B_j}_{G}\hat{\mathbf{R}}(t + \hat{t}_d + \frac{k\hat{t}_r}{N})\left({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{B_j}(t + \hat{t}_d + \frac{k\hat{t}_r}{N})\right) + {}^{C}\hat{\mathbf{p}}_{B}\right) \tag{6.22} $$

6.5.2 Feature error marginalization

The feature is not part of the state vector, however it is calculated from and thus related to the state. To solve this isse, we persue the method mentioned in [25] and try eradicate the feature

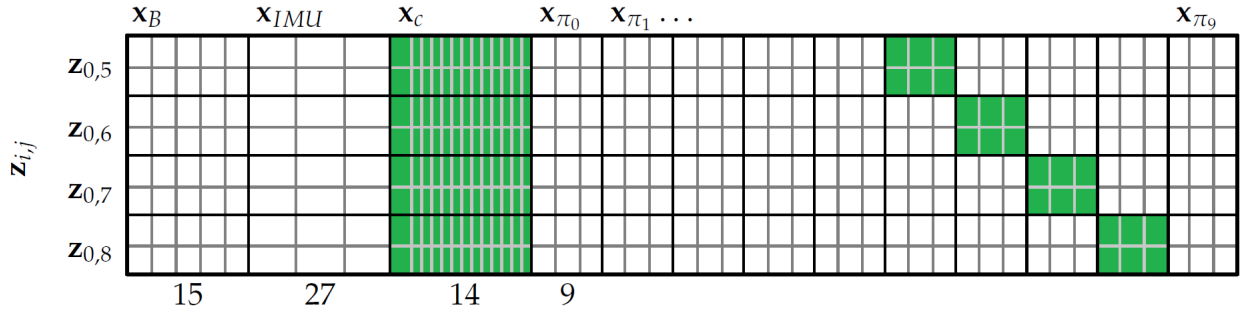


Figure 6.3 : The piled up Jacobian $\mathbf{H}_x$ for feature i acquires nonzero entries (shown in green) at the poses in which it was detected and at the camera calibration index.

from the residual. Since $\mathbf{H}_{fi}$ is a $2n \times 3$ matrix a full column rank, the null-space of it possesses dimension $2n-3$. If A_i is a unitary matrix whose columns shape the basis of the left null-space of $\mathbf{H}_{fi}$, then $\mathbf{A}_i^T\mathbf{H}_{fi} = 0_{2n-3\times 1}$ and $\mathbf{A}_i\mathbf{A}^T = I_{2n-3}$. Now we marginalize out the feature error.

$$ \begin{aligned} \mathbf{A}_i^T\mathbf{r} &\simeq \mathbf{A}_i^T\mathbf{H}_{xi}\tilde{\mathbf{x}} + \mathbf{A}_i^T\mathbf{H}_{fi} + \mathbf{A}_i^T\mathbf{n}_i \\ \mathbf{A}_i^T\mathbf{r}_i &\simeq \mathbf{A}_i^T\mathbf{H}_{xi}\tilde{\mathbf{x}} + \mathbf{A}_i^T\mathbf{n}_i \\ \mathbf{r}_i^o &\simeq \mathbf{H}_i^o\tilde{\mathbf{x}} + \mathbf{n}_i^o \end{aligned} \tag{6.23} $$

[1] Certain rolling shutter cameras use row-wise and the others use column-wise. For this section we plainly presumed it to be the column notation

6.5.3 Outlier detection

Prior to employing $\mathbf{r}_i^o$ and $\mathbf{H}_i^o$ in an EKF update, we implement a statistical examination to separate inliers from outliers. Outliers happen if the feature tracker stumbles and matches features improperly or if the tracked feature is non-static, like on a car or pedestrian. The whole residual vector from a feature is indicated an inlier or an outlier and all outliers are denied.

The Chi-square examination is a widely utilized testing to check if a hypothesis accommodates an observation. The examination result in a single value, γ, by adding up the square variations between an observation and a hypothesis, the divided by the variance.

$$\gamma = \sum_{i=1}^{k} \left(\frac{X_i - \mu_i}{\delta_i} \right)^2 \tag{6.24}$$

As far as we are concerned, the residual $\mathbf{r}_i^o$ is formerly the difference between the observation and hypothesis, and the residual co-variance represented through $\mathbf{H}_i^o \, \Sigma \, \mathbf{H}_i^{oT}$.

$$\gamma_i = \mathbf{r}_i^o \left(\mathbf{H}_i^o \, \Sigma \, \mathbf{H}_i^{oT} \right)^{-1} \mathbf{r}_i^{oT} \tag{6.25}$$

When γ is big, it is obvious that few or all of the observations are away from the hypothesis. We do comparison between γ and X liket the Chi-squared cumulative density function for X and k degrees of freedom is 0.95. That is to say, the variable X which gives 0.95 from the Chi-square CDF is the upper bound for statistical importance, so if γ is lower than X, the ith feature is an inlier. The Chi-squared (X^2) CDF is as below. Note that k is the degrees of freedom, for $\mathbf{r}_i^o$ it is $2n-3$.

$$\begin{aligned} \mathrm{cfd}_{x^2}(X,k) &= \frac{\gamma(\frac{k}{2}, \frac{x}{2})}{\Gamma(\frac{k}{2}, \frac{k}{2})} \\ \Gamma(s,x) &= \int_x^{\infty} t^{s-1} e^{-t} dt \\ \gamma(s,x) &= \int_0^{x} t^{s-1} e^{-t} dt \end{aligned} \tag{6.26}$$

In reality, the Chi-square cdf is not assessed, instead a table of potential estimations for X and k that yield 0.95 is utilized for efficient matches.

6.5.4 EKF update

After the outliers are dropped, the rest of residual vectors from all L evaluated inlier features are piled up jointly.

$$\begin{bmatrix} \mathbf{r}_1^o \\ \vdots \\ \mathbf{r}_L^o \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1^o \\ \vdots \\ \mathbf{H}_L^o \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{n}_1^o \\ \vdots \\ \mathbf{n}_L^o \end{bmatrix} \tag{6.27}$$

$$\mathbf{r} = \mathbf{H}\tilde{\mathbf{x}} + \mathbf{n} \tag{6.28}$$

The size of r is at this time $d \times 1$ with $d = \sum_{j=1}^{L}(2n_j - 3)$. Since the noise vectors for every feature are non correlated, the co-variance matrix of n is $\mathbf{R}_n = \delta_{im}^2 \mathbf{I}_d$. The EKF Update progresses by computing the Kalman Gain.

$$\mathbf{K} = \mathbf{\Sigma}\,\mathbf{H}^T\,(\mathbf{H}\mathbf{\Sigma}\,\mathbf{H}^T + \mathbf{R}_n)^{-1} \tag{6.29}$$

For big values of d, the difficulty of the earlier equation increases extremely fast. For instance, if 11 features each perceived in 11 images, then $d = 209$. To decrease the complication, we investigate the matrix $\mathbf{H}$. The QR decomposition of $\mathbf{H}$ displays huge load of the information is noise and could be neglected.

$$\mathbf{H} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} T_h \\ 0 \end{bmatrix} \tag{6.30}$$

The matrices Q_1 and Q_2 represent unit matrix and possess columns that compose the bases for the range and null-space of H, individually. we will reformulate equation 6.28 as below:

$$\begin{aligned} \mathbf{r} &= \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T}_H \\ 0 \end{bmatrix} \tilde{\mathbf{x}} + \mathbf{n} \\ \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix}^T \mathbf{r} &= \begin{bmatrix} \mathbf{T}_H \\ 0 \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix}^T \mathbf{n} \\ \begin{bmatrix} \mathbf{Q}_1^T \mathbf{r} \\ \mathbf{Q}_2^T \mathbf{r} \end{bmatrix} &= \begin{bmatrix} \mathbf{T}_H \\ 0 \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{Q}_1^T \mathbf{n} \\ \mathbf{Q}_2^T \mathbf{n} \end{bmatrix} \end{aligned} \tag{6.31}$$

it has become evident that $\mathbf{Q}_2^T \mathbf{r}$ is solely noise and is able be neglected yielding a smaller dimensional residual and Jacobian. The resulting noise vector, $\mathbf{Q}_1^T \mathbf{n}$, possesses co-variance $\mathbf{R}_q = \mathbf{Q}_1^T \mathbf{R}_n \mathbf{Q}_1 = \delta_{im} \mathbf{I}_r$ (since $\mathbf{Q}_1$ is a unit matrix) where r is the rank of columns in $\mathbf{Q}_1$ and the resulting dimension of residual.

$$\begin{aligned} \mathbf{Q}_1^T \mathbf{r} &= \mathbf{T}_H \tilde{\mathbf{x}} + \mathbf{Q}_1^T \mathbf{n} \\ \mathbf{r}_q &= \mathbf{T}_H \tilde{\mathbf{x}} + \mathbf{n}_q \end{aligned} \tag{6.32}$$

Eventually, the EKF update could happen with the new residual and Jacobian matrix.

$$\begin{aligned} \mathbf{K} &= \boldsymbol{\Sigma}_{k+1|k}\mathbf{T}_H^T(\mathbf{T}_H\boldsymbol{\Sigma}_{k+1|k}\mathbf{T}_H^T + \mathbf{R}_q)^{-1} \\ \boldsymbol{\Sigma}_{k+1|k+1} &= (\mathbf{I}_\beta - \mathbf{K}\mathbf{T}_H)\boldsymbol{\Sigma}_{k+1|k}(\mathbf{I}_\beta - \mathbf{K}\mathbf{T}_H)^T + \mathbf{K}\mathbf{R}_q\mathbf{K}^T \\ \Delta\mathbf{x} &= \mathbf{K}\mathbf{r}_q \end{aligned} \tag{6.33}$$

$$\Delta\mathbf{x} = \left[\underbrace{\delta\boldsymbol{\theta}_B^T, \Delta\mathbf{p}_B^T, \Delta\mathbf{v}_B^T, \Delta\mathbf{b}_g^T, \Delta\mathbf{b}_a^T}_{\text{Body Pose Update}}, \ldots \underbrace{\delta\boldsymbol{\theta}_{B_i}^T, \Delta\mathbf{p}_{B_j}^T, \Delta\mathbf{v}_{B_j}^T}_{\text{Pose j Update}}, \ldots \right] \tag{6.34}$$

The identity matrix $\mathbf{I}_\beta$ possesses the size of the co-variance matrix: $9m+56$ for m poses in the sliding window. The vector $\Delta\mathbf{x}$ is then utilized to update the present state estimate

$$\hat{\mathbf{x}}_{k+1|k+1} = \begin{bmatrix} {}^B_G\hat{\mathbf{q}} \otimes \delta\mathbf{q}_B \\ {}^G\hat{\mathbf{p}}_B + \Delta\mathbf{p}_B \\ {}^G\hat{\mathbf{v}}_B + \Delta\mathbf{v}_B \\ \hat{\mathbf{b}}_g + \Delta\mathbf{b}_g \\ \hat{\mathbf{b}}_a + \Delta\mathbf{b}_a \\ \vdots \\ {}^{B_i}_G\hat{\mathbf{q}} \otimes \delta\mathbf{q}_{B_i} \\ {}^G\hat{\mathbf{p}}_{B_i} + \Delta\mathbf{p}_{B_i} \\ {}^G\hat{\mathbf{v}}_{B_i} + \Delta\mathbf{p}_{B_i} \end{bmatrix} \quad \text{with } \delta\mathbf{q} = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix} \tag{6.35}$$

6.6 Post EKF Update

Afterwards, the update of the EKF has been accomplished, when poses in the state vector are numerous, the first pose is dropped and the associated rows and columns of the co-variance matrix are omitted. Other old poses which possess no tracked features are able to be dropped at this stage as well.

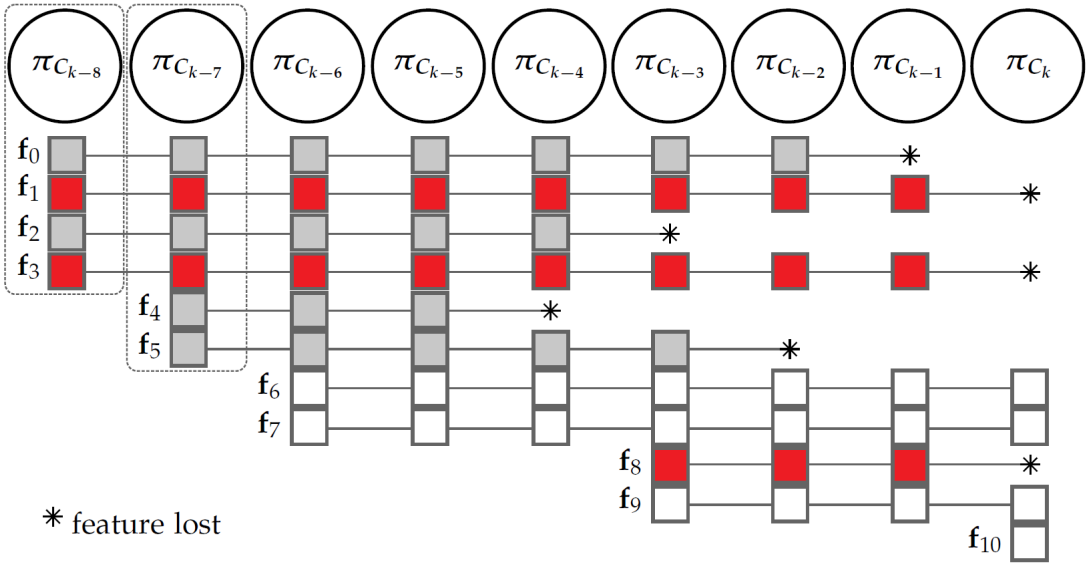


Figure 6.4 : The sliding window of the poses of the camera poses at moment k and the associated features. For features which are gone in the present frame (shown in red), an update is executed employing their triangulated positions as restrictions. After updating, the first camera pose with no tracked features left is neglected (in this example, the oldest two camera poses). The grey boxes are features that were already utilized for the update process.

7. IMPLEMENTATION

7.1 Data sets

Collecting and synchronizing different sensor data is itself a huge topic and a tiresome of task. Moreover, the process of performing the calibration is another challenge. This is why in this thesis I used public datasets i.e. EuRoC MAV ETH dataset, which contains the necessary data which I used for bench-marking my results.

These datasets have been recorded in the context of the European Robotics Challenge (EuRoC), to evaluate the visual-inertial SLAM and 3-D reconstruction capabilities on (Micro Aerial Vehicle) MAVs. It has been tested by more than 20 teams and has been utilized for evaluation in several publications [39].
The content of the datasets ranges from slow flights in a small, cluttered workspace to dynamic flights in a large industrial machine hall. This permits testing and tuning under several circumstances.

Figure 7.0 : Aptina MT9V034 global shutter, WVGA monochrome, 2 × 20 FPS

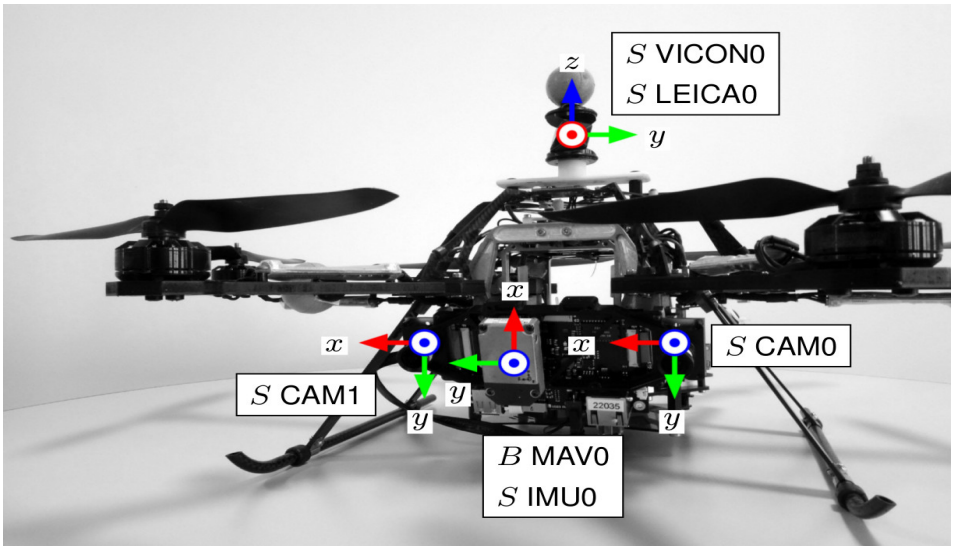


Figure 7.1 : The picture shows the AscTec "Firefly" hexacopter that was utilized to record the data. Via a visual-inertial sensor unit stereo images and measurements from an IMU has been collected. A Leica station has been utilized to document the ground-truth for position, meanwhile a Vicon system was utilized to document the 6-D pose ground-truth.

The available data in the datasets are:

- Visual-Inertial Sensor Unit
  - **Stereo Images** (Aptina MT9V034 global shutter, WVGA monochrome, 2 $\times$ 20 FPS)
  - **MEMS IMU** (ADIS16448, angular rate and acceleration, 200 Hz)
  - Shutter-centric temporal alignment
- Ground-Truth
  - **Vicon** motion capture system (6D pose)
  - **Leica** MS50 laser tracker (3D position)
  - Leica MS50 3D structure scan
- Calibration
  - Camera intrinsics
  - Camera-IMU extrinsics
  - Spatio-temporally aligned ground-truth

Two types of datasets are presented:

The first batch of datasets has been documented in a large machine hall (Figure 7.2) and intends to test visual-inertial motion estimation algorithms or SLAM frameworks. For ground truth, a 3-D position was equipped with a laser tracker. A second batch of datasets was recorded in the vicon room of ETH Zürich with an approximate size of $8m \times 8.4m \times 4m$ provided with a motion recording system with different obstacle configurations. Each type consists of three datasets with increasing complexity. The entire datasets have been documented through an AscTec Firefly MAV, provided with a VI-Sensor (Figure 7.1) and a short summary of the trajectories is given in Table 7.1.

Figure 7.2 : The machine hall in which the datasets has been recorded.

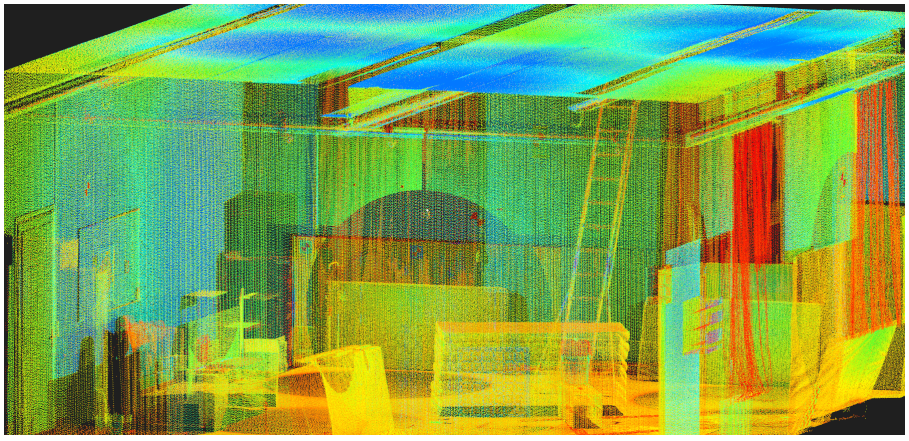


Figure 7.3 : 3D scan of ground-truth.

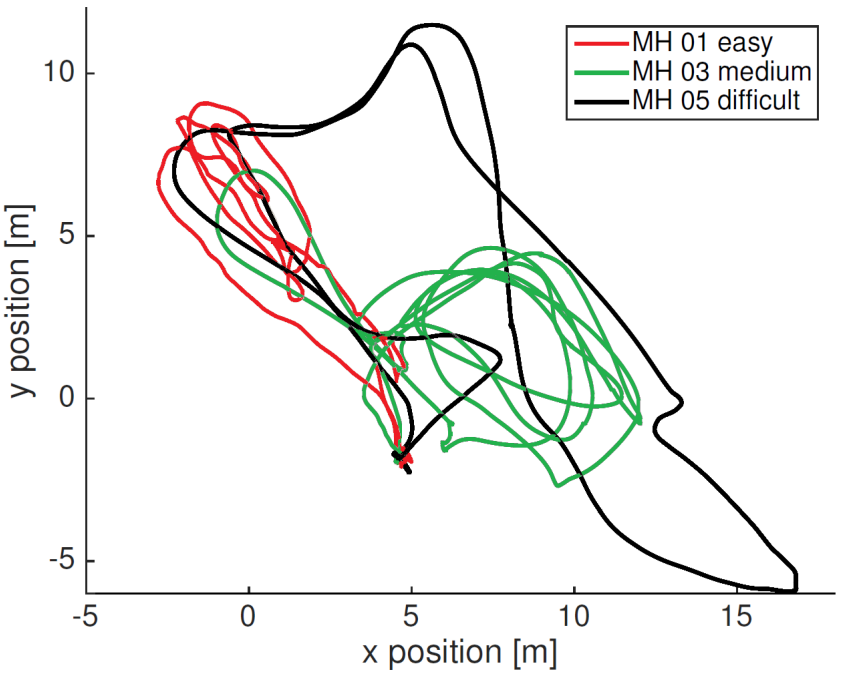


Figure 7.4 : An illustration of 3 flight trajectories, documented in the machine hall. The datasets complexity varies W.R.R the trajectory length, flight dynamics, and light settings.

7.1.1 Sensor setup

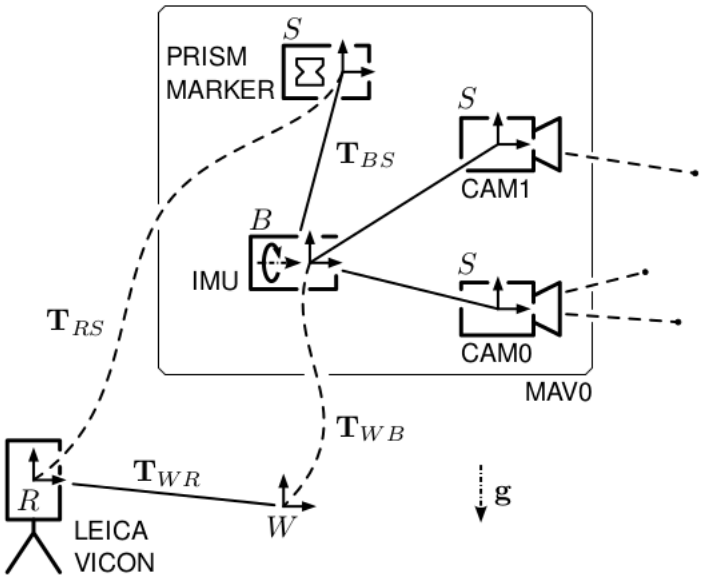


Figure 7.5 : The sensor-system that has been utilized to collect the datasets. Every sensor gives data with respect to its reference frame S. A calibration for the whole extrinsic values connecting the sensors with the body frame B and intrinsic values are supplied along the datasets.

Table 7.1 : Dataset characteristics

Name	Length / Duration	Avg. Vel. / Angular Vel.	Note
MH_01_easy	80.6m	0.44ms^{-1}	good texture,
	182s	0.22rad s^{-1}	bright scene
MH_02_easy	73.5m	0.49ms^{-1}	good texture,
	150s	0.21 rad s^{-1}	bright scene
MH_03_medium	130.9m	0.99ms^{-1}	fast motion,
	132s	0.29 rad s^{-1}	bright scene
MH_04_difficult	91.7m	0.93ms^{-1}	fast motion,
	99 s	0.24 rad s^{-1}	dark scene
MH_05_difficult	97.6m	0.88ms^{-1}	fast motion,
	111 s	0.21 rad s^{-1}	dark scene
V1_01_easy	58.6m	0.41ms^{-1}	slow motion,
	144 s	0.28 rad s^{-1}	bright scene
V1_02_medium	75.9m	0.91 ms^{-1}	fast motion,
	83.5 s	056 rad s^{-1}	bright scene
V1_03_difficult	79.0m	0.75ms^{-1}	fast motion,
	105 s	0.62 rad s^{-1}	motion blur
V2_04_easy	36.5m	0.33ms^{-1}	slow motion,
	112 s	0.28 rad s^{-1}	bright scene
V2_05_medium	83.2m	0.72ms^{-1}	fast motion,
	115 s	0.59 rad s^{-1}	bright scene
V2_05_difficult	86.1m	0.75ms^{-1}	fast motion,
	115 s	0.66 rad s^{-1}	motion blur

Table 7.2 : Sensors and ground truth instruments.

Sensor	Type	Rate	Characteristics
Cameras	MT9V034	2×20Hz	WVGA, global shutter
IMU	ADIS16448	200 Hz	MEMS, intr. calibrated
Position	Leica MS50	20 Hz	Accuracy $\approx$ 1mm
Pose	Vicon	100 Hz	6D
Structure	Leica MS50	-	Accuracy: $\approx$ 1mm

The visual and inertial data has been logged and time-stamped on-board the MAV, meanwhile ground truth has been logged on the base station. A maximum likelihood estimator has been utilized to align the data temporarily, and to calibrate the position of the ground truth body coordinate frames with W.R.T sensor unit. Both raw and spatio-temporally aligned data is supplied with the datasets (Figure 7.4). In addition, table 7.2 shows an outline of the sensors and reference systems for the measurements provided in the datasets.

7.1.2 Data Format

Each dataset includes one or more "sensor-systems" (one MAV in our case), and each sensor-system may have several "sensors". Each sensor comes with a **`sensor.yaml`** file which determines its calibration parameters and a **`data.csv`** file which either includes the sensor readings or points to **`data`** files in the optional data folder:

```
<sensor system id>
   <sensor id>
        sensor.yaml
        data.csv
        data
```

A ground truth source, raw or post processed, is handled as every other sensor. For example:

```
mav_0
   imu_0
        sensor.yaml
        data.csv
   cam_0
        sensor.yaml
        data.csv
        data
             1403636579250000000.png
             1403636579300000000.png
             ...
   vico_0
        sensor.yaml
        data.csv
```

Camera

The `data.csv` file of a camera contains time stamps, followed by the file name of the associated camera image. All the images are stored in the data folder in a loss-less compression format (PNG). As for every other sensor, the yaml file for the camera's includes the transform between the sensor and the sensor-system, commonly referred to as the extrinsics.

IMU

The `data.csv` file of a `imu` sensor includes the following parameters (ordered): time-stamp, ω_S [rad s^{-1}], a_S [m s^{-2}]. ω_S indicates the angular rate readings (i.e. the gyroscope's x, y, and z measurements, ordered as mentioned), and a_S indicates the specific force readings (i.e. the accelerometer measurements), both expressed in the frame of reference of the sensor. The IMU's yaml file includes the gyroscope and accelerometer noise model parameters.

Most importantly the datasets are also available as **ROS bag** files, these bag files can be played back in ROS (**Robot Operating System** on which this whole thesis in based upon) to the same topics they have been recorded from, or even remapped to new topics.

Bags are the primary mechanism in ROS for data logging, which indicates that they possess a variety of offline utilization. Researchers employ the bag file tool-chain to record datasets, then visualize, label them, and store them for later use.

7.2 Application Details

For this thesis the open source VINS-Fusion [40] has been used which is a robust and versatile monocular visual-inertial state estimator, it includes open-source system integration, real-time demonstration for drone navigation based on KF.

In other words, VINS-Fusion is a real-time SLAM framework for Monocular Visual-Inertial Systems. It is based on an optimized sliding-window formulation which offers high accuracy VIO. It presents efficient IMU pre-integration with bias correction, automatic estimator initialization, failure detection and recovery, loop detection, and global pose graph optimization, map merge, rolling shutter support. VINS-Fusion is principally intended for state estimation and feedback control of autonomous drones, but it can offer accurate localization for AR (Augmented Reality) applications as well.

VINS-Fusion runs on Linux i.e. Ubuntu 16.04, and is fully integrated with **ROS** and works with Rviz package.

ROS is an open-source, meta-operating system. It offers the services ranges from hardware abstraction, low-level device control, implementation of commonly-used functionality to message-passing between processes, and package management. It also offers tools and libraries for obtaining, building, writing, and running code across multiple computers.

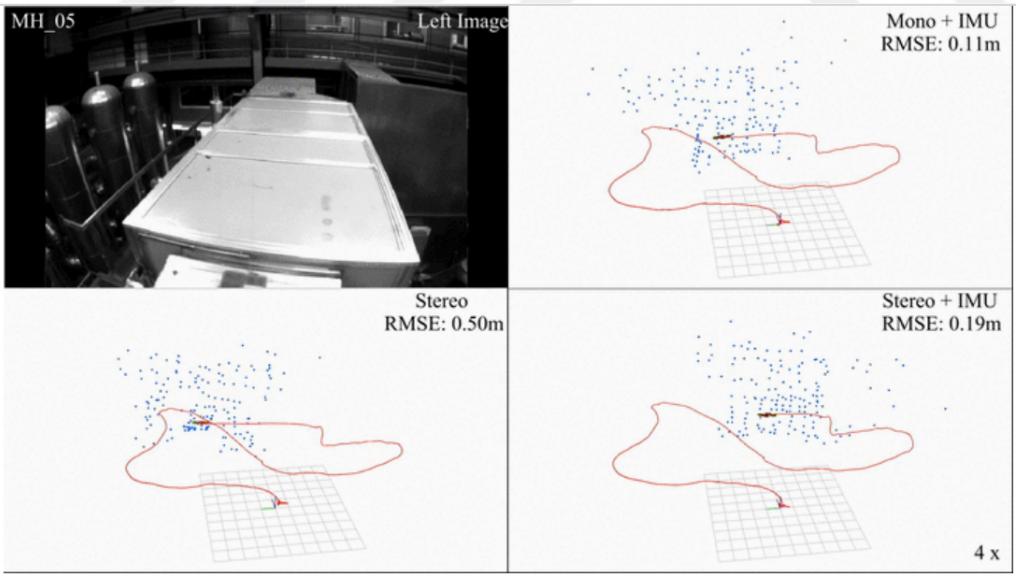


Figure 7.6 : A sample print screen from VINS-Fusion showing the pose estimation applied on MH_5 Difficult dataset using stereo camera system alone, stereo with IMU integrated system, and mono with IMU integrated system. Along with their associated RMSE (Root Mean Square Error)

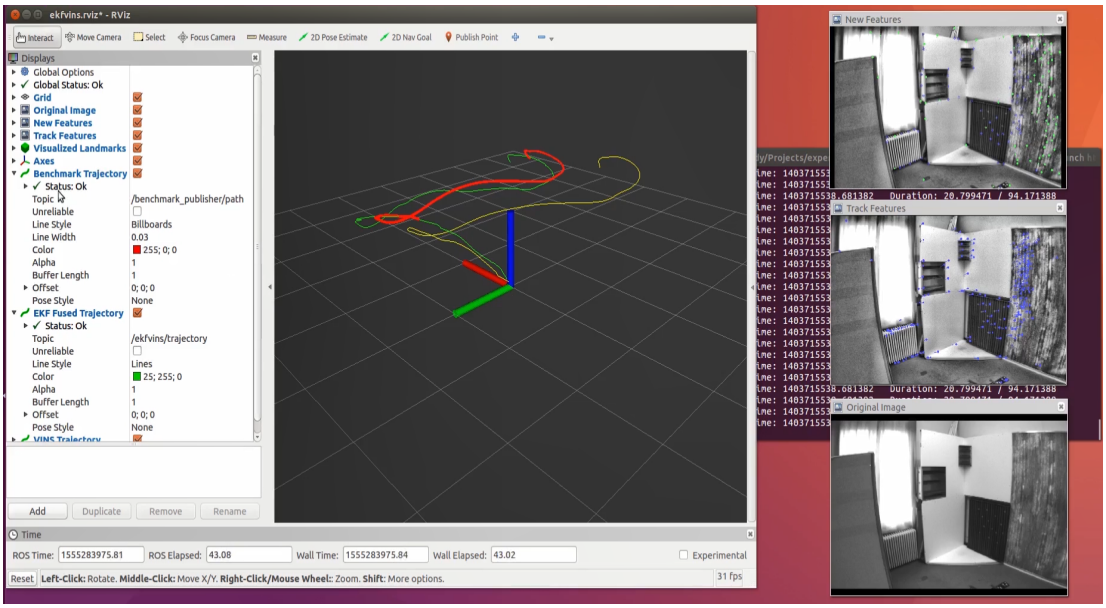


Figure 7.7 : A sample print screen from Rviz while on-run showing in three windows the original image, track features, and the new features along with their estimated graphs

7.3 Codes

I started the algorithm development first in Matlab, but then I rewrote it and did the implementation in C++ to be more efficient and be able to ultimately be ported to ROS, the Robot Operating System [41]. The desktop app **ROS-Rviz** is mostly compatible with C++, because user-interface components are coded in Objective-C. I utilized ROS bag for feature detection and matching and the Eigen-library for all matrix mathematics.

7.3.1 Introduction of EKF-VINS

I worked on an EKF algorithm which is constrained under state estimation. This is an extension of VINS which tries to couple visual and inertial over the current navigation state and the past cloned poses only (i.e., the priori distribution features in the state vector). The key idea is to utilize all the camera tracked features within a time frame to impose nonholonomic constrained among the poses as we go forward in time. These tracked features are computed independently and they are not part of the state estimation vector which is estimated mainly from IMU measurements. Then the

structure from motion (SFM) is computed using the camera data measurements and the structures/features are marginalized for the sigma point generation in EKF.

7.3.2 EKF-VINS Implementation

VINS has quadratic complexity w.r.t. the number of features in the state vector which is inefficient due to higher computational complexity. However, this has been solved by multi-state constraint Kalman filter (MSCKF) framework [25] and it performs VINS over a sliding window of n poses without including features in the state vector, thus having complexity independent of the total number of features observed in the environment. The key idea of the proposed algorithm is to form a novel measurement model that utilizes utilize all the camera tracked features within a time frame to impose nonholonomic constrained among the poses as we go forward in time to impose probabilistically optimal constraints between poses, without estimating these features as part of the state vector.

7.3.3 Sample code

Listing 7.1: The C++ code used for the EKF implementation

```
#include <boost/thread.hpp>
#include <Eigen/Core>
#include <opencv2/core/core.hpp>
#include <nav_msgs/Odometry.h>
#include <nav_msgs/Path.h>
#include <geometry_msgs/PoseStamped.h>
#include <geometry_msgs/TransformStamped.h>
#include "ekfvins/AppEKFFusion.h"
#include "MathsUtil.h"
.
.
.
    /*
     * Initialization
     */
    if (!mbIsInitialized)
    {
        static Eigen::Vector3d wm = Eigen::Vector3d::Zero();
        static Eigen::Vector3d am = Eigen::Vector3d::Zero();
        while (!lImuDataSeq.empty())
        {
            wm += (lImuDataSeq.front())->AngularVel;
            am += (lImuDataSeq.front())->LinearAccel;
            .
            .
            .
    }
    /**
     * Static/Moving detection
     */
    if (!mbIsMoving)
    {
        Eigen::Vector3d angle;
        Eigen::Vector3d displ;
```

```
35          .
36          .
37          .
38          }
39          // If the change is lager than the threshold
40          if (angle.norm()>mnThresholdAngle || displ.norm()>mnThresholdDispl)
41              // Is moving
42              mbIsMoving = true;
43          else
44              // Is static
45              return;
46      }
47      nImageCountAfterMoving++;
48      /**
49       * Visual tracking & Propagation
50       */
51      boost::thread thdTracking(&CamTracker::track, mpCamTracker, im,
52      std::ref(xkk), std::ref(lImuDataSeq));
53      boost::thread thdPropagate(&IMUTransition::propagate,
54      mpIMUTransition, std::ref(xkk), std::ref(Pkk), std::ref(lImuDataSeq));
55
56      thdTracking.join();
57      thdPropagate.join();
58      /**
59       * Update
60       */
61      if (nCloneStates>mnMinCloneStates)
62      {
63          mpUpdater->update(mpIMUTransition->xk1k,
64          mpIMUTransition->Pk1k, mpCamTracker->mvFeatTypesForUpdate,
65          mpCamTracker->mvlFeatMeasForUpdate);
66          xkk = mpUpdater->xk1k1;
67          Pkk = mpUpdater->Pk1k1;
68      }
69      else
70      {
71          xkk = mpIMUTransition->xk1k;
72          Pkk = mpIMUTransition->Pk1k;
73      }
74      /**
75       * State augmentation
76       */
77      if (nImageCountAfterMoving>1)
78      {
79          if (nCloneStates<mnSlidingWindowSize)
80          {
81              // xkk
82              Eigen::MatrixXd tempx(26+7*(nCloneStates+1),1);
83              tempx << xkk, xkk.block(10,0,7,1);
84              xkk = tempx;
85              .
86              .
87              .
88      /**
89       * Composition
90       */
91      Eigen::Vector4d qG = xkk.block(0,0,4,1);
92      Eigen::Vector3d pG = xkk.block(4,0,3,1);
93      .
94      .
95      .
96      /**
97       * Interact with ROS rviz
98       */
99      // Broadcast tf
100     geometry_msgs::TransformStamped transformStamped;
101     transformStamped.header.stamp = ros::Time::now();
102     transformStamped.header.frame_id = "world";
103     transformStamped.child_frame_id = "imu";
104     .
105     .
106     // Publish odometry for benchmarking
107     nav_msgs::Odometry odometry;
108     odometry.header.stamp = ros::Time(timestamp+mnCamTimeOffset);
```

```
109     odometry.header.frame_id = "world";
110     odometry.pose.pose.position.x = pGk(0);
111     .
112     .
113     .
114     // Visualize the trajectory
115     geometry_msgs::PoseStamped pose;
116     pose.header.frame_id = "world";
117     pose.pose.position.x = pGk(0);
118     pose.pose.position.y = pGk(1);
119     pose.pose.position.z = pGk(2);
120     pose.pose.orientation.x = qkG(0);
121     pose.pose.orientation.y = qkG(1);
122     pose.pose.orientation.z = qkG(2);
123     pose.pose.orientation.w = qkG(3);
124     path.header.frame_id = "world";
125     path.poses.push_back(pose);
```

First we estimate the pose and understand the different pose and features tracking using the OpenCV library. For the state updating we are just using the IMU data without camera data. However, for the state augmentation part I used the camera data to add more visual features taken from the image and, then augmenting them to the IMU and finding the current Jacobian to make a better estimation of the odometry, then we compose them both together.

8. RESULTS

In order to validate the performance of the EKF algorithm we developed in various environments, we evaluate the algorithm on Euroc datasets. And by handling the feature extraction and tracking perfectly, we picked sequence V1_01_easy for this purpose to illustrate the performance of the EKF algorithm because the features tracking process was where the features can be tracked comparatively smooth and therefore that made it beneficial to do the comparison between our algorithm and the VINS-fusion algorithm.

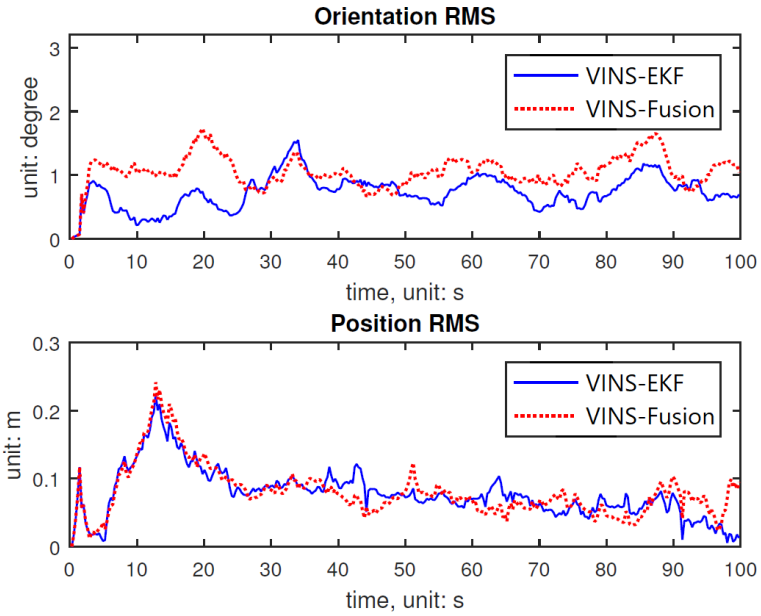


Figure 8.1 : The RMS of position estimate and orientation from VINS-EKF and VINS-Fusion utilizing the Euroc dataset sequence V1_ 01 easy.

After testing the developed VINS-EKF algorithm on all of 11 sequences in EuRoC datasets. We compared the proposed VINS-EKF against the VINS-Fusion, on the keyframe-based visual-inertial SLAM system, performing nonlinear iterative optimization for estimation. The RMSE results after 6-DOF pose alignment are illustrated in Table 8.1. It is significant to note that in general, the VINS-EKF performs better than the VINS-Fusion, unless in couple of sequences and this is due to the fact that they possess less amount of features the environment to be tracked.

Table 8.1 : Estimation accuracy (RMSE) in EuRoC dataset.

		VINS-Fusion		VINS-EKF	
	Length (m)	Orien. (deg)	Pos. (m)	Orien. (deg)	Pos. (m)
V1_01_easy	58.6	2.350	**0.085**	2.151	0.142
V1_02_medium	75.9	3.363	0.299	0.777	**0.156**
V1_03_difficult	79.0	0.586	0.265	0.729	**0.137**
V2_04_easy	36.5	0.651	**0.216**	1.014	0.311
V2_05_medium	83.2	2.986	0.341	1.214	**0.313**
V2_05_difficult	86.1	5.912	0.441	1.275	**0.377**
MH_01_easy	80.6	1.051	**0.387**	1.236	0.590
MH_02_easy	73.5	1.062	**0.698**	0.946	0.740
MH_03_medium	130.9	2.336	0.550	1.351	**0.358**
MH_04_difficult	91.7	0.286	**0.431**	3.525	1.037
MH_05_difficult	97.6	1.136	0.858	1.392	**0.674**

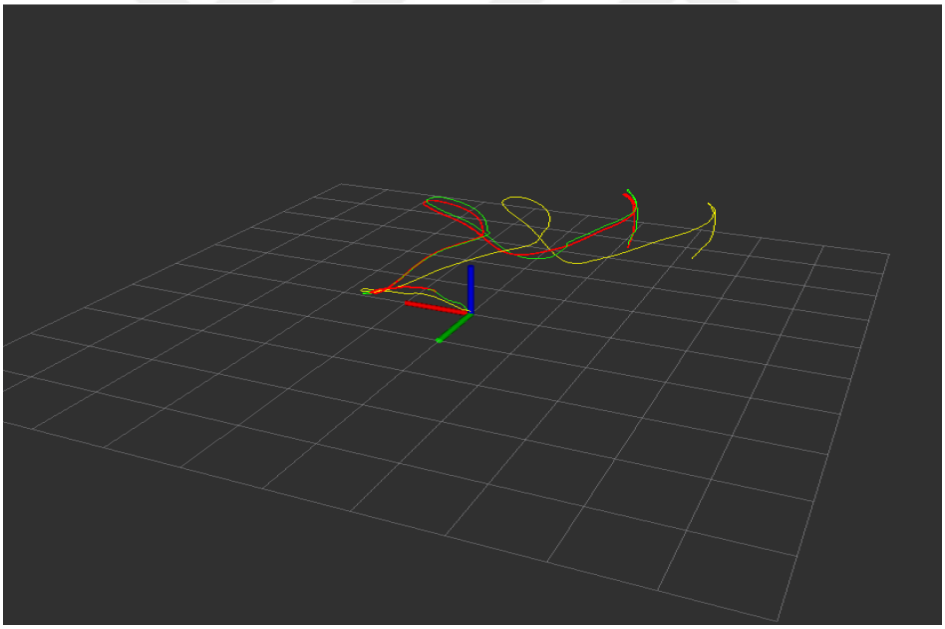


Figure 8.2 : A figure shows three trajectories Green: EKF Fused Trajectory, Red: Benchmark Trajectory, and Yellow: VINS-Fusion Trajectory.

9. CONCLUSION

In this thesis, we displayed the outcomes of the Extended Kalman Filter for visual inertial odometry on EuRoC datasets. The EKF has been known to out perform the traditional KF in the two aspects accuracy and runtime. The high accuracy is a consequence of two strategies: eliminating feature outliers before utilizing them in the EKF, and waiting till the feature readings are available before estimating the feature depth. The runtime speed of the EKF is constant and more rapid than KF SLAM because features are not added to the state vector. Instead, a sliding window of camera poses is maintained. The disadvantage of this in our implementation is the requirement of constant motion. The problem of VI-sensor drift has been solved by estimating the transformation between world-frame and vision-frame in the EKF. This solution ensures smooth operation of the system.

REFERENCES

[1] **Wikipedia**, (2004), Google driverless car Wikipedia, The Free Encyclopedia, [Online; accessed 22-July-2004].

[2] **Bräunl, T.** (2008). *Localization and Navigation*, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/978-3-540-70534-5_16.

[3] **Engel, J.**, **Koltun, V. and Cremers, D.** (2018). Direct Sparse Odometry, *IEEE Trans. Pattern Anal. Mach. Intell.*, *40*(3), 611–625.

[4] **Chen, L.L. and Nakamura, K.** (1998). Head-centered representation and spatial memory in rat posterior parietal cortex, *Psychobiology*, *26*(2), 119–127, https://doi.org/10.3758/BF03330599.

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

[6] **Hartley, R.I. and Zisserman, A.** (2004). *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, second edition.

[7] **Molyneaux, D.**, **Newcombe, R.**, **Shotton, J.**, **Hodges, S.**, **Freeman, D.**, **Davison, A. and Fitzgibbon** (2011). KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera, ACM, pp.559–568.

[8] **Urzua, S.**, **Munguía, R. and Grau, A.** (2019). Monocular SLAM System for MAVs Aided with Altitude and Range Measurements: a GPS-free Approach, *Journal of Intelligent & Robotic Systems*, *94*(1), 203–217, https://doi.org/10.1007/s10846-018-0775-y.

[9] **Durrant-Whyte, H. and Bailey, T.** (2006). Simultaneous localization and mapping: part I, *IEEE Robotics Automation Magazine*, *13*(2), 99–110.

[10] **Leonard, J.J. and Durrant-Whyte, H.F.** (1991). Simultaneous Map Building and Localization for an Autonomous Mobile Robot, *Proc IEEE Int Workshop on Intelligent Robots and Systems*, Osaka, Japan.

[11] **Klein, G. and Murray, D.** (2007). Parallel Tracking and Mapping for Small AR Workspaces, ISMAR '07, IEEE Computer Society, Washington, DC, USA, pp.1–10, https://doi.org/10.1109/ISMAR.2007.4538852.

[12] **Li, M.**, **Kim, B. and Mourikis, A.I.** (2013). Real-Time Motion Estimation on a Cellphone using Inertial Sensing and a Rolling-Shutter Camera,

Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, pp.4697–4704.

[13] **Davison, A.J.** (2003). Real-Time Simultaneous Localisation and Mapping with a Single Camera, *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, IEEE Computer Society, Washington, DC, USA, pp.1403–, http://dl.acm.org/citation.cfm?id=946247.946734.

[14] **Civera, J.**, **Davison, A.J. and Montiel, J.M.M.** (2008). Inverse Depth Parametrization for Monocular SLAM, *IEEE Transactions on Robotics*, *24*(5), 932–945.

[15] **Nieto, J.**, **Guivant, J. and Nebot, E.** (2006). DenseSLAM: Simultaneous Localization and Dense Mapping, *The International Journal of Robotics Research*, *25*(8), 711–744, https://doi.org/10.1177/0278364906067379, https://doi.org/10.1177/0278364906067379.

[16] **Montemerlo, M.**, **Thrun, S.**, **Koller, D. and Wegbreit, B.** (2002). FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, *In Proceedings of the AAAI National Conference on Artificial Intelligence*, AAAI, pp.593–598.

[17] **Kuemmerle, R.**, **Grisetti, G.**, **Strasdat, H.**, **Konolige, K. and Burgard, W.** (2011). g2o: A General Framework for Graph Optimization, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, pp.3607–3613.

[18] **Davison, A.J.**, **Montiel, J.**, **Strasdat, H. and Konolige, K.**, (2011), Double window optimisation for constant time visual SLAM, https://doi.ieeecomputersociety.org/10.1109/ICCV.2011.6126517.

[19] **A. Newcombe, R.**, **J. Lovegrove, S. and J. Davison, A.** (2011). DTAM: Dense tracking and mapping in real-time, pp.2320–2327.

[20] **Pizzoli, M.**, **Forster, C. and Scaramuzza, D.** (2014). REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time.

[21] **Engel, J.**, **Sturm, J. and Cremers, D.** (2013). Semi-dense Visual Odometry for a Monocular Camera, ICCV '13, IEEE Computer Society, Washington, DC, USA, pp.1449–1456, http://dx.doi.org/10.1109/ICCV.2013.183.

[22] **Strasdat, H.**, **Montiel, J.M.M. and Davison, A.J.** (2010). Scale Drift-Aware Large Scale Monocular SLAM, http://www.roboticsproceedings.org/rss06/p10.html.

[23] **Pinies, P.**, **Lupton, T.**, **Sukkarieh, S. and Tardos, J.D.** (2007). Inertial Aiding of Inverse Depth SLAM using a Monocular Camera, 2797–2802.

[24] **Weiss, S.**, **Achtelik, M.**, **Lynen, S.**, **Achtelik, M.**, **Kneip, L.**, **Chli, M. and Siegwart, R.** (2013). Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium, *J. Field Robotics*, *30*, 803–831.

[25] **Mourikis, A. and Roumeliotis, S.** (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation, *2007 IEEE International Conference on Robotics and Automation, ICRA'07*, Proceedings - IEEE International Conference on Robotics and Automation, pp.3565–3572.

[26] **Li, M. and Mourikis, A.I.** (2013). High-Precision, Consistent EKF-based Visual-Inertial Odometry, *International Journal of Robotics Research*, *32*(6), 690–711.

[27] **Li, M. and Mourikis, A.I.** (2012). Vision-aided Inertial Navigation for Resource-Constrained Systems, *Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS)*, Vilamoura, Portugal, pp.1056–1063.

[28] **Thrun, S.**, **Burgard, W. and Fox, D.** (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press.

[29] **Trawny, N. and I Roumeliotis, S.** (2005). Indirect Kalman Filter for 6D Pose Estimation A Tutorial for Quaternion Algebra Multiple Autonomous Robotic Systems Laboratory Technical Report Number 2005-002 , Rev . 57, *Engineering*.

[30] **Fortun, D.**, **Bouthemy, P. and Kervrann, C.** (2015). Optical flow modeling and computation: A survey, *Computer Vision and Image Understanding*, *134*, 1 – 21, `http://www.sciencedirect.com/science/article/pii/S1077314215000429`, image Understanding for Real-world Distributed Video Networks.

[31] **Huey Lai, W.**, **Kek, S.L. and Gaik, T.** (2017). Solving Nonlinear Least Squares Problem Using Gauss-Newton Method, *International Journal of Innovative Science, Engineering and Technology*, *4*, 258–262.

[32] **Tuytelaars, T. and Mikolajczyk, K.** (2008). K.: Local invariant feature detectors: A survey, *FnT Comp. Graphics and Vision*, 177–280.

[33] **Rosten, E. and Drummond, T.** (2006). Machine Learning for High-Speed Corner Detection, *Computer Vision – ECCV 2006*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.430–443.

[34] **Bradski, G.** (2000). The opencv library, *Dr. Dobb's Journal of Software Tools*, 120; 122–125.

[35] **Lowe, D.G.** (1999). Object Recognition from Local Scale-Invariant Features, *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, IEEE Computer Society, Washington, DC, USA, pp.1150–, `http://dl.acm.org/citation.cfm?id=850924.851523`.

[36] **Bay, H.**, **Tuytelaars, T. and Van Gool, L.** SURF: Speeded Up Robust Features, *Computer Vision ECCV 2006*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.404–417.

[37] **Strecha, C.**, **Lepetit, V.**, **Trzcinski, T.**, **Calonder, M.**, **Ã–zuysal, M. and Fua, P.** (2012). BRIEF: Computing a Local Binary Descriptor Very Fast, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *34*(07), 1281–1298.

[38] **Bradski, G.**, **Konolige, K.**, **Rabaud, V. and Rublee, E.** (2011). ORB: An efficient alternative to SIFT or SURF, *2011 IEEE International Conference on Computer Vision (ICCV 2011)*, IEEE Computer Society, Los Alamitos, CA, USA, pp.2564–2571, `https://doi.ieeecomputersociety.org/10.1109/ICCV.2011.6126544`.

[39] **Burri, M.**, **Nikolic, J.**, **Gohl, P.**, **Schneider, T.**, **Rehder, J.**, **Omari, S.**, **Achtelik, M.W. and Siegwart, R.** (2016). The EuRoC micro aerial vehicle datasets, *The International Journal of Robotics Research*, *35*(10), 1157–1163.

[40] **Qin, T.**, **Li, P. and Shen, S.** (2018). VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator, *IEEE Transactions on Robotics*, *34*(4), 1004–1020.

[41] **Quigley, M.**, **Conley, K.**, **Gerkey, B.P.**, **Faust, J.**, **Foote, T.**, **Leibs, J.**, **Wheeler, R. and Ng, A.Y.** (2009). ROS: an open-source Robot Operating System, *ICRA Workshop on Open Source Software*.

APPENDICES

APPENDIX A: Quaternions

APPENDIX A.1: Quaternion Definition

APPENDIX A.1: Small Rotations

APPENDIX B: Error-State Transition Matrix

APPENDIX B.1: Orientation Components

APPENDIX B.2: Velocity Components

APPENDIX B.3: Position Components

APPENDIX B.4: Discrete Implementation

A. Quaternions

A.1. Quaternion Definition

A quaternion is defined as follows

$$\mathbf{q} = \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3 + q_4 \tag{A.1}$$

The numbers $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are *hyperimaginary* numbers.

$$\mathbf{i}^2 = -1 \qquad \mathbf{j}^2 = -1 \qquad \mathbf{k}^2 = -1 \tag{A.2}$$

$$-\mathbf{ij} = \mathbf{ji} = \mathbf{k} \qquad -\mathbf{jk} = \mathbf{kj} = \mathbf{i} \qquad -\mathbf{ki} = \mathbf{ik} = \mathbf{j} \tag{A.3}$$

This convention is different form the original quaternion convention from Hamilton. The main motivation for this convention is the resulting natural and intuitive multiplication order for quaternions.

$${}^B_G\bar{\mathbf{q}} = {}^B_A\bar{\mathbf{q}} \otimes {}^A_G\bar{\mathbf{q}} \tag{A.4}$$

The $\otimes$ operator describes quaternion multiplication.

$$\bar{\mathbf{q}} \otimes \bar{\mathbf{p}} = (q_4 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k})(p_4 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}) \tag{A.5}$$

$$= \begin{bmatrix} q_4p_1 + q_3p_2 - q_2p_3 + q_1p_4 \\ -q_3p_1 + q_4p_2 + q_1p_3 + q_2p_4 \\ q_2p_1 - q_1p_2 + q_4p_3 + q_3p_4 \\ -q_1p_1 - q_2p_2 - q_3p_3 + q_4p_4 \end{bmatrix} \tag{A.6}$$

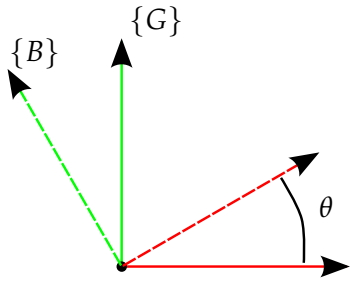
The function $\mathbf{C}(\bar{\mathbf{q}})$ transforms the unit quaternion into a rotation matrix.

$${}^B_G\mathbf{C}(\bar{\mathbf{q}}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{A.7}$$

A.1.1. Axis Angle Representation

A unit quaternion can be represented by a three dimensional unit vector, $\mathbf{k}$ and an angle of rotation, θ around this vector.

$$\bar{\mathbf{q}} = \begin{bmatrix} k_x \sin(\theta/2) \\ k_y \sin(\theta/2) \\ k_z \sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix} \tag{A.8}$$



$$ {}^{B}_{G}\mathbf{R} = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{B}_{G}\mathbf{q} = \begin{bmatrix} 0 \\ 0 \\ sin(\theta/2) \\ cos(\theta/2) \end{bmatrix} $$

Figure A.1. A simple example of the rotation and quaternion convention from [39]

A.2. Small Rotations

Consider a very small quaternion $\delta\bar{\mathbf{q}}$. If we use the axis angle represenation, this can be written as the following

$$ \delta\bar{\mathbf{q}} = \begin{bmatrix} \mathbf{k}\ \sin(\delta\theta/2) \\ \cos(\delta\theta/2) \end{bmatrix} \tag{A.9} $$

$$ \approx \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix} \tag{A.10} $$

In the above expression, the bold term $\delta\boldsymbol{\theta}$ is the minimal 3DoF euler angle error. The approximation results in a convenient approximation for the associated rotation matrix.

$$ {}^{B}_{G}\mathbf{C}(\delta\bar{\mathbf{q}}) \approx \mathbf{I}_3 - \lfloor\delta\boldsymbol{\theta}\times\rfloor \tag{A.11} $$

B. Error-State Transition Matrix

In this section we derive the full error-state transition matrix including bias terms and IMU calibration terms. We base our derivation on that of the error-state transition matrix without the IMU calibration terms found in [23]. The covariance propagation during the prediction step from timestep t_ℓ to $t_{\ell+1}$ has the following sparse form

$$\mathbf{\Sigma}_{\ell+1} = \begin{bmatrix} \mathbf{\Phi}_{B_\ell} & \mathbf{\Gamma}_{imu_\ell} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{\Sigma}_\ell \begin{bmatrix} \mathbf{\Phi}_{B_\ell} & \mathbf{\Gamma}_{imu_\ell} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}^T + \begin{bmatrix} \mathbf{Q}_\ell & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{B.1}$$

Closer inspection of $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ reveal that the first 9 rows contain all the important information.

$$\mathbf{\Phi}_{B_\ell} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{\Phi}_{qb_g} & \mathbf{\Phi}_{qb_a} \\ \mathbf{\Phi}_{pq} & \mathbf{I}_3 & \mathbf{I}_3\Delta t & \mathbf{\Phi}_{pb_g} & \mathbf{\Phi}_{pb_a} \\ \mathbf{\Phi}_{vq} & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{\Phi}_{vb_g} & \mathbf{\Phi}_{vb_a} \\ \mathbf{0_3} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0_3} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \quad \mathbf{\Gamma}_{imu_\ell} = \begin{bmatrix} \mathbf{\Gamma}_{q\vec{\mathbf{T}}_g} & \mathbf{\Gamma}_{q\vec{\mathbf{T}}_s} & \mathbf{\Gamma}_{q\vec{\mathbf{T}}_a} \\ \mathbf{\Gamma}_{p\vec{\mathbf{T}}_g} & \mathbf{\Gamma}_{p\vec{\mathbf{T}}_s} & \mathbf{\Gamma}_{p\vec{\mathbf{T}}_a} \\ \mathbf{\Gamma}_{v\vec{\mathbf{T}}_g} & \mathbf{\Gamma}_{v\vec{\mathbf{T}}_s} & \mathbf{\Gamma}_{v\vec{\mathbf{T}}_a} \\ \mathbf{0_{3\times 9}} & \mathbf{0_{3\times 9}} & \mathbf{0_{3\times 9}} \\ \mathbf{0_{3\times 9}} & \mathbf{0_{3\times 9}} & \mathbf{0_{3\times 9}} \end{bmatrix} \tag{B.2}$$

We can write the relevant terms of the error transition in three equations. These describe how the state error evolves during propagation. In order to improve readability, we write ${}^{B_{\ell|\ell}}_{G}\hat{\mathbf{R}}$ as $\hat{\mathbf{R}}_\ell$ and ${}^{B_{\ell+1|\ell}}_{G}\hat{\mathbf{R}}$ as $\hat{\mathbf{R}}_{\ell+1}$. We also write ${}^G\tilde{\mathbf{p}}_{B_\ell}$ as ${}^G\tilde{\mathbf{p}}_\ell$.

$${}^G\tilde{\boldsymbol{\theta}}_{\ell+1} = {}^G\tilde{\boldsymbol{\theta}}_\ell + \mathbf{\Phi}_{qb_g}\tilde{\mathbf{b}}_{g_\ell} + \mathbf{\Phi}_{qb_a}\tilde{\mathbf{b}}_{a_\ell} + \mathbf{\Gamma}_{q\vec{\mathbf{T}}_g}\tilde{\mathbf{T}}_{g_\ell} + \mathbf{\Gamma}_{q\vec{\mathbf{T}}_s}\tilde{\mathbf{T}}_{s_\ell} + \mathbf{\Gamma}_{q\vec{\mathbf{T}}_a}\tilde{\mathbf{T}}_{a_\ell} \tag{B.3}$$

$${}^G\tilde{\mathbf{p}}_{\ell+1} = \mathbf{\Phi}_{pq}{}^G\tilde{\boldsymbol{\theta}}_\ell + {}^G\tilde{\mathbf{p}}_\ell + {}^G\tilde{\mathbf{v}}_\ell\Delta t + \mathbf{\Phi}_{pb_g}\tilde{\mathbf{b}}_{g_\ell} + \mathbf{\Phi}_{pb_a}\tilde{\mathbf{b}}_{a_\ell} + \mathbf{\Gamma}_{p\vec{\mathbf{T}}_g}\tilde{\mathbf{T}}_{g_\ell} + \mathbf{\Gamma}_{p\vec{\mathbf{T}}_s}\tilde{\mathbf{T}}_{s_\ell} + \mathbf{\Gamma}_{p\vec{\mathbf{T}}_a}\tilde{\mathbf{T}}_{a_\ell} \tag{B.4}$$

$${}^G\tilde{\mathbf{v}}_{\ell+1} = \mathbf{\Phi}_{vq}{}^G\tilde{\boldsymbol{\theta}}_\ell + {}^G\tilde{\mathbf{v}}_\ell + \mathbf{\Phi}_{vb_g}\tilde{\mathbf{b}}_{g_\ell} + \mathbf{\Phi}_{vb_a}\tilde{\mathbf{b}}_{a_\ell} + \mathbf{\Gamma}_{v\vec{\mathbf{T}}_g}\tilde{\mathbf{T}}_{g_\ell} + \mathbf{\Gamma}_{v\vec{\mathbf{T}}_s}\tilde{\mathbf{T}}_{s_\ell} + \mathbf{\Gamma}_{v\vec{\mathbf{T}}_a}\tilde{\mathbf{T}}_{a_\ell} \tag{B.5}$$

To find these error terms, we begin with equation (6.38) which we write here again.

$$\begin{bmatrix} {}^G\tilde{\boldsymbol{\theta}}_{\ell+1} \\ {}^G\tilde{\mathbf{p}}_{\ell+1} \\ {}^G\tilde{\mathbf{v}}_{\ell+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{\Phi}_{pq} & \mathbf{I}_3 & \mathbf{I}_3\Delta t \\ \mathbf{\Phi}_{vq} & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} {}^G\tilde{\boldsymbol{\theta}}_\ell \\ {}^G\tilde{\mathbf{p}}_\ell \\ {}^G\tilde{\mathbf{v}}_\ell \end{bmatrix} + \begin{bmatrix} \hat{\mathbf{R}}_\ell^T\,{}^{B_\ell}\tilde{\boldsymbol{\theta}}_{\Delta\ell} \\ \hat{\mathbf{R}}_\ell^T\,\tilde{\mathbf{y}}_{\Delta\ell} \\ \hat{\mathbf{R}}_\ell^T\,\tilde{\mathbf{s}}_{\Delta\ell} \end{bmatrix} \tag{B.6}$$

B.1. Orientation Components

From [23] we have the definition of ${}^G\tilde{\boldsymbol{\theta}}_{\Delta\ell}$.

$${}^G\tilde{\boldsymbol{\theta}}_{\Delta\ell} \simeq \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\mathbf{R}\,{}^{B_\tau}\tilde{\boldsymbol{\omega}}\,d\tau \tag{B.7}$$

The true rotational velocity of the body, ${}^{B}\boldsymbol{\omega}$, in terms of the measured value and the true IMU calibration parameters, is as follows.

$$ {}^{B}\boldsymbol{\omega} = \mathbf{T}_g^{-1} \left({}^{B}\boldsymbol{\omega}_m - \mathbf{T}_s \, {}^{B}\mathbf{a} - \mathbf{b}_g - \mathbf{n}_g \right) \tag{B.8} $$

Here $\mathbf{b}_g$ and $\mathbf{n}_g$ are the gyroscope bias and noise, respectively, and ${}^{B}\mathbf{a}$ is the acceleration in the body frame.

$$ {}^{B}\mathbf{a} = \mathbf{T}_a^{-1} \left({}^{B}\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a \right) \tag{B.9} $$

We can write the rotational velocity error in terms of the state error using the Taylor expansion.

$$ {}^{B}\tilde{\boldsymbol{\omega}} = {}^{B}\boldsymbol{\omega} - {}^{B}\hat{\boldsymbol{\omega}} \tag{B.10} $$

$$ {}^{B}\boldsymbol{\omega} \simeq {}^{B}\hat{\boldsymbol{\omega}} + \frac{\partial {}^{B}\boldsymbol{\omega}}{\partial \mathbf{x}} \tilde{\mathbf{x}} \tag{B.11} $$

$$ {}^{B}\tilde{\boldsymbol{\omega}} \simeq \frac{\partial {}^{B}\boldsymbol{\omega}}{\partial \mathbf{x}} \tilde{\mathbf{x}} \tag{B.12} $$

Substituting this into equation (B.7) gives us

$$ {}^{G}\tilde{\boldsymbol{\theta}}_{\Delta\ell} \simeq \hat{\mathbf{R}}_{\ell+1} \hat{\mathbf{R}}_{\ell}^{T} \int_{t_\ell}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\mathbf{R} \, \frac{\partial {}^{B}\boldsymbol{\omega}(\tau)}{\partial \mathbf{x}} \tilde{\mathbf{x}} \, d\tau \tag{B.13} $$

We proceed by calculating the partial derivatives of ${}^{B}\boldsymbol{\omega}$ with respect to the state vector. Because the three $\mathbf{T}$ matrices are stored in vectorized format, they must be differentiated element-wise. For example, the first column of the 3×9 matrix $\boldsymbol{\Gamma}_{q\vec{\mathbf{T}}_g}$ will use the first element of $\frac{\partial \mathbf{T}_g}{\partial \vec{\mathbf{T}}_g}$ shown here.

$$ \frac{\partial \mathbf{T}_g}{\partial \vec{\mathbf{T}}_g} = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \dots & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix} \tag{B.14} $$

These terms therefore must be evaluated separately for each of the 9 columns. This is because there is no matrix multiplication operation that can transform a 9×1 vector into a 3×3 matrix.

$$ \frac{\partial {}^{B}\boldsymbol{\omega}}{\partial \mathbf{b}_g} = -\mathbf{T}_g^{-1} \tag{B.15} $$

$$ \frac{\partial {}^{B}\boldsymbol{\omega}}{\partial \mathbf{b}_a} = \mathbf{T}_g^{-1} \, \mathbf{T}_s \, \mathbf{T}_a^{-1} \tag{B.16} $$

$$ \frac{\partial {}^{B}\boldsymbol{\omega}}{\partial \mathbf{T}_g} = -\mathbf{T}_g^{-1} \, \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_g} \, \mathbf{T}_g^{-1} \left({}^{B}\boldsymbol{\omega}_m - \mathbf{T}_s \, {}^{B}\mathbf{a} - \mathbf{b}_g \right) \tag{B.17} $$

$$ \frac{\partial {}^{B}\boldsymbol{\omega}}{\partial \mathbf{T}_s} = -\mathbf{T}_g^{-1} \, \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_s} \, {}^{B}\mathbf{a} \tag{B.18} $$

$$ \frac{\partial {}^{B}\boldsymbol{\omega}}{\partial \mathbf{T}_a} = -\mathbf{T}_g^{-1} \, \mathbf{T}_s \, \mathbf{T}_a^{-1} \, \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_a} \, \mathbf{T}_a^{-1} \left({}^{B}\mathbf{a}_m - \mathbf{b}_a \right) \tag{B.19} $$

Combining these with equations (B.13) and (6.38) give the error terms. When using these in practice, we employ trapezoidal integration.

$$\mathbf{\Phi}_{qb_g} = -\hat{\mathbf{R}}_\ell^T \int_{t_\tau}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, d\tau \tag{B.20}$$

$$\mathbf{\Phi}_{qb_a} = \hat{\mathbf{R}}_\ell^T \int_{t_\tau}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \hat{\mathbf{T}}_s\, \hat{\mathbf{T}}_a^{-1}\, d\tau \tag{B.21}$$

$$\mathbf{\Gamma}_{q\vec{\mathbf{T}}_g} = -\hat{\mathbf{R}}_\ell^T \int_{t_\tau}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_g}\, \hat{\mathbf{T}}_g^{-1} \left({}^B\boldsymbol{\omega}_m(\tau) - \hat{\mathbf{T}}_s\, {}^B\hat{\mathbf{a}} - \hat{\mathbf{b}}_g \right) d\tau \tag{B.22}$$

$$\mathbf{\Gamma}_{q\vec{\mathbf{T}}_s} = -\hat{\mathbf{R}}_\ell^T \int_{t_\tau}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_s}\, {}^B\hat{\mathbf{a}}(\tau)\, d\tau \tag{B.23}$$

$$\mathbf{\Gamma}_{q\vec{\mathbf{T}}_a} = \hat{\mathbf{R}}_\ell^T \int_{t_\tau}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \hat{\mathbf{T}}_s\, \hat{\mathbf{T}}_a^{-1}\, \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_a}\, \hat{\mathbf{T}}_a^{-1} \left({}^B\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_a \right) d\tau \tag{B.24}$$

$$\tag{B.25}$$

B.2. Velocity Components

We turn now to the velocity components of the error-state transition matrix. Using equation (6.38) we begin with the definition of $\tilde{\mathbf{s}}$.

$$\tilde{\mathbf{s}}_{\Delta\ell} = \int_{t_\ell}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\mathbf{R}\, {}^{B_\tau}\mathbf{a}\, d\tau - \int_{t_\ell}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\hat{\mathbf{a}}\, d\tau \tag{B.26}$$

Using the approximation ${}^{B_\ell}_{B_\tau}\mathbf{R} \simeq \left(\mathbf{I}_3 - \lfloor {}^B\tilde{\boldsymbol{\theta}}_{\Delta\tau}\times \rfloor\right) {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}$, we get the following.

$$\tilde{\mathbf{s}}_{\Delta\ell} \simeq \int_{t_\ell}^{t_{\ell+1}} \left(\mathbf{I}_3 - \lfloor {}^B\tilde{\boldsymbol{\theta}}_{\Delta\tau}\times \rfloor\right) {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\mathbf{a}\, d\tau - \int_{t_\ell}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\hat{\mathbf{a}}\, d\tau \tag{B.27}$$

$$= \int_{t_\ell}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\tilde{\mathbf{a}}\, d\tau - \int_{t_\ell}^{t_{\ell+1}} \lfloor {}^B\tilde{\boldsymbol{\theta}}_{\Delta\tau}\times \rfloor {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\hat{\mathbf{a}}\, d\tau \tag{B.28}$$

$$= \int_{t_\ell}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\tilde{\mathbf{a}}\, d\tau + \int_{t_\ell}^{t_{\ell+1}} \lfloor {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\hat{\mathbf{a}}\times \rfloor\, {}^B\tilde{\boldsymbol{\theta}}_{\Delta\tau}\, d\tau \tag{B.29}$$

$$= \int_{t_\ell}^{t_{\ell+1}} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\tilde{\mathbf{a}}\, d\tau + \int_{t_\ell}^{t_{\ell+1}} \hat{\mathbf{R}}_\ell \lfloor {}^G\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T\, {}^B\tilde{\boldsymbol{\theta}}_{\Delta\tau}\, d\tau \tag{B.30}$$

Just as with the rotational velocity, we make use of the Taylor expansion to subsitute the acceleration error term with the state error.

$$\tilde{\mathbf{s}} = \int_{t_\ell}^{t_{\ell+1}} \overset{B_\ell}{B_\tau}\hat{\mathbf{R}} \frac{\partial {}^B\mathbf{a}(\tau)}{\partial \mathbf{x}} \tilde{\mathbf{x}}\, d\tau + \int_{t_\ell}^{t_{\ell+1}} \hat{\mathbf{R}}_\ell \lfloor {}^G\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T\, {}^B\tilde{\boldsymbol{\theta}}_{\Delta\tau}\, d\tau \tag{B.31}$$

We proceed by differentiating ${}^B\mathbf{a}$ according to the state vector.

$$\frac{\partial {}^B\mathbf{a}}{\partial \mathbf{b}_g} = \mathbf{0} \tag{B.32}$$

$$\frac{\partial {}^B\mathbf{a}}{\partial \mathbf{b}_a} = -\mathbf{T}_a^{-1} \tag{B.33}$$

$$\frac{\partial {}^B\mathbf{a}}{\partial \mathbf{T}_g} = \mathbf{0} \tag{B.34}$$

$$\frac{\partial {}^B\mathbf{a}}{\partial \mathbf{T}_s} = \mathbf{0} \tag{B.35}$$

$$\frac{\partial {}^B\mathbf{a}}{\partial \mathbf{T}_a} = -\mathbf{T}_a^{-1} \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_a} \mathbf{T}_a^{-1} \left({}^B\mathbf{a}_m - \mathbf{b}_a \right) \tag{B.36}$$

Substitution with equation (B.31) and (6.38), along with ${}^B\tilde{\boldsymbol{\theta}}_{\Delta\tau}$ and the orientation results above gives us the error-state transition components.

$$\boldsymbol{\Phi}_{vb_g} = -\int_{t_\ell}^{t_{\ell+1}} \lfloor {}^G\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{\tau} \overset{B_\ell}{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, ds\, d\tau \tag{B.37}$$

$$\boldsymbol{\Phi}_{vb_a} = -\mathbf{R}_\ell^T \int_{t_\ell}^{t_{\ell+1}} \overset{B_\ell}{B_\tau}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_a^{-1}\, d\tau + \int_{t_\ell}^{t_{\ell+1}} \lfloor {}^G\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{\tau} \overset{B_\ell}{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \hat{\mathbf{T}}_s\, \hat{\mathbf{T}}_a^{-1}\, ds\, d\tau \tag{B.38}$$

$$\boldsymbol{\Gamma}_{v\vec{\mathbf{T}}_g} = -\mathbf{R}_\ell^T \int_{t_\ell}^{t_{\ell+1}} \hat{\mathbf{R}}_\ell \lfloor {}^G\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{t_\tau} \overset{B_\ell}{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1} \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_g}\, {}^B\hat{\boldsymbol{\omega}}(s)\, ds\, d\tau \tag{B.39}$$

$$\boldsymbol{\Gamma}_{v\vec{\mathbf{T}}_s} = -\int_{t_\ell}^{t_{\ell+1}} \lfloor {}^G\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{t_\tau} \overset{B_\ell}{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1} \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_s}\, {}^B\hat{\mathbf{a}}(s)\, ds\, d\tau \tag{B.40}$$

$$\begin{aligned}\boldsymbol{\Gamma}_{v\vec{\mathbf{T}}_a} = &-\mathbf{R}_\ell^T \int_{t_\ell}^{t_{\ell+1}} \overset{B_\ell}{B_\tau}\hat{\mathbf{R}}\mathbf{T}_a^{-1} \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_a} \mathbf{T}_a^{-1} \left({}^B\mathbf{a}_m(\tau) - \mathbf{b}_a \right) d\tau \\ &+ \int_{t_\ell}^{t_{\ell+1}} \lfloor {}^G\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{t_\tau} \overset{B_\ell}{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \hat{\mathbf{T}}_s\, \hat{\mathbf{T}}_a^{-1} \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_a} \hat{\mathbf{T}}_a^{-1} \left({}^B\mathbf{a}_m(s) - \hat{\mathbf{b}}_a \right) ds\, d\tau\end{aligned} \tag{B.41}$$

With ${}^G\hat{\mathbf{a}} = {}^G\dot{\hat{\mathbf{v}}} - {}^G\mathbf{g}$ for gravity vector ${}^G\mathbf{g}$.

B.3. Position Components

Here we show the bias and IMU calibration terms of the error-state transition matrix relating to the position. Using equation (6.38) we begin with the definition of $\hat{\mathbf{y}}$.

$$\hat{\mathbf{y}} = \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{s} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, {}^{B_\tau}\hat{\mathbf{a}}\, d\tau\, ds \tag{B.42}$$

Because $\mathbf{y}$ is the integral of $\mathbf{s}$ it is clear that $\tilde{\mathbf{y}}$ has the following form.

$$\tilde{\mathbf{y}}_{\Delta\ell} = \int_{t_\ell}^{t_{\ell+1}} \tilde{\mathbf{s}}(\tau)\, d\tau \tag{B.43}$$

With this definition, we can use the previously calculated velocity components and integrate them one more time. This is intuitively clear because the position is the integral of the velocity.

$$\mathbf{\Phi}_{pb_g} = -\int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{w} \lfloor {}^{G}\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{\tau} {}^{B_\ell}_{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, ds\, d\tau\, dw \tag{B.44}$$

$$\mathbf{\Phi}_{pb_a} = -\mathbf{R}_\ell^T \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{w} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_a^{-1}\, d\tau\, dw + \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{w} \lfloor {}^{G}\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{\tau} {}^{B_\ell}_{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \hat{\mathbf{T}}_s\, \hat{\mathbf{T}}_a^{-1}\, ds\, d\tau\, dw \tag{B.45}$$

$$\mathbf{\Gamma}_{p\vec{\mathbf{T}}_g} = -\mathbf{R}_\ell^T \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{w} \hat{\mathbf{R}}_\ell \lfloor {}^{G}\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{t_\tau} {}^{B_\ell}_{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_g}\, {}^{B}\hat{\boldsymbol{\omega}}(s)\, ds\, d\tau\, dw \tag{B.46}$$

$$\mathbf{\Gamma}_{p\vec{\mathbf{T}}_s} = -\int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{w} \lfloor {}^{G}\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{t_\tau} {}^{B_\ell}_{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_s}\, {}^{B}\hat{\mathbf{a}}(s)\, ds\, d\tau\, dw \tag{B.47}$$

$$\begin{aligned}\mathbf{\Gamma}_{p\vec{\mathbf{T}}_a} = &-\mathbf{R}_\ell^T \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{w} {}^{B_\ell}_{B_\tau}\hat{\mathbf{R}}\mathbf{T}_a^{-1} \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_a}\, \mathbf{T}_a^{-1} \left({}^{B}\mathbf{a}_m(\tau) - \mathbf{b}_a \right) d\tau\, dw \\ &+ \int_{t_\ell}^{t_{\ell+1}} \int_{t_\ell}^{w} \lfloor {}^{G}\hat{\mathbf{a}}(\tau)\times \rfloor \hat{\mathbf{R}}_\ell^T \int_{t_\ell}^{t_\tau} {}^{B_\ell}_{B_s}\hat{\mathbf{R}}\, \hat{\mathbf{T}}_g^{-1}\, \hat{\mathbf{T}}_s\, \hat{\mathbf{T}}_a^{-1} \frac{\partial \mathbf{T}}{\partial \vec{\mathbf{T}}_a}\, \hat{\mathbf{T}}_a^{-1} \left({}^{B}\mathbf{a}_m(s) - \hat{\mathbf{b}}_a \right) ds\, d\tau\, dw\end{aligned} \tag{B.48}$$

B.4. Discrete Implementation

To use the error-state transition matrix in practice, we use trapezoidal integration with the following approximate results.

$$\mathbf{\Phi}_{qb_g} \simeq -\frac{\Delta t}{2}\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1} \tag{B.49}$$

$$\mathbf{\Phi}_{vb_g} \simeq \frac{\Delta t^2}{4}\left(\lfloor({}^G\mathbf{a}_{\ell+1} - {}^G\mathbf{g})\times\rfloor)(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T)\right)\hat{\mathbf{T}}_g^{-1} \tag{B.50}$$

$$\mathbf{\Phi}_{pb_g} \simeq \frac{\Delta t}{2}\mathbf{\Phi}_{vb_g} \tag{B.51}$$

$$\mathbf{\Phi}_{qb_a} \simeq \frac{\Delta t}{2}\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1}\hat{\mathbf{T}}_s\hat{\mathbf{T}}_a^{-1} \tag{B.52}$$

$$\mathbf{\Phi}_{vb_a} \simeq -\frac{\Delta t}{2}\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_a^{-1} + \frac{\Delta t^2}{4}\lfloor{}^G\mathbf{a}_{\ell+1} - {}^G\mathbf{g}\times\rfloor\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1}\hat{\mathbf{T}}_s\hat{\mathbf{T}}_a^{-1} \tag{B.53}$$

$$\mathbf{\Phi}_{pb_a} \simeq \frac{\Delta t}{2}\mathbf{\Phi}_{vb_a} \tag{B.54}$$

$$\mathbf{\Gamma}_{q\vec{\mathbf{T}}_g} \simeq -\frac{\Delta t}{2}\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1}\frac{\partial \mathbf{T}_g}{\partial \vec{\mathbf{T}}_g}\,{}^B\hat{\boldsymbol{\omega}}_{\ell+1} \tag{B.55}$$

$$\mathbf{\Gamma}_{v\vec{\mathbf{T}}_g} \simeq -\frac{\Delta t^2}{4}\lfloor{}^G\mathbf{a}_{\ell+1} - {}^G\mathbf{g}\times\rfloor\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1}\frac{\partial \mathbf{T}_g}{\partial \vec{\mathbf{T}}_g}\,{}^B\hat{\boldsymbol{\omega}}_{\ell+1} \tag{B.56}$$

$$\mathbf{\Gamma}_{p\vec{\mathbf{T}}_g} \simeq \frac{\Delta t}{2}\mathbf{\Gamma}_{v\vec{\mathbf{T}}_g} \tag{B.57}$$

$$\mathbf{\Gamma}_{q\vec{\mathbf{T}}_s} \simeq -\frac{\Delta t}{2}\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1}\frac{\partial \mathbf{T}_s}{\partial \vec{\mathbf{T}}_s}\,{}^B\hat{\mathbf{a}}_{\ell+1} \tag{B.58}$$

$$\mathbf{\Gamma}_{v\vec{\mathbf{T}}_s} \simeq -\frac{\Delta t^2}{4}\lfloor{}^G\mathbf{a}_{\ell+1} - {}^G\mathbf{g}\times\rfloor\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1}\frac{\partial \mathbf{T}_s}{\partial \vec{\mathbf{T}}_s}\,{}^B\hat{\mathbf{a}}_{\ell+1} \tag{B.59}$$

$$\mathbf{\Gamma}_{p\vec{\mathbf{T}}_s} \simeq \frac{\Delta t}{2}\mathbf{\Gamma}_{v\vec{\mathbf{T}}_s} \tag{B.60}$$

$$\mathbf{\Gamma}_{q\vec{\mathbf{T}}_a} \simeq \frac{\Delta t}{2}\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1}\hat{\mathbf{T}}_s\hat{\mathbf{T}}_a^{-1}\frac{\partial \mathbf{T}_a}{\partial \vec{\mathbf{T}}_a}\,{}^B\hat{\mathbf{a}}_{\ell+1} \tag{B.61}$$

$$\begin{aligned}\mathbf{\Gamma}_{v\vec{\mathbf{T}}_a} \simeq{}& \frac{\Delta t^2}{4}\lfloor{}^G\mathbf{a}_{\ell+1} - {}^G\mathbf{g}\times\rfloor\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_g^{-1}\hat{\mathbf{T}}_s\hat{\mathbf{T}}_a^{-1}\frac{\partial \mathbf{T}_a}{\partial \vec{\mathbf{T}}_a}\,{}^B\hat{\mathbf{a}}_{\ell+1} \\ &- \frac{\Delta t}{2}\left(\hat{\mathbf{R}}_{\ell+1}^T + \hat{\mathbf{R}}_{\ell}^T\right)\hat{\mathbf{T}}_a^{-1}\frac{\partial \mathbf{T}_a}{\partial \vec{\mathbf{T}}_a}\,{}^B\hat{\mathbf{a}}_{\ell+1}\end{aligned} \tag{B.62}$$

$$\mathbf{\Gamma}_{p\vec{\mathbf{T}}_a} \simeq \frac{\Delta t}{2}\mathbf{\Gamma}_{v\vec{\mathbf{T}}_a} \tag{B.63}$$

$$\tag{B.64}$$

CURRICULUM VITAE

Name Surname: Abdel Salam Bawarshi

Place and Date of Birth: Damascus - 02.Apr.1990

E-Mail: bawarshi15@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2014, Damascus University, Faculty of Mechanical and Electrical Engineering, Department of General Mechanical Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- *2010* Internship Program as a Mechanical Engineering Technician at Jasmine Hills Project, Rural Damascus/Syria.
- *2010-2014* Mechanical Engineer, Production and Operations Department at Al-Wasim Factory, Damascus/Syria.
- *2016-2017* Translator/Interpreter at Maya Vakfı - Balat Field Office, Istanbul/-Turkey.
- *2018-Present* Graduate Research Assistant at İstanbul Technical University ARC (Aerospace Research Center), Istanbul/Turkey.

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Bawarshi S. Abdel, Allaf A.
  2012. Review on Thermal Energy Storage with Phase Change: Materials, Heat Transfer Analysis and Applications. *Damascus University - Journal of Engineering Sciences*, March 22-24, 2007 Antalya, Turkey.